

Intel® NetStructure™ ZT 8102 Gigabit Ethernet Switch

Technical Product Specification

May 2003

Order Number: 273876-001



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined.” Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Intel® NetStructure™ ZT 8102 Gigabit Ethernet Switch may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This Technical Product Specification as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at http://www.intel.com.

AnyPoint, AppChoice, BoardWatch, BunnyPeople, CablePort, Celeron, Chips, CT Media, Dialogic, DM3, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel Centrino, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Create & Share, Intel GigaBlade, Intel InBusiness, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel Play, Intel Play logo, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel TeamStation, Intel Xeon, Intel XScale, IPLink, Itanium, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PC Dads, PC Parents, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, RemoteExpress, SmartDie, Solutions960, Sound Mark, StorageExpress, The Computer Inside., The Journey Inside, TokenExpress, VoiceBrick, VTune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2003, Intel Corporation

Contents

1 Introduction 8

1.1 PICMG* 2.16 Compliant 8
1.2 ZNYX OpenArchitect* Switch Management 8
1.3 Extensible Customization of Routing Policies 8
1.4 Powerful CarrierClass* Features 9
1.5 OpenArchitect* Software Structure 9
1.6 Features Overview 9
1.6.1 Hardware 9
1.6.2 Layer 2 Switching Functions 10
1.6.3 Layer 3 Switching Functions 10
1.6.4 Management Functions 11
1.6.5 Warranty 11
1.7 Specifications 11
1.7.1 Electrical 11
1.7.2 Mechanical 11
1.7.3 Environmental 11
1.7.4 Safety and Emission (EMI) Certifications 12
1.7.5 Standards 12
1.8 ZT 8102 Functional Blocks 13
1.9 PICMG 2.16 Board Keying 13
1.10 IPMI 13

2 Installation and Initial Setup 16

2.1 Installing the Board 16
2.2 Power on 17
2.3 Uninstalling the Board 17
2.4 Identifying External Components 18
2.5 Connecting the Cables 19
2.5.1 1000BaseSX Cabling 19
2.5.2 RS-232 Console Management Port Cabling 19
2.5.3 Out Of Band (OOB) 10/100 Ethernet Management Port 19
2.6 Status LEDs 19
2.7 ZT 8102 LED Reference 21
2.8 Getting Started with Management 22
2.9 Accessing the Local Console 22
2.9.1 Connecting to the RS-232 Management Console Port 23
2.9.2 To log in to the switch the first time 23
2.9.3 Configuration Management Overview 23
2.9.4 Setting the IP Address for In-Band Management 23
2.9.5 Setting the IP Address for Out-of-Band Management 24

3 Switch Configuration 26

3.1 Switch Configuration Procedure 26
3.2 Example Configuration Scripts 26
3.3 Overview of VLAN Interfaces 27
3.3.1 Tagging and Untagging VLANs 28

3.3.2 Switch Port Interfaces 28
3.4 Layer 2 Switch Configuration 28
3.4.1 Using the Provided S50Layer 2 Script 29
3.5 Spanning Tree 29
3.5.1 To Enable Spanning Tree: 30
3.5.2 Port Path Cost 30
3.5.3 Fast Forward Port 30
3.5.4 Fast Forward Bridge 31
3.5.5 Using the Provided S50Layer 2sp Script 31
3.6 Layer 3 Switch Configuration 31
3.6.1 Using the Provided S50Layer 3 Script 32
3.6.2 To Modify the Layer 3 Script 33
3.6.3 Layer 3 Switch Using Multiple VLANs 33
3.6.4 Using the Provided S50multivlan Script 34
3.6.5 To Modify the Layer 3 Multivlan Script 35
3.7 Layer 3 Routing Protocols with GateD 35
3.7.1 Using the Provided S55gatedRip1 Script 35
3.7.2 To Modify the GateD Scripts: 37

4 Switch Administration 38

4.1 Setting the Root Password 38
4.2 Adding Additional Users 38
4.3 Setting up a Default Route 39
4.4 Name Service Resolution 39
4.5 DHCP Client Configuration 39
4.6 DHCP Server Configuration 39
4.7 Network Time Protocol (NTP) Client Configuration 39
4.8 Network File System (NFS) Client Configuration 40
4.9 NFS Server Configuration 41
4.10 Connecting to the Switch Using FTP 41
4.11 FTPD Server Configuration 41
4.12 Connecting to the Switch Using TFTP 41
4.13 TFTPD Server Configuration 41
4.14 SNMP Agent 42
4.14.1 Supported MIBS 42
4.14.2 Supported Traps 43
4.14.3 SNMP and Switch Interface Definitions 43
4.14.4 ifStackTable Entries 44
4.14.5 SNMP Configuration 44
4.14.6 SNMP Applications 45
4.15 Port Mirroring 45
4.16 Link and LED Control 46
4.17 Link Event Monitoring 46
4.18 Web Interface (thttpd) 46
4.18.1 Starting thttpd 46
4.18.2 Restricting Access 47
4.18.3 Web Interface 47
4.18.4 Customizing the Web Interface 48

5 High Availability (HA) Networking 50
5.1 Advanced Network Services (ANS) for Hosts 50
5.1.1 RAINlink for Hosts 50
5.1.2 Redundant Connections Using ANS or RAINlink 50
5.1.3 VRRP 51
5.2 Switch Surviving Partner 52
5.2.1 ZLMD 53
5.2.2 Switch Replacement and Reconfiguration 53
5.3 Configuring Surviving Partner 54
5.3.1 zspconfig 55
5.3.2 Example HA Chassis Configuration 58
5.4 Central Authority 61

6 Switch Maintenance 64
6.1 Overview of the OpenArchitect switch boot process 64
6.2 Saving Changes 66
6.3 Modifying Files and Updating the Switch 66
6.4 Recovering from a System Failure 66
6.4.1 System Boots with a Console Cable 66
6.5 Booting with the –i option 67
6.6 System Hangs During Boot 67
6.7 Booting the Duplicate Flash Image 67
6.8 Upgrading the OpenArchitect Image 68
6.9 Upgrading or Adding Files 69
6.10 Excluding Saving Files to Flash 69
6.11 Using Apt-Get 69

7 Switch Commands 72

A RS-232 Serial Cable 124
A.1 Pin Assignments 124
A.2 Building the Cable 124

B Real Panel Pinouts 126

C Warranty Information 131
C.1 Intel® NetStructure™ Compute Boards and Platform Products Limited Warranty 131

D Agency Approvals 134
D.1 CE Certification 134
D.2 Safety 134
D.3 Emissions Test Regulations 134
D.4 Regulatory Information 135

E Customer Support 138
E.1 G.1 Technical Support and Return for Service 138
E.2 G.2 Sales Assistance 138

Glossary 140

Figures

1 Intel® NetStructure™ ZT 8102 Functional Block Diagram 13
2 Injector/Ejector Operations 16
3 Intel® NetStructure™ ZT 8102 Front Panel 18
4 LEDs 20
5 Multiple VLANs Example 27
6 Layer 2 Switch 28
7 Layer 3 Switch 32
8 Multiple VLAN Configuration 34
9 Failover Example 51
10 VRRP 51
11 VRRP with Single Point of Failure on LA 52
12 VRRP with HA on LAN 52
13 Failover Scenario 54
14 zspconfig Configuration and Runtime Scripts 55
15 Surviving Partner Overview 57
16 HA Chassis 58
17 HA Example 59
18 ROM Devices in OpenArchitect 64
19 Boot Process Flow 65
20 Init Script Flow 66
21 Serial Cable Diagram 125

Tables

1 Power Requirements 11
2 Front Panel LEDs 21
3 IEEE 802.1D Recommendations 30
4 Administrative Status Example 44
5 J1 Connector Pinout 126
6 Backplane Pin Descriptions 126
7 J2 Connector Pinout 127
8 J3 Connector Pinout 128
9 J4 Connector Pinout 128
10 J5 Connector Pinout 129

Revision History

Date	Revision	Description
May 2003	001	Initial Release of this document.

Introduction 1

The Intel® NetStructure™ ZT 8102 Gigabit Ethernet Switch is a high performance managed switch in a 6U CompactPCI form factor. For fast connection speeds and flexibility, it has fourteen 10/100/1000 Mbps Ethernet ports routed to the backplane and two gigabit Ethernet ports routed to the front panel as SFP (Small Form factor Pluggable) fiber uplink ports. The in-chassis switch minimizes external wiring, thus improving density and reliability.

The switch can be managed using Telnet, a Web browser, SNMP, or through IPMI via the Intel® NetStructure™ ZT 7102 Chassis Management Module. The ZT 8102 switches and routes at full wire speed with its non-blocking architecture, and it has sophisticated multicast protocols to limit unnecessary traffic. It provides an in-chassis switch fabric that you can configure to operate in a redundant configuration.

1.1 PICMG* 2.16 Compliant

The CompactPCI Packet Switching Backplane (CPSB) standard developed by the PCI Industrial Computer Manufacturer Group's subcommittee 2.16 (PICMG 2.16) defines an embedded Ethernet environment for telco chassis. This environment includes two switch fabric slots that create a dual star Ethernet network to the 14 node slots. Placing the ZT 8102 in a fabric slot provides embedded Ethernet services to each node card across the Packet Switching Backplane of the chassis. A standard configuration is one ZT 8102 placed in each of the two fabric slots in a chassis for creation of a redundant, high availability system.

1.2 ZNYX OpenArchitect* Switch Management

The ZNYX OpenArchitect* software component – open source Linux*, IP protocol stack, control applications and the OA Engine provides extensive managed IP routing protocols and other open standards for switch management. Examples include network services; Virtual Redundant Router Protocol; Routing Information Protocol; Open Shortest Path First; IP Multicast, Quality of Service and Class of Service; access control lists; and management via Simple Network Management Protocol MIBs, Common Open Policy Services and web.

1.3 Extensible Customization of Routing Policies

The OpenArchitect* software environment enables rapid porting of other UNIX/Linux-based protocols, including open source RFCs. It also enables the development of application-specific protocol configuration scripts. This extensibility and access to the open source networking software provides OEMs and integrators low cost, rapid development environment for creating new value-added services.

1.4 Powerful CarrierClass* Features

The ZT 8102 has High Availability hardware features for advanced telecommunication applications. The switch implements the PICMG 2.1 Full Hot Swap support. This feature provides field replaceable capabilities where a switch can fail and be replaced without impacting the operational performance of a chassis.

Also supported is the Intelligent Platform Management Interface (IPMI) standard for message-based interfaces that monitor the physical health characteristics of the ZT 8102. The switch provides operational status information to an IPMI management application. End customers benefit with advanced notice of potential problems.

The ZT 8102 also implements the Media Dependent Interface called Auto MDI-X on the two front panel 10/100 ports. Auto MDI-X allows connections to any device, switches, hubs, or systems using a regular straight-through or crossover CAT 5 cable. The RJ45 port will auto detect and switch MDI/MDI-X modes. This IEEE standard makes cabling – especially between switches – faster and less error prone.

1.5 OpenArchitect* Software Structure

OpenArchitect is based on an embedded Linux operating system and includes a number of libraries and modules. The key element is the Linux routing table, which is an object in a network-enabled Linux* implementation.

The purpose of the routing table is to tell the packet forwarding software where to forward the data packets. In “standard” Linux, the packet-forwarding algorithm is operated in software. Normally, the routing tables are maintained by operator configuration and the various routing protocols that run in the application environment of Linux.

OpenArchitect uses an innovative new approach for forwarding packets. It provides embedded software daemons that ensure that the Linux routing tables are replicated (or shadowed) in a forwarding table. The OpenArchitect silicon directly monitors the forwarding table and the switch fabric completes most of the “real-time” work in switching network packets. The switch fabric consults its own forwarding table for each incoming packet; and either filters or forwards the packet to any egress port, the embedded CPU, or to any combination.

The OpenArchitect environment includes additional features. For example, installing the OpenArchitect switch gives you immediate implementation of Linux routing protocols. Also, you have complete support of routing table updates and a standardized method for configuration.

1.6 Features Overview

1.6.1 Hardware

- Hot-swappable
- 14 10/100/1000 Mbps backplane ports
- 2 GbE fiber front-panel SFP ports
- 1 10/100 Mbps front-panel-out-of-band management port (RJ-45)

- 1 100 Mbps backplane out-of-band inter-switch fabric link (LPf)
- 64 MB SDRAM
- 32 MB flash
- Ethernet Port Link, Speed, and Activity LEDs for each port
- Power and Hot Swap, Health, status LEDs
- RS-232 serial port on the front panel
- IPMI Bus (IPMB) connection to the Backplane
- Reset switch
- Support for Rear Transition Module

1.6.2 Layer 2 Switching Functions

- 32 Gbps non-blocking switching fabric
- Auto-negotiation for 10/100/1000 Mbps speed
- Jumbo Frames Support (9 k)
- 802.1D Spanning Tree
- 802.1Q, port-based VLANs
- Store and forward switching mode
- IEEE 802.3x flow control (full-duplex)
- Back pressure flow control (half-duplex)
- Link Aggregation
- IGMP Snooping
- Port Mirroring
- GMRP (Group Multicast Registration Protocol)

1.6.3 Layer 3 Switching Functions

- DHCP/BOOTP relay
- Wire speed IP forwarding rate per system
- Hardware-based Layer 3 IP switching
- 2 KByte for IP address caching
- RIP (Routing Information Protocol) v1 and v2
- OSPF (Open Shortest Path First)
- VRRP (Virtual Router Redundancy Protocol) RFC 2338 (ZT 8102HA only)
- IP v4
- IGMP (Internet Group Management Protocol) v2

1.6.4 Management Functions

- IPMI support
- RS-232 Console Management Port
- Telnet console
- SNMP v1, v2, and v3
- Web-based management

1.6.5 Warranty

2 years

1.7 Specifications

1.7.1 Electrical

- Less than 64 W power consumption
- 3.3 V, 5.0 V, 12 V supplies.

Table 1. Power Requirements

Power Requirements	Typical
Supply Voltage, Vcc	+5 VDC +5%, -3%
Supply Current, Vcc=5.0 VDC	5.2 A
Supply Voltage, V3.3 V	+3.3 VDC +5%, -3%
Supply Current, V3.3 V=3.3 VDC	7.2 A
Supply Voltage, V12.0 V	+12 VDC ±10%
Supply Current, V12 V=12.0 V	360 mA

1.7.2 Mechanical

- Measures 9.2” x 6.3” (233.35 mm x 160 mm)
- Width: 0.8” (1 slot - 4HP)
- Connector: IEC-1076-4-101 (J1-J5)

1.7.3 Environmental

- Operating Temperature (requires 300 LFM airflow): 0° to 40° C
- Storage Temperature: -25° to +55° C
- Non-Condensing Relative Humidity: less than 95 percent at 40° C
- Designed for NEBS Level 3

- MTBF > 80,000 hours

1.7.4 Safety and Emission (EMI) Certifications

- UL/CUL
- IEC60950/EN60950
- FCC Class A
- CE Mark
- ICES
- VCCI
- C-Tick

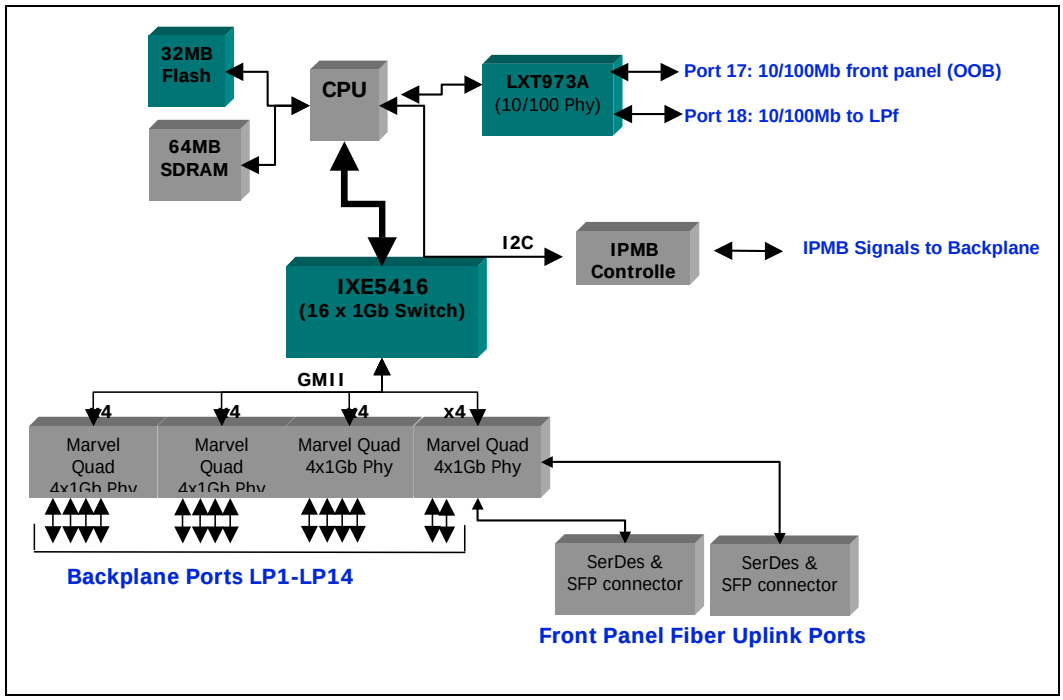
1.7.5 Standards

- PICMG* 2.0 R3.0, PICMG 2.16, PICMG 2.1 R2.0, PICMG 2.9, R1.0, PICMG 2.10, R1.0
- IEEE* 802.1D Spanning Tree
- IEEE 802.1Q Tagged VLAN
- MIB, MIB-II, Bridge MIB, RIP MIB, CIDR MIB, 802.1p MIB, RFC 1157, RFC 1516
- Repeater MIB, RFC 1643 Ethernet MIB, RFC 2737 Entity MIB, RFC 2239 IEEE 802.3 MAU MIB
- IEEE 802.3 10BASE-T, IEEE 802.3u 100BASE-TX, and IEEE 802.3ab 1000BASE-T
- IEEE 802.1P Priority Tagging, 802.3ac VLAN TAG, 802.3ad Link Aggregation, 802.3x Flow Control RFC 768, 783, 791/950, 792, 826, 854, 855, 856, 857,1058, 1519, 1542, 1723, 2068, 2113, 2328, 2131, 2236, 2338

1.8 ZT 8102 Functional Blocks

Below is a functional block diagram of the ZT 8102.

Figure 1. Intel® NetStructure™ ZT 8102 Functional Block Diagram



1.9 PICMG 2.16 Board Keying

The ZT8102 Ethernet Switch is keyed for standard fabric keying as per the PICMG 2.16 specification. The key is installed in J4 and is Blue/Lilac in color which represents standard fabric keying. Fabric boards must be keyed as standard fabric boards when they support 19 or less link ports.

1.10 IPMI

The ZT 8102 has the capability of monitoring critical on board parameters using the flexible, industry standard, Intelligent Platform Management Interface (IPMI). The ZT 8102 is equipped with an on-board Baseboard Management Controller (BMC) chip and IPMI v1.5 compatible firmware already installed on the board.

Some of the functions available on this board through the IPMI interface include:

- Monitoring board temperatures in two locations
- Monitoring of on board voltages (+12 V, +5 V, +3.3 V)
- Remote reset and shutdown of the board

- Board power up/power down by BD_SEL# signal on the CompactPCI backplane

In order to take advantage of the features provided by the firmware, IPMI aware applications must be developed. Information on IPMI v1.5 can be found at:

http://www.intel.com/design/servers/ipmi/spec.htm

For more information about how to program software to interact with the IPMI firmware, refer to the *Intelligent Platform Management Interface v1.5 Specification* and the *Intelligent Platform Management Interface Implementer's Guide*.

Installation and Initial Setup 2

This chapter provides installation and initial setup information for the switch.

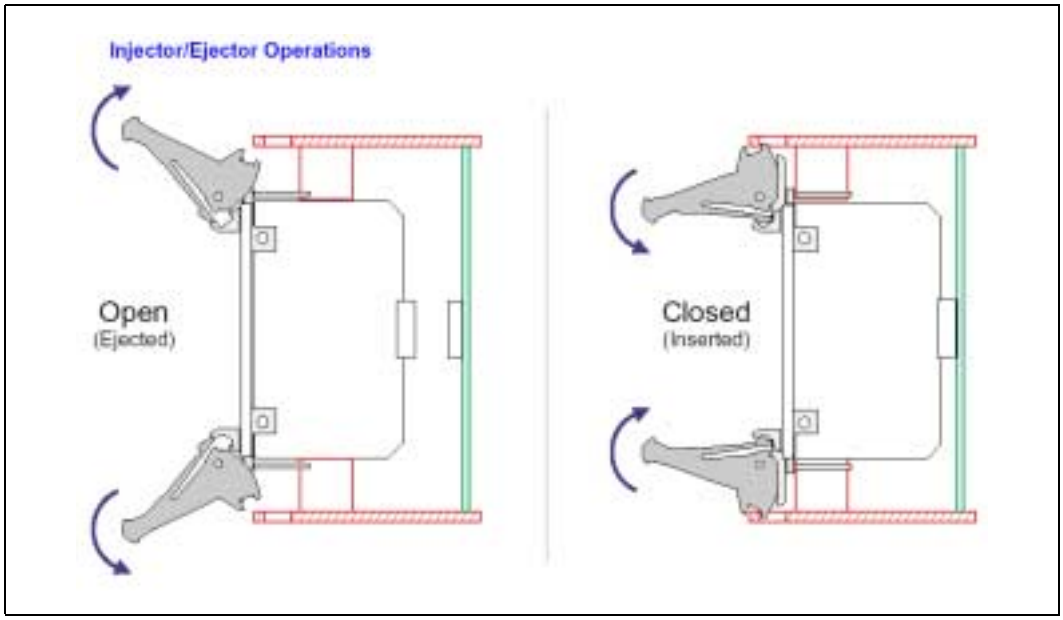
2.1 Installing the Board

These instructions explain the mechanical aspects of installing a ZT 8102 board. The board should be installed in a PICMG* 2.16-compliant fabric slot.

Caution: Many components in the system contain sensitive electronic components. Service personnel should follow proper grounding procedures when installing or servicing this equipment.

1. System power does not need to be off to insert a ZT 8102 board.
2. Discharge any static electricity from your body by touching the metal chassis, or by using an anti-static wrist strap. If you do not have a ground strap, maintain physical contact with the case to maintain the same electrical potential with the system.
3. Prepare the board by opening the injector/ejector mechanisms.

Figure 2. Injector/Ejector Operations



4. Carefully align the edges of the board with the left and right card guides in the appropriate slot. It may be helpful to look into the enclosure to verify correct alignment of the rails in the guides.
5. Taking care to keep the board aligned in the guides, slide the board in until the injector/ejector mechanisms engage the retention bars.

6. Simultaneously push in the board and rotate the injector/ejector mechanisms to their closed positions (rotate inward) to seat the backplane connectors. When the board is in place, it will boot if the system power is on.
7. Make the desired connections at the faceplate and configure the board.

2.2 Power on

After the power switch is turned on, the LED indicators should respond as follows:

- All LED indicators will momentarily blink, which represents a reset of the system.
- After approximately 15 seconds, the Health LED indicator will turn green to indicate the switch is in a ready state.
- The hot-swap LED indicator will turn off.
- The port LED link indicators will be off if there is no Ethernet connection and on if there is an Ethernet connection.

2.3 Uninstalling the Board

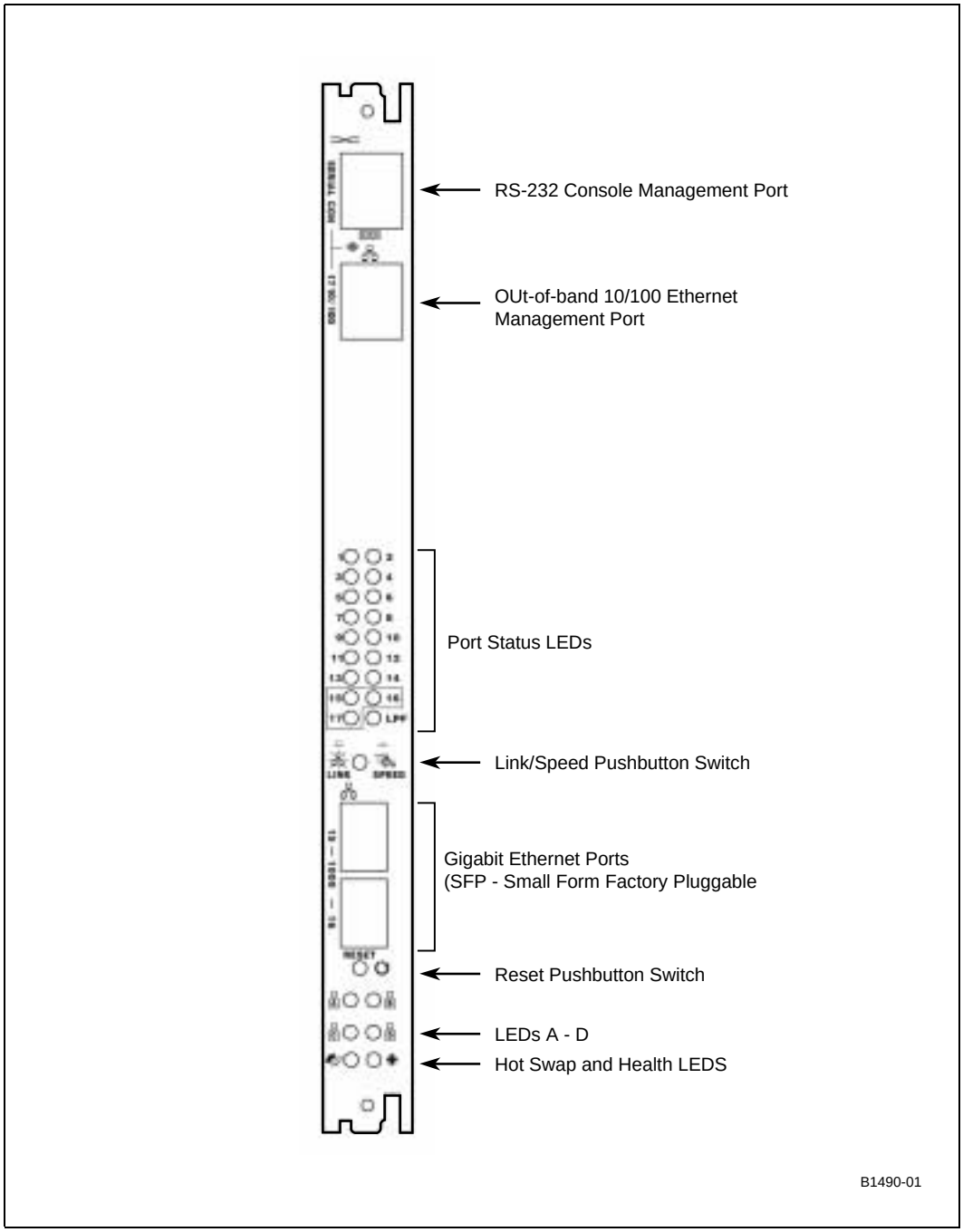
These instructions explain the mechanical aspects of removing a Intel® NetStructure™ ZT 8102 Gigabit Ethernet Switch board from a system.

1. You do not need to turn off the system power to remove a ZT 8102 board.
2. Discharge any static electricity from your body by touching the metal chassis, or by using an anti-static wrist strap. If you do not have a ground strap, maintain physical contact with the case to maintain the same electrical potential with the system.
3. Disconnect connections at the faceplate (Ethernet and serial ports).
4. The board should be in a “safe” state to be removed or data may be lost. Signal the system that a board is about to be removed by partially unlatching the ejectors on the board to be removed. Do not fully open the ejectors, as this levers the board out of the enclosure and prematurely breaks its backplane connection.
5. Wait for the blue hot swap LED on the board’s faceplate to illuminate. This indicates that board processes have finished and the board is safe to extract. If the hot swap LED fails to light after 30 seconds, re-latch the ejectors and unlatch them again. In this case, the board is safe to extract (though the hot swap LED may not light).
6. Once the hot swap LED illuminates, open the injector/ejector mechanisms fully, rotating the handles outward until the board disengages from the backplane (see Figure 2 on page 16).
7. Slide the board evenly out of the enclosure.
8. Install a replacement board or cover the empty slot with a filler panel to maintain the enclosure’s shielding and cooling performance.

2.4 Identifying External Components

This section describes the front panel and the LED indicators of the ZT 8102 switch. The front panel consists of LED indicators, an RS-232 Console Management Port, a link/speed toggle button, a reset button, two Gigabit Ethernet SFP ports, and an out-of-band management 10/100 Ethernet port.

Figure 3. Intel® NetStructure™ ZT 8102 Front Panel



2.5 Connecting the Cables

Your switch setup may require some or all of the following three types of cables: 1000BaseSX, RS-232 or Category 5E.

2.5.1 1000BaseSX Cabling

The ZT 8102 uses a Small Form Pluggable (SFP) multi-mode fiber optic module with an LC connector for the front panel Gigabit Ethernet ports (port 15 and 16). The modules can drive up to 500 meters using 50/125 micron cable and up to 220 meters with 62.5/125 micron cable. The following SFPs have been tested with the ZT 8102:

- Finisar* FTRJ-8519-3-2.5
- Agilent* HFBR 5710L
- Agilent HFBR 5710LP

2.5.2 RS-232 Console Management Port Cabling

The switch management console can be accessed via the RS-232 Console Management Port located on the front panel of the ZT 8102. The switch Console management Port can be used to configure, manage, or recover from a system failure. Use a standard null-modem console cable to access the switch console or use the pin out specified in Appendix A to build your own.

2.5.3 Out Of Band (OOB) 10/100 Ethernet Management Port

The ZT 8102 has a 10/100 Mbps Ethernet out-of-band management port (port 17) that can used to manage the switch. The out-of-band management port is isolated from the rest of the switch fabric and is only used for a management network or by a PC that is directly connected to the port. The 10/100 Mbps Ethernet out-of-band management port cannot be used to connect external devices to the rest of the switching fabric (ports 1-16). By default, port 17 will auto-negotiate port settings. Use Category 5e cabling for all 10/100 connections to out-of-band management port 17. Port 17 on the ZT 8102 is auto-MDI sensing and will automatically determine whether or not an MDI (straight-through) or MDI-X (crossover) cable is attached.

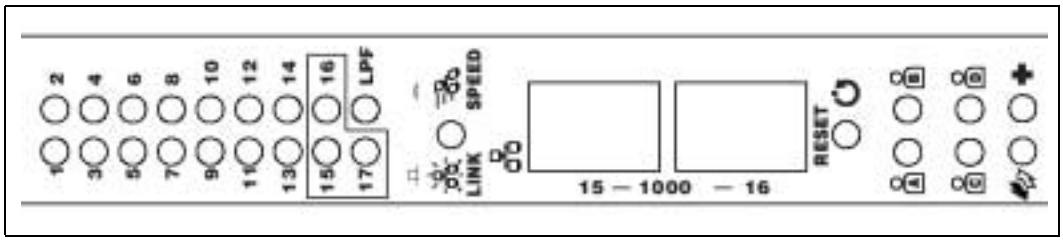
The 10/100 Mbps Ethernet out-of-band management port is designated as *eth0* in the switch software and can be configured using the *ifconfig* command. For example, the command to set the IP address to 192.0.0.1 on Port 17 and bring the port up is:

```
ifconfig eth0 192.0.0.1 netmask 255.255.255.0 up
```

2.6 Status LEDs

The LED lights on the ZT 8102 switch provide diagnostic and activity information during power up and normal use of the switch.

Figure 4. LEDs



2.7 ZT 8102 LED Reference

The following table describes the front panel LEDs and push-button switches.

Table 2. Front Panel LEDs

LED	Function	
HEALTH	Off	Not Powered and 'No Attention' status
	Amber	Powered or not powered and 'Attention' status. 'Attention' status is indicated by any fault condition of the CLOCK, OK, or INT FLT LEDs
	Green	Powered and 'Not Attention' status
HOT SWAP	Blue	After the extraction levers are released (pulled apart), ON indicates the switch unit is ready to remove. During “hot” insertion (system is active), Hot Swap LED is ON and turns OFF when the switch is correctly seated. OFF during power
INT FLT(LED A)	Amber	ON indicates a failure of the internal tests. OFF during power up
EXT FLT(LED B)	Amber	ON indicates a failure external to the adapter (e.g., inability to establish a link on a configured port or another connectivity problem external to the adapter).
OK(LED C)	Green	ON indicates the initialization has completed successfully, and at least one interface is configured up. OFF during power up.
CLK(LED D)	Green	Blinking when driver has loaded. OFF during power up.
LINK/SPEED Switch		Change the Port LEDs display function from "Link /Activity" mode (default) to "Link /Speed" mode.
RESETSwitch		Pressing RESET for at least three seconds restarts the switch.
Port LEDs1 - 14 (10/100/1000) 15-16 (SPFs) 17(10/100 OOB) LPF Fabric (10/100 OOB)	Off	Link/Activity Mode (default) No link Link/Speed Mode (LINK/SPEED Switch depressed) OFF indicates 10 Mbps connectionGreen
	Green	Link/Activity Mode (default) ON indicates link BLINKING indicates traffic Link/Speed Mode (LINK/SPEED Switch depressed) ON indicates 100 Mbps connection
	Amber	Link/Activity Mode (default) Port is not forwarding packets; disabled by management or blocked by Spanning Tree Protocol Link/Speed Mode (LINK/SPEED Switch depressed) ON indicates 1000 Mbps connection

NOTE: The LPF (Link Port Fabric) port is connected directly to the switch fabric interconnect. The LPF port is used to transfer frames between switches when two switches exist in the same chassis.

2.8 Getting Started with Management

The switch contains the following components:

- CPU – central processor unit
- Memory for data storage
- Flash memory for configuration data, operational programs, and SNMP agent firmware.

These components allow you to manage and monitor the switch from either the RS-232 Console Management Port, the out-of-band network port (port 17), or from the network itself (ports 1-16). You can configure and manage the switch from these locations:

- A terminal or a workstation running terminal emulation software and connected to the switch via the RS-232 Console Management Port.
- A workstation connected to out-of-band network port (port 17) and running Telnet.
- A workstation connected to out-of-band network port (port 17) and running a Web browser.
- A workstation connected to the network (ports 1-16) and running Telnet.
- A workstation connected to the network (ports 1-16) and running a Web browser.

To access the switch via Telnet or a Web browser you must assign the switch an IP address for your network or you must modify the IP address of your workstation to be on the same default IP subnet that the switch uses. To modify the switch's IP address you must access the switch using the RS-232 Console Management Port.

Note: If you are managing the switch via telnet or web browser and you shutdown the switch port that is being used for the connection, the management session will be lost.

This section explains how to:

- Set up access to the Local Console
- Configure the switch's IP address

Once you complete these tasks, you can access the switch using any of the five available methods (Section 1.6.4, "Management Functions" on page 11.)

2.9 Accessing the Local Console

The Local Console is a terminal or a workstation running a terminal emulation program that is connected directly to the switch via the RS-232 Console Management Port on the front of the switch. The Local Console can be used to set up and manage the switch even when the network is down.

2.9.1 Connecting to the RS-232 Management Console Port

1. Connect a null-modem serial cable between the RS-232 Management Console Port on the front panel of the switch and the COM port on your PC.
2. Use a terminal (such as a VT-100) or a computer running a terminal emulation program such as Windows* HyperTerminal to access the switch console.
3. Configure your terminal setting to the following:
   - Baud rate: 9600
   - Data width: 8 bits
   - Parity: None
   - Stop bits: 1
   - Flow Control: None

If you are having problems making this connection on a computer, make sure the emulation is set to VT-100. If you still don't see anything, press CTRL+R to refresh the screen.

Note: The RS-232 Console Management Port on the front panel of the switch uses a Cisco* cable kit (Cisco Order Number: ACS-DSBUASYN). This kit includes a DB-25 terminal adapter, a DB-9 terminal adapter, and RJ-45 rollover (null modem) cable. Optionally, to build this cable, refer to Appendix A, “RS-232 Serial Cable.”

2.9.2 To log in to the switch the first time

After you are connected to the switch, enter the login name “root” No password is required.

login: root

2.9.3 Configuration Management Overview

Every command that is issued using the command line is stored in temporary memory (i.e., configuration will be lost once power is turned off). In order to save the configuration, use the *zsync* command (see “zsync” on page 118).

If a particular configuration is needed every time the switch boots, the best method is to set up a script file that will execute every time the switch is booted.

2.9.4 Setting the IP Address for In-Band Management

The switch needs a valid IP address for your network to access the switch via Telnet or the Web. By default, the switch has a factory default IP address of 10.90.90.90 with a subnet mask of 255.255.255.0 assigned to the interface *zhp0* which is the default VLAN for ports 1-16. Any device that is connected to ports 1-16 on the switch can access the switch using this default IP address. You can change the default IP address of *zhp0* using the using the *ifconfig* command. For example, to change the IP address of *zhp0* to 192.168.1.1 use the following:

```
ifconfig zhp0 down
ifconfig zhp0 192.168.1.1 netmask 255.255.255.0 up
```

2.9.5 Setting the IP Address for Out-of-Band Management

If you would like to access the switch via Telnet or Web on the 10/100 Mbps Ethernet out-of-band management port (port 17), then you must assign an IP address to that port. At factory default, you cannot access the switch at the default IP address of 10.90.90.90 via port 17. The 10/100 Mbps Ethernet out-of-band management port is designated as *eth0* in the switch software and can be configured using the *ifconfig* command. For example, to set the IP address of *eth0* (port 17) to 192.0.0.1 use the following:

```
ifconfig eth0 192.0.0.1 netmask 255.255.255.0 up
```

Note: The 10/100 Mbps Ethernet LPF port is designated as *eth1* in the switch software.

To make the IP address assignments permanent, the IP configuration commands must be part of the startup configuration script that runs when the switch is booted. The default configuration script for the switch is /etc/rcZ.d/S50Layer 2. See Chapter 3 for more details on configuration scripts.

Switch Configuration 3

After the ZT 8102 switch has been installed and powered up for the first time, you can use the information in this chapter to help you configure the switch for your networking environment.

3.1 Switch Configuration Procedure

Layer 2 and Layer 3 switch configurations can be accomplished with a few simple commands. Once you've configured your switch, the commands should be placed into a startup configuration script. Like most Linux systems, the OpenArchitect* switch boot process runs initialization commands and scripts in */etc/init.d/*. In particular, OpenArchitect runs */etc/init.d/rcS* which in turn executes all scripts located in */etc/rcZ.d* starting with an uppercase "S" in alphabetical order. Any configuration scripts you create should be named in the standard Linux/Unix manner, starting with an uppercase "S" and numbered in the sequence you would like them executed. The final step once the switch has been properly configured is to use the *zsync* command to save all files into flash for reloading.

3.2 Example Configuration Scripts

Intel supplies example scripts that can be used as templates. Use one of the scripts located in the switch */etc/rcZ.d/examples* directory to help you configure the switch. The default configuration for the switch is located in the script *file /etc/rcZ.d/S50Layer 2.*

The following scripts are included. Each is examined in more detail later in the appropriate section describing common Layer 2 and Layer 3 configurations.

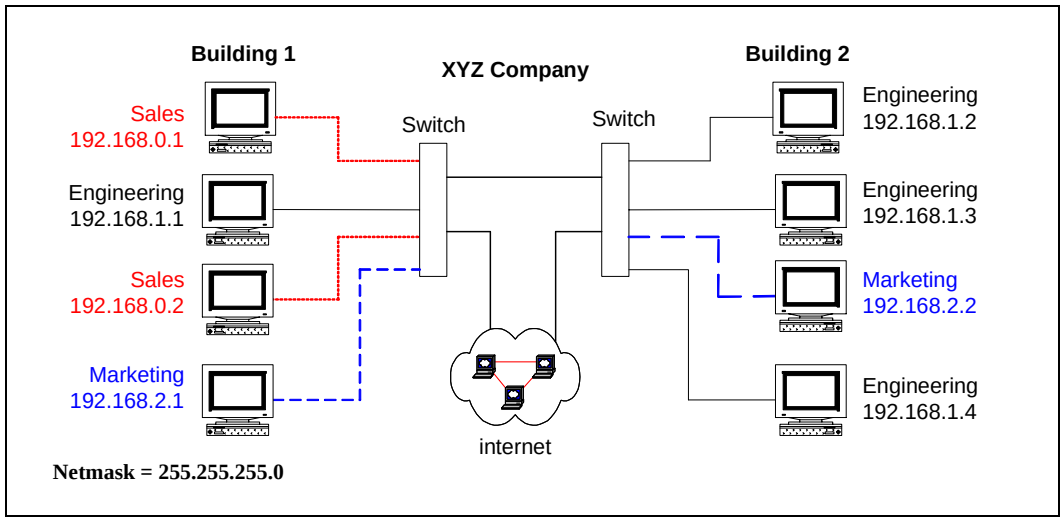
S50layer2	Sets up a basic Layer 2 switch. All 14 10/100/1000 ports and the two SPF ports (port 15 and 16) are set up on one IP network (VLAN).
S50layer2sp	Sets up a basic Layer 2 switch. All 14 10/100/1000 ports and the two SPF ports (port 15 and 16) are set up on one IP network (VLAN), and turns on bridge support for Spanning Tree.
S50layer3	Sets up a basic Layer 3 switch. Each of the 14 10/100/1000 ports and the two SPF ports (port 15 and 16) are set up on individual IP networks (VLANs). Layer 3 switching is enabled.
S50multivlan	Sets up multiple untagged VLANs. The first VLAN includes the first seven 10/100/1000 ports, the next contains the last seven 10/100/1000 ports, and one for each SPF port. Layer 3 switching is enabled.
S55gatedRip1	Used with a Layer 3 switch and calls the GateD daemon to enable RIP 1 routing protocol.
S55gatedRip2	Used with a Layer 3 switch and calls the GateD daemon to enable RIP 2 routing protocol.
S55gatedOspf	Used with a Layer 3 switch and calls the GateD daemon to enable OSPF routing protocol.

3.3 Overview of VLAN Interfaces

When you initially boot up the switch, one virtual host port is automatically created by OpenArchitect to enable interaction between the software and hardware. This initial host port, called *zhp0*, is a network interface that provides communication between all 16 ports. Therefore, linking to any port on the switch enables you to connect with OpenArchitect.

A *zhp* device is associated with one Virtual Local Area Network (VLAN). A virtual local area network (VLAN) is a logical mapping of workstations and network devices on some basis other than geographic location (e.g., by department, type of user, or primary application). The primary purpose of a VLAN is to isolate traffic and enable communication to flow more efficiently within groups of mutual interest. VLANs reduce the time it takes to implement workstation and network moves, adds and changes. The switch is used to bridge from one VLAN to another. Figure 5 is an example of a hypothetical company that occupies two buildings to illustrate multiple VLANs in a possible VLAN network structure.

Figure 5. Multiple VLANs Example



In the diagram above, three VLANs are utilized and divided as follows:

VLAN	Group	IPNetwork	Interface
100	Sales	192.168.0.0	zhp0
200	Engineering	192.168.1.0	zhp1
300	Marketing	192.168.2.0	zhp2

In the example, the sales employees are all located in Building 1. They are combined into VLAN 100 (zhp0), on the 192.168.0.0 IP network, because they interact among customers and each other on a daily basis. The engineering and marketing groups have employees working in both Buildings 1 and 2. Each of these groups are also separated into different VLANs (200 and 300, respectively) and in different subnets.

Outside communication into the company can also be placed on a separate VLAN to maintain control of outside traffic for security reasons.

3.3.1 Tagging and Untagging VLANs

The switch is capable of switching 802.1Q tagged and untagged data packets. 802.1Q tagged packets are labeled in the data header for switching to a specific VLAN by a switch port. Untagged packets enable all untagged packets for a specific VLAN to go through a port.

In the example detailed in Figure 5 above, all data packets in VLAN 200 and 300 would be tagged so that correspondence between the groups in these VLANs would be routed to the correct workstations.

3.3.2 Switch Port Interfaces

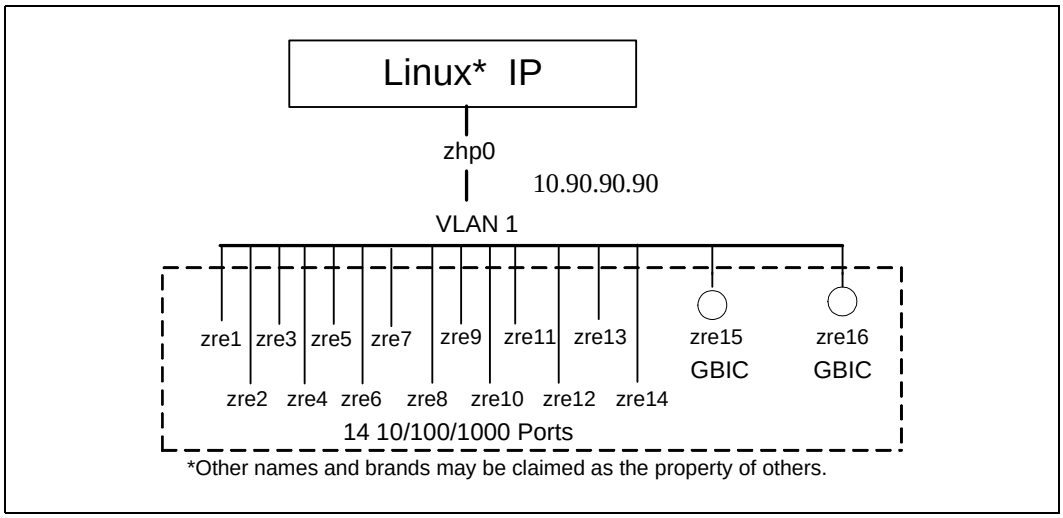
For each switch port, a separate interface is created with its own MAC address called a *zre*. After the initial power up, 16 *zre* interfaces are created, one for each in-band port. You cannot directly access or modify the *zre* interfaces.

During the initial power up of the switch, the default configuration creates a Layer 2 switch. The Layer 2 configuration places all of the *zre* interfaces in the same *zhp* interface. The number after *zre* represents the corresponding switch port number (i.e., *zre1* represents port 1 on the switch).

3.4 Layer 2 Switch Configuration

The steps to build a Layer 2 switch involve creating a group of switch ports in a VLAN (or Layer 2 switching domain) and bringing that interface up. *zconfig* creates the VLAN group of switch ports as well as a network interface. You can use *ifconfig* on the network interface (*zhp*) to bring up the VLAN group.

Figure 6. Layer 2 Switch



During the initial power up, a startup script called */etc/rcZ.d/S50Layer 2* is executed at boot time creating a single untagged VLAN (IP interface labeled as *zhp0*) which includes all ethernet and gigabit ports (ports 1-16) as one Layer 2 switch *(see Figure 3-2)*. The interface to the host is then assigned the IP address of 10.90.90.90 to allow access to the switch. The *S50Layer 2* script does the following:

- Uses *zconfig* to create and configure a single, untagged VLAN that contains all 16 switch ports.

```
/usr/sbin/zconfig zhp0: vlan1=zre1..16
/usr/sbin/zconfig zre1..16=untag1
```

- Uses *ifconfig* to assign the IP address 10.90.90.90 to the interface.

```
/usr/sbin/ifconfig zhp0 10.90.90.90 netmask 255.255.255.0 up
```

If you wished to create another VLAN that only contained the two mini GBIC ports, first use *zconfig* from the command to build the VLAN and create a network interface for the host.

```
zconfig zhp1: vlan2=zre15,zre16
```

Then, bring up the interface with *ifconfig*:

```
ifconfig zhp1 192.168.1.1 up
```

Note that ports *zre15* and *zre16* are members of both vlan1 and vlan2, and that they are tagged for vlan2. A port cannot be untagged for more than one VLAN. You can view the configured VLANs with *zconfig*.

```
zconfig -a
```

3.4.1 Using the Provided S50Layer 2 Script

The S50Layer 2 script can be used and example, or edited to customize your Layer 2 setup. For example, to reconfigure the IP address on your Layer 2 switch**,**

- Open the *S50Layer 2* file in the Linux *vi* editor.
- Change the IP address value listed under the Linux *ifconfig* command line.
- Save your changes by running *zsync*.

```
zsync
```

- Reboot the switch.

3.5 Spanning Tree

The Spanning Tree Protocol (STP) configures a simply connected active topology from the arbitrarily connected components of a Bridged Local Area Network. STP participants use a simple dialog carried in packets called Bridge Protocol Data Units (BPDUs) for finding the least expensive path between two networks and for eliminating loops from the topology. If nodes attached to ports fail or are added or deleted, the topology dynamically changes to accommodate the new configuration. If your network topology is such that there is no real redundancy or chance for loops, you do not need to turn on Spanning Tree.

zl2d is used to create Linux bridges consisting of the name of the previously created *zhp* device or devices preceded with a “b’ (e.g. if you are creating a Bridge device from *zhp0*, the resulting device would be *bzhp0*). *zl2d* then starts a background task that monitors the port information of the Linux bridge at a specified interval and updates the Spanning Tree state fields in the hardware when necessary.

brctl is used for configuring certain STP parameters. For an explanation of these parameters, see the IEEE 802.1d specification. The following demonstrates a simple example of setting up a Layer 2 switch and starting STP.

3.5.1 To Enable Spanning Tree:

Create a VLAN containing the ports that will be a part of the Linux bridge running Spanning Tree. This example will use ports 0-4 (untagged):

```
zconfig zhp0: vlan1=zre1..4
zconfig zre1..4=untag1
```

Create a bridge device from the zhp device:

```
zl2d start zhp0
```

A Bridge device named *bzhp0* should now exist consisting of ports *zre1* through *zre4* with Spanning Tree enabled. To view the bridge device, use the *brctl* command:

```
brctl show
brctl showbr bzhp0
```

3.5.2 Port Path Cost

Each port has an associated cost that contributes to the total cost of the path to the Root Bridge when the port is the root port. The lower the cost, the better the path. The ZT 8102 uses the following IEEE 802.1D recommendations based on the connection speed of your port

Table 3. IEEE 802.1D Recommendations

Link Speed	Recommended Value	Recommended Range
10 Mb/s	100	50-600
100 Mb/s	19	10-60
1 Gb/s	4	3-10

To change the port path, use the *brctl setpathcost* option. For example, to set the port cost of port 1 to a cost of 4, use the following command:

```
brctl setpathcost bzhp0 zre1 4
```

3.5.3 Fast Forward Port

Fast Forward Port alters the STP algorithm to expedite Spanning Tree recovery by bypassing the Listening and Learning states of STP, and transitioning directly to Forwarding. This can reduce STP recovery time from potentially 50 seconds down to less than 1 second. Fast Forward Port is implemented on a per port basis. Fast Forward port should ONLY be enabled on ports connected to end-stations. Do not enable Fast Forward port on ports that can result in the creation of loops, such as other network devices like switches that have multiple connections.
To enable Fast Forward Port, setup a bridge device with Spanning Tree enabled, then use *brctl* to enable Fast Forward Port for the specific port on a specific bridge:

```
brctl ffport bzhp0 zre1 1
```

3.5.4 Fast Forward Bridge

Fast Forward Bridge improves on the STP algorithm providing quicker convergence when STP picks a new Root Port. If Fast Forward Bridge is enabled for the bridge, whenever STP picks a blocked port as the new Root Port, that port bypasses the Listening and Learning states and transition immediately to Forwarding. By avoiding the Listening and Learning states and transitioning directly to Forwarding on the new Root port, the time for STP convergence is reduced from potentially 50 seconds down to less than 1 second.

To enable Fast Forward Bridge, setup a bridge device with Spanning Tree enabled, then use *brctl* to enable Fast Forward Bridge for that bridge device,

```
brctl ffbridge bzhp0 1
```

3.5.5 Using the Provided S50Layer 2sp Script

The *S50Layer 2sp* script can be used as an example or edited to customize your Layer 2 setup, including Spanning Tree. For example, to reconfigure the IP address on your Layer 2 switch:

- Copy */etc/rcZ.d/examples/S50Layer 2sp* to */etc/rcZ.d/S50Layer 2sp*.
  cp /etc/rcZ.d/examples/S50Layer 2sp /etc/rcZ.d
- Remove the existing *S50Layer 2* script (a copy exists in the examples directory):
  rm /etc/rcZ.d/S50Layer 2
- Open the *S50Layer 2sp* file in the Linux *vi* editor, and make the changes necessary for your configuration. Save your changes by running *zsync*.
  ```
  zsync
  ```
- Reboot the switch.

3.6 Layer 3 Switch Configuration

The previous section outlines the Layer 2 switch configuration that is automatically configured when you initially bring up the switch. In order to communicate between Layer 2 interfaces or VLANs, you must properly setup routing.

The steps to build a Layer 2 switch involve creating a group of switch ports in a VLAN (or Layer 2 switching domain) and bringing that interface up. *zconfig* creates the VLAN group of switch ports as well as a network interface. Use *ifconfig* on the network interface to bring up the VLAN group with Layer 2 switching. Layer 3 routing information is then used to route between the Layer 2 network devices.

Take a simple example of two VLANs configured on the switch, each with four ports. First teardown any existing configuration:

```
zconfig –t
```

Use *zconfig* to create two new VLANs, each with four ports, and untag them:

```
zconfig zhp0: vlan1=zre1..4
zconfig zre1..4=untag1
zconfig zhp1: vlan2=zre5..8
zconfig zre5..8=untag2
```

Now, use *ifconfig* to assign each *zhp* interface an IP address:

```
ifconfig zhp0 10.0.0.1
ifconfig zhp1 11.0.0.1
```

At this point, the Linux host has enough information to route between the networks of the directly attached interfaces, 10.0.0.0 via *zhp0*, and 11.0.0.0 via *zhp1*.

The next step is to enable the *zl3d* daemon to move that routing information from the host to the ZT 8102 switching tables in silicon. Once enabled, *zl3d* will monitor the Linux routing tables for changes in configuration and update the switch silicon tables. Start *zl3d* to update the switch tables:

```
zl3d zhp0 zhp1
```

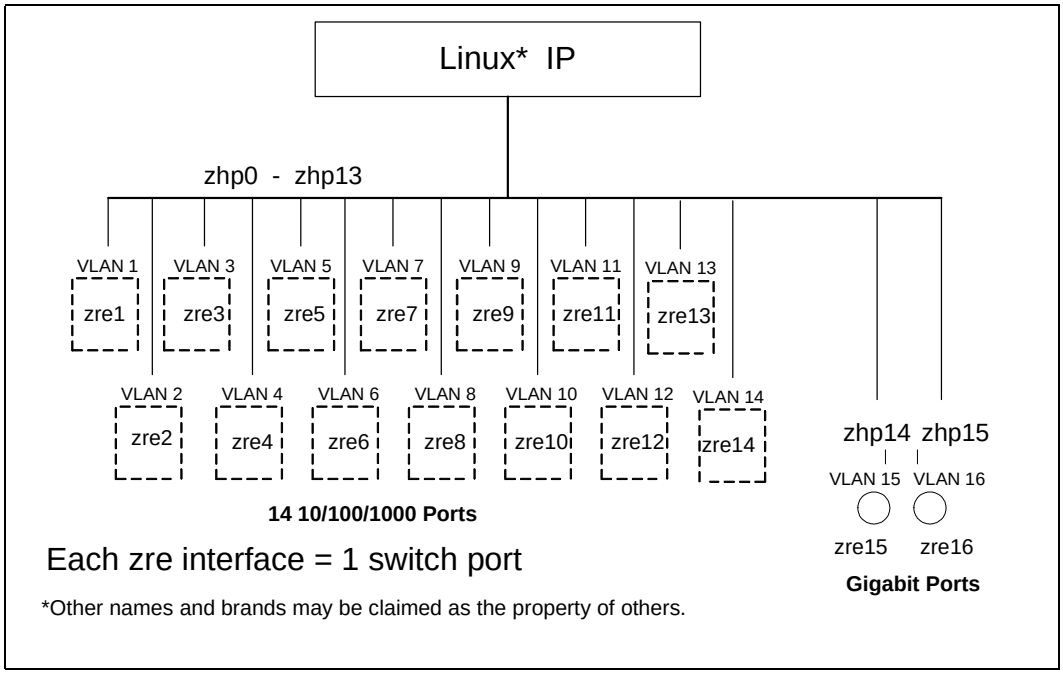
The ZT 8102 switch is now configured as a Layer 3 switch to route packets between VLAN 1 (*zhp0*) and VLAN 2 (*zhp1*).

3.6.1 Using the Provided S50Layer 3 Script

To modify the configuration to a Layer 3 switch, remove the *S50Layer 2* file from the */etc/rcZ.d* directory, and replace it with the example script file, *S50Layer 3*.

In the *S50Layer 3* file, each 10/100/1000 port and gigabit port is assigned its own Virtual Local Area Network (VLAN) interface (port interfaces are labeled as *zhpN*, where *N* is an integer). Each VLAN is associated with an individual *zhp* interface. Remember, *zre* and *zhp* interfaces can begin with a zero value but a VLAN cannot. Each *zre* interface is assigned a separate IP address in the example script.

Figure 7. Layer 3 Switch



The S50Layer 3 script executes the following commands:

- Runs *zconfig* command to create 16 untagged VLANs (one for each switch port).

```
/usr/sbin/zconfig zhp0..15: vlan1..16=zre1+
```

```
/usr/sbin/zconfig zre1..16=untag1+
```

Note: Double periods (..) after vlan1 and untag1 are used to indicate a range of values. The plus (+) sign after zre1 is a wildcard character that means auto-incremented and causes each *zhp* interface to hold only one *zre* (i.e., zhp0 has zre1 on vlan1, zhp1 has zre1 on vlan2).

- Runs the Linux *ifconfig* command for each interface to assign default IP addresses (10.0.0.42-10.0.15.42), sets the netmask and brings up the interfaces.

```
ifconfig zhp0 10.0.00.42 netmask 255.255.255.0 up
ifconfig zhp1 10.0.01.42 netmask 255.255.255.0 up
ifconfig zhp2 10.0.02.42 netmask 255.255.255.0 up
.
.
.
ifconfig zhp13 10.0.13.42 netmask 255.255.255.0 up
ifconfig zhp14 10.0.14.42 netmask 255.255.255.0 up
ifconfig zhp15 10.0.15.42 netmask 255.255.255.0 up
```

- Runs the *zl3d* application. The *zl3d* application monitors the Linux routing tables and updates the switch routing tables for each interface configured above.

```
/usr/sbin/zl3d zhp0..15
```

zl3d initially creates and adds each zhp interface (VLAN) to the switch routing tables. The zhp0..zhp15 is shorthand for the list of interfaces (zhp0, zhp1, Ö, zhp15) to monitor with *zl3d*.

3.6.2 To Modify the Layer 3 Script

1. Modify the example script you copied into the /etc/rcZ.d directory. Adjust and assign IP addresses as applicable. In the example below, the IP address is changed for the interface in the *ifconfig* command line of the script.

From:

```
ifconfig zhp0 10.0.0.42 netmask 255.255.255.0 broadcast 10.0.0.255 up
```

To:

```
ifconfig zhp0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

2. Adjust the number of zhp interfaces, that are added to the routing tables, depending on the number of VLANs you are adding for your network. Include any other details, as applicable.
3. Run the *zsync* command to save your changes.

```
zsync
```

4. Reboot the switch.
5. After rebooting, your switch works from your customized Layer 3 configuration.

3.6.3 Layer 3 Switch Using Multiple VLANs

An example script is also provided for setting up multiple VLANs each with multiple ports.

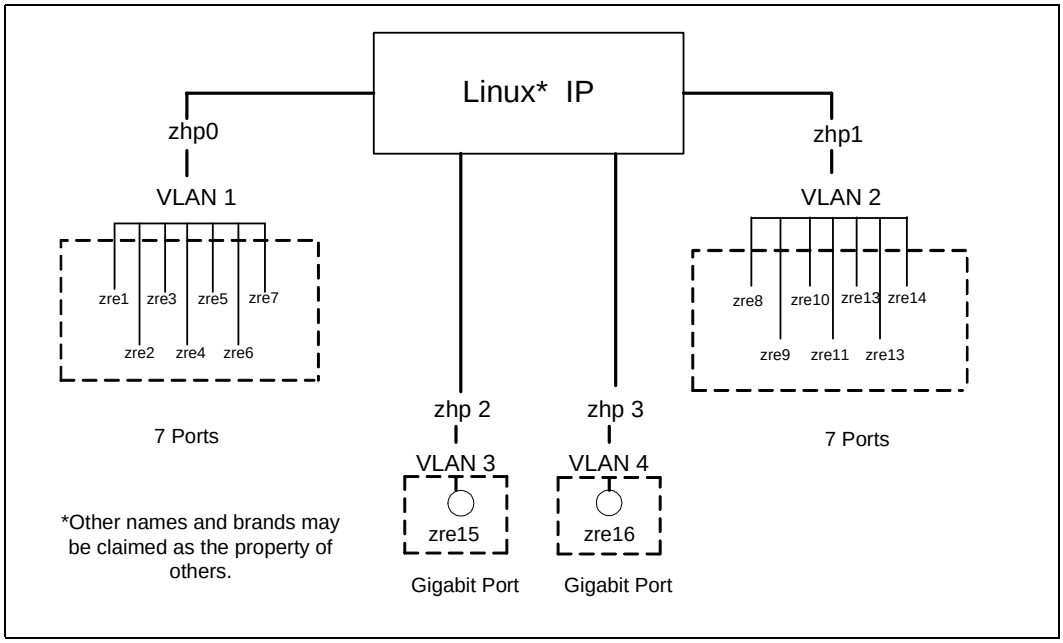
3.6.4 Using the Provided S50multivlan Script

The Layer 3 switch example file, S50multivlan, is included to help you configure multiple VLANs to a Layer 3 switch. A VLAN can include one or more switch ports. In the S50multivlan file, four VLANs are created (see Figure 34):

1. VLAN 1, zhp0: for the first set of seven ports
2. VLAN 2, zhp1: for the second set of seven ports
3. VLAN 3, zhp2: for one gigabit port
4. VLAN 4, zhp3: for the second gigabit port.

Each VLAN interface is labeled zhpN in the file, where N is a value from 0-3. Each interface is untagged and assigned its own IP address.

Figure 8. Multiple VLAN Configuration



The S50multivlan script executes the following commands:

- Runs *zconfig* to create and start four VLANs. Switch ports 1-7 are placed in the first VLAN, ports 8-14 in the second VLAN, port 15 (gigabit) is placed in the third VLAN, and port 16 (gigabit) occupies the fourth VLAN.

```
/usr/sbin/zconfig zhp0: vlan1=zre1..7
/usr/sbin/zconfig zre1..7=untag1
/usr/sbin/zconfig zhp1: vlan2=zre8..14
/usr/sbin/zconfig zre8..14=untag2
/usr/sbin/zconfig zhp2: vlan3=zre15
/usr/sbin/zconfig zre15=untag3
/usr/sbin/zconfig zhp3: vlan4=zre16
/usr/sbin/zconfig zre16=untag4
```

Note: Double periods (..) after vlan1 and untag1 are used to indicate a range of values.

- Runs the Linux *ifconfig* command for each interface to assign the IP address, assign the netmask, the broadcast address, and brings them up.

```
ifconfig zhp0 10.0.0.42 netmask 255.255.255.0 broadcast 10.0.0.255 up
ifconfig zhp1 10.0.1.42 netmask 255.255.255.0 broadcast 10.0.1.255 up
ifconfig zhp2 10.0.2.42 netmask 255.255.255.0 broadcast 10.0.2.255 up
ifconfig zhp3 10.0.3.42 netmask 255.255.255.0 broadcast 10.0.3.255 up
```

- Runs the *zl3d* command. The *zl3d* application monitors the Linux routing tables and updates the switch routing tables for each interface configured above.

```
/usr/sbin/zl3d zhp0 zhp1 zhp2 zhp3

/usr/sbin/zl3d
```

is the location of the *zl3d* application, which initially creates and adds each *zhp* interface (VLAN) to the switch routing tables.

```
zhp0 zhp1 zhp2 zhp3
```

A list of interfaces to monitor with zl3d.

3.6.5 To Modify the Layer 3 Multivlan Script

1. Modify the example script you copied into the */etc/rcZ.d* directory. Adjust and assign the number of IP addresses as applicable. In the example below, the IP address is changed for the interface in the *ifconfig* command line of the script.

```
ifconfig zhp0 10.0.0.42 netmask 255.255.255.0 up
ifconfig zhp0 192.168.1.1 netmask 255.255.255.0 up
```

2. Adjust the number of *zhp* interfaces depending on the number of VLANs you are adding for your network.
3. Include any other details, as applicable.
4. Run the *zsync* command to save your changes.

   zsync
5. Reboot the switch.
6. After rebooting, your switch works from your customized Layer 3 configuration with multiple VLANs per port.

3.7 Layer 3 Routing Protocols with GateD

An advanced networking configuration may require using the GateD software platform for deployment of Routing Information Protocols (RIP 1 or RIP 2) and Open Shortest Path First (OSPF) protocols. Once you've configured your Layer 2 and Layer 3 devices, start *gated*.

3.7.1 Using the Provided S55gatedRip1 Script

To use GateD protocol with the switch, you need to copy two files into the same directory as your Layer 3 configuration file. From the */etc/rcZ.d/*examples folder, copy the example script file and its corresponding GateD configuration file (e.g., *S55gatedRip1* and *gated.conf.rip1*).

The example startup script executes the following commands (*S55gatedRip1* is used as an example):

- Starts GateD with Rip1 (two separate files are required):

```
/usr/sbin/gated -f /etc/rcZ.d/gated.conf.rip1
/usr/sbin/gated
```

finds and starts the GateD application.

```
-f gets the configuration information from the specified file.
/etc/rcZ.d/gated.conf.rip1 is a sample gated configuration file for RIP1
```

The GateD conf file specifies the following configuration commands:

- Implements the passive function so GateD is prevented from rerouting information to a different interface if insufficient information is received.

```
interface 10.0.0.42 passive
interface 10.0.1.42 passive
interface 10.0.2.42 passive
.
.
.
interface 10.0.13.42 passive
interface 10.0.14.42 passive
interface 10.0.15.42 passive
```

- Defines the netmask used in the interface.

```
define 10.0.0.42 netmask 255.255.255.0;
define 10.0.1.42 netmask 255.255.255.0;
define 10.0.2.42 netmask 255.255.255.0;
.
.
.
define 10.0.13.42 netmask 255.255.255.0;
define 10.0.14.42 netmask 255.255.255.0;
define 10.0.15.42 netmask 255.255.255.0;
```

- Enables RIP1 protocol.

```
};
```

- rip1 yes{
  Shuts off sending and receiving packets from all interfaces.

```
interface all noripin noripout
```

- Opens sending and receiving packets for selected interfaces.

```
interface 10.0.0.42 ripin ripout version 1;
interface 10.0.1.42 ripin ripout version 1;
interface 10.0.2.42 ripin ripout version 1;
.
.
.
interface 10.0.13.42 ripin ripout version 1;
interface 10.0.14.42 ripin ripout version 1;
interface 10.0.15.42 ripin ripout version 1;=
```

- Imports routes learned through the RIP protocol.

```
import proto rip {
        all;
```

```
};
```

- Exports all directly connected routes and routes learned from the RIP protocol.

```
export proto rip {
       proto direct }
        all;
};
proto rip {
      all;
};
```

3.7.2 To Modify the GateD Scripts:

1. Copy two GateD files, the OpenArchitect “S” file and its corresponding conf file, into the rcZ.d directory (i.e., *S55gatedRip1* and *gated.conf.rip1*). Notice the files are placed in the same directory as the Layer 3 configuration file.

For RIP1:

```
cp /etc/rcZ.d/examples/S55gatedRip1 /etc/rcZ.d
cp /etc/rcZ.d/examples/gated.conf.rip1 /etc/rcZ.d
```

For RIP2:

```
cp /etc/rcZ.d/examples/S55gatedRip2 /etc/rcZ.d
cp /etc/rcZ.d/examples/gated.conf.rip2 /etc/rcZ.d
```

For OSPF:

```
cp /etc/rcZ.d/examples/S55gatedOspf /etc/rcZ.d
cp /etc/rcZ.d/examples/gated.conf.ospf /etc/rcZ.d
```

2. Open and make configuration changes to the listed conf file to coincide with the current Layer 3 configuration (i.e., adjust IP addresses and number of interfaces available). See GateD documentation if you have questions regarding the *conf* file.
3. Run the *zsync* command to save your changes. Be sure your changes are correct.

```
zsync
```

4. Reboot the switch. After rebooting, your switch operates as a Layer 3 switch with GateD routing.

Switch Administration 4

One of the main benefits of the ZT 8102 switch is that it runs a Linux operating system, so much of the switch administration is already familiar to most Linux administrators. Intel recommends complementing these instructions with a standard Linux reference guide, such as *Linux Network Administrator's Guide* available from O'Reilly.* The following is a brief description of some of the more common switch administrative tasks.

Note: If you are managing the switch via telnet and you shutdown the switch port that is being used for the telnet connection, the management session will be lost.

4.1 Setting the Root Password

The switch is shipped with a default user root and no password. To set the root password, use the password command:

```
sh-2.04# passwd
Changing password for root
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
Enter new password:
Re-enter new password:
Password changed.
sh-2.04#
```

Note: Even when just changing the password, you need to save the file system overlay with the *zsync* command, or you will lose your changes upon reboot.

4.2 Adding Additional Users

Additional users can be added with the *adduser* command. Additional users are desirable for connecting to the switch via *ftpd* and other daemons that require a login other than root and a password. To create a user named guest, run *adduser:*

```
sh-2.04# adduser guest
Changing password for guest
Enter the new password (minimum of 5, maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
Enter new password:
Re-enter new password:
Password changed.
sh-2.04# zsync
sh-2.04#
```

4.3 Setting up a Default Route

If you wish to access the switch from some place other than a directly attached network, you may want to setup a default route. Use the route command to set a default gateway.

```
route add default gw 10.0.0.254
```

Put the entry into the */etc/init.d/*rcS startup script to automatically set a default route upon reboot.

4.4 Name Service Resolution

Name service lookups will be done locally using */etc/hosts*. You can also tell the switch which name server to use by including an entry in */etc/resolv.conf*.

4.5 DHCP Client Configuration

A utility is included to dynamically determine the IP address of the switch interfaces. To set the IP address dynamically, execute the command:

```
dhclient zhp0
```

The default device name, *zhp0*, works with the default configuration of the switch and will attempt to obtain an IP address from the local DHCP server. To use DHCP to set your IP addresses automatically on boot up, uncomment the following line in */etc/init.d/rcS* by removing the # sign:

```
/usr/sbin/dhclient zhp0
```

4.6 DHCP Server Configuration

The switch includes a DHCP server. To start the DHCP server, configure */etc/dhcpd.conf* for your network, and run

```
dhcpd
```

Consult Linux Network administration manuals for more information on DHCP and configuration options.

To use DHCP to set your IP addresses automatically on boot up, uncomment the following line in */etc/init.d/rcS* by removing the # sign

```
dhcpd
```

4.7 Network Time Protocol (NTP) Client Configuration

NTP is a protocol for setting the real time clock on a system. There are numerous primary and secondary servers available on the Internet. For more NTP information and a list of available NTP servers, see the following URL:
http://www.eecis.udel.edu/~ntp

You will need to have your network settings properly configured to reach an available NTP server on your local network or the internet. To set the time and date, execute *ntpdate* with the server of your choice. For example,

```
ntpdate –u ntp.ucsd.edu
```

The *–u* is required if the switch is operating behind some types of firewalls.

If you wish for *ntpdate* to set your date and time automatically each time you boot, uncomment the example *ntpdate* command line in /etc/init.d/rcS by removing the # sign. *ntpdate* returns the Universal Time (UTC, formerly Greenwich Mean Time, or GMT). To display the local time, set the TZ variable to the appropriate name and the number of hours offset from UTC. For instance,

```
export TZ=PST8
```

for Pacific Standard Time offset from UTC by eight hours. To set an environment variable, add the entry to /etc/profile. Remember to *zsync* to make your changes permanent.

4.8 Network File System (NFS) Client Configuration

The switch includes an NFS client for mounting remote file systems. You will need to start NFS server processes in order to use NFS. You will need to start the following servers:

```
/sbin/portmap
/sbin/rpc.statd
/sbin/rpc.mountd
```

Once the above servers are started, you can mount a remote NFS file system.

```
mount rhost:nfs_file_system local_mount_point
```

If the remote NFS file system you're mounting is on an OA switch, you should mount with caching disabled.

```
mount rhost:nfs_file_system -o noac local_mount_point
```

All the necessary servers are included *in /etc/init.d/rcS* but are commented out by default. To automatically start all NFS client services each time you boot, uncomment the NFS Client servers. Go to *the /etc/init.d/rcS* file. Uncomment the following command lines by removing the # sign.

```
/sbin/portmap
/sbin/rpc.statd
/sbin/rpc.mountd -r
```

You can also include commands to mount remote NFS file systems at boot time. There is an example line included at the appropriate location in */etc/init.d/rcS*. Uncomment and alter the mount command included for your particular configuration.

Note: A “sleep” of five seconds is included to allow time for the links to come up prior to attempting the mount.

```
sleep 5
mount 10.0.0.1:/nfs -t nfs -o noac /mnt
```

4.9 NFS Server Configuration

The switch contains an NFS server that you can mount the switch file system from other systems. To enable the NFS server, first follow the steps to enable the NFS client. Then, edit /etc/exports to include the file systems you wish to export. Consult a standard Linux Network Administrator's Guide (or man pages) regarding options for exported file systems. Generally, an entry in /etc/exports looks like the following:

```
/nfs          *.localdomain.com(ro)
```

Now start nfsd to export the mount points and begin answering requests from remote clients.

```
/sbin/rpc.nfsd -r
```

To export file systems automatically on boot, edit /etc/init.d/rcS, uncomment the /sbin/rpc.nfsd command line by removing the #.

```
/sbin/rpc.nfsd -r
```

4.10 Connecting to the Switch Using FTP

Use FTP to transfer files to or from the switch. See a Linux reference guide for details of the FTP command. In general, you can use FTP to connect to any system running an FTP server to either get (transfer files from the remote host to the switch) or put (transfer files from the switch to the remote host) files.

```
ftp <remote_host>
```

4.11 FTPD Server Configuration

The switch itself can also be configured to run an FTP server (ftpd). See the Linux Reference Guide for details of the *ftpd* command. You will need to add a user to the switch in order to connect via ftp from a remote host, since root is not allowed ftp access. See the earlier section in this chapter regarding how to add a user. The FTP daemon is started by default. If you wish to shutdown the FTP daemon, comment out the betaftpd line in */etc/init.d/rcS*.

4.12 Connecting to the Switch Using TFTP

Trivial File Transfer Protocol or TFTP, is a very simple protocol used to transfer files. It is designed to be small and easy to implement. Therefore, it lacks most of the features of a regular FTP, like user authentication. You can use *tftp* to connect to any system running a tftp server (tftpd) including other switches.

```
tftp <remote_host>
```

4.13 TFTPD Server Configuration

The TFTP server is started by *inetd* using the configuration set up in */etc/inetd.conf*. The use of *tft* does not require an account or password on the remote system. Due to the lack of authentication information, *tftpd* will allow only publicly readable files to be accessed. The default location of these files is */tftpboot*.

4.14 SNMP Agent

Simple Network Management Protocol (SNMP) is the de facto standard for network management. An SNMP agent maintains a structure of data for a network device in a virtual information database, called a Management Information Base (MIB). A network management station is capable of accessing the MIB of the network device to monitor and configure the network device.

The switch utilizes the NET-SNMP (formerly UCD-SNMP) agent core. Additional information on the agent can be found at: http://www.net-snmp.com. The switch agent will respond to SNMPv1, SNMPv2, and SNMPv3 requests.

Protocols supported on the OpenArchitect switch by *gated*, such as RIP and OSPF communicate with SNMP agent via the SNMP Multiplexing (SMUX) protocol.

4.14.1 Supported MIBS

The switch includes MIB support as documented by each of the RFCs listed. The MIBs themselves are located on the switch in the */usr/share/snmp/mibs* directory.

RFC 1155:	Structure and Identification of Management Information for TCP/IP-based Internets
RFC 1227:	SNMP MUX Protocol and MIB
RFC 1493:	Definitions of Managed Objects for Bridges (obsoletes RFC 1286)
RFC 1657:	Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMI-V2
RFC 1724:	RIP Version 2 MIB Extension (obsoletes RFC 1389)
RFC 1850:	OSPF Version 2 Management Information Base (obsoletes RFC 1253, which obsoletes RFC 1252, which obsoletes RFC 1248)
RFC 2011:	SNMPv2 Management Information Base for the Internet Protocol Using SMIv2
RFC 2012:	SNMPv2 Management Information Base for the Transmission Control Protocol Using SMIv2
RFC 2012:	SNMPv2 Management Information Base for the User Datagram Protocol Using SMIv2
RFC 2013:	Management Information Base for Network Management of TCP/IP-based Internets: MIB-II (obsoletes RFC 1213, which obsoletes RFC 1158)
RFC 2021:	Remote Network Monitoring Management Information Base Version 2
RFC 2096:	IP Forwarding Table MIB
RFC 2571:	An Architecture for Describing SNMP Management Frameworks
RFC 2572:	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
RFC 2573:	SNMP Applications
RFC 2574:	User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
RFC 2575:	View-based Security Model (VACM) for version 3 of the Simple Network Management Protocol (SNMP)
RFC 2576:	Coexistence between Version 1, Version 2 and Version 3 of the Internet-standard Network Management Framework
RFC 2665:	Definitions of Managed Objects for Ethernet-like Interfaces

RFC 2674:	Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions
RFC 2742:	Definitions of Managed Objects for Extensible SNMP Agents
RFC 2787:	Definitions of Managed Objects for the Virtual Router Redundancy Protocol
RFC 2819:	Remote Network Monitoring Management Information Base
RFC 2863:	The Interfaces Group MIB (obsoletes RFC 2233, which obsoletes RFC 1573, which obsoletes RFC1229)
RFC 2932:	IPv4 Multicast Routing MIB
RFC 3165:	Definitions of Managed Objects for the Delegation of Management Scripts
RFC 3231:	Definitions of Managed Objects for Scheduling Management Operations
Intel Networks Private MIB	Custom Intel MIB to support software and hardware features not covered by standard MIBs.
UCD-SNMP Enterprise MIB	UCD-SNMP MIB related to management and monitoring of the LINUX host

4.14.2 Supported Traps

Upon certain events, the switch can be configured to send notification of the event, called an SNMP Trap out to a defined recipient/manager or managers. The switch will send SNMP traps for the following conditions:

SNMPv2-MIB:	coldStart
SNMPv2-MIB:	authenticationFailure
IF-MIB:	linkUp
IF-MIB:	linkDown
UCD-SNMP-MIB:	ucdShutdown
RMON-MIB:	risingAlarm
RMON-MIB:	fallingAlarm
VRRP:	vrrpTrapNewMaster
VRRP:	vrrpTrapAuthFailure
EGP (rfc1213):	egpNeighborLoss
BGP4-MIB:	bgpEstablished
BGP4-MIB:	bgpBackwardTransition

4.14.3 SNMP and Switch Interface Definitions

The switch defines three types of devices:

zre	physical port
zrl	trunk of ports
zhp	interface consisting of ports (zre's) and trunks of ports (zrl's)

A zrl (trunk device) is treated as an aggregate of its constituent zres (ports). A zhp is an aggregate of its immediately contributing sub-interfaces (zres and zrls). The ports that make up a trunk do not contribute to the zhp.

The administrative status of a zre and zhp are independent of each other. If the administrative status is down, then the operational status will be down independent of the underlying link state. To see the operational link status for a zre, use the ifconfig command. When the administrative status is up, the operational status is dependent on the underlying physical state. For example, if zhp0 contains zre1 and zre2 the following would be true for the operational status given the administrative status is up on zre1, zre2, and zhp0:

Table 4. Administrative Status Example

Physical Link Status		SNMP Operational Status		
zre1	**zre2**	**zre1**	**zre2**	**zhp0**
down	down	down	down	down
down	up	down	up	up
up	down	up	down	up
up	up	up	up	up

The administrative status is directly controlled by ifconfig up/down. The administrative status of the zhps and zres do not affect each other.

4.14.4 ifStackTable Entries

In the actual ifStackTable as shown in the MIB walk the following two OIDs (which denote ifIndexes) show the relationships.

ifMIB.ifMIBObjects.ifStackTable.ifStackEntry.ifStackStatus.0.1 = active(1)

ifMIB.ifMIBObjects.ifStackTable.ifStackEntry.ifStackStatus.0.2 = active(1)

If they are X.Y then

- if X = 0 there is nothing “above” this interface
- if Y = 0 there is nothing “below” this interface

Otherwise interface X has interface Y as a logical constituent.

4.14.5 SNMP Configuration

The SNMP agent is called *snmpd* and is started by default from the Linux boot up script */etc/init.d/rcS*. If you do not wish to start *snmpd*, comment out the snmpd line in */etc/init.d/rcS*.

Configuration of the switch SNMP agent is the same as configuration of any standard Linux host that uses the NET-SNMP agent. Configuration information for persistent data and security information is kept in snmpd.conf under the default SNMP configuration location, which for the switch is */usr/share/snmp*. snmpd.conf is the location to change “sys” information such as the *syslocation* and *syscontact*, as well as permissions such as the *rocommunity* or *rwcommunity*. The file also contains information regarding which traps are enabled and to whom traps should be sent.

Note: For NET-SNMP agents, the objects (sysLocation.0, sysContact.0 and sysName.0) are ordinarily read-write. However, specifying the value for one of these objects by giving the appropriate token in snmpd.conf makes the corresponding object read-only, and attempts to set the value of the object will result in a notWritable error response.

The processing for link up and link down traps is now user configurable. As the default, traps conform to RFC2863, meaning the trap contents will include:

```
ifIndex, ifAdminStatus and ifOperstatus
```

You can alter this behavior by specifying:

```
cisco_link_traps on
```

If cisco_link_traps are turned on as described then link up and link down traps will have a cisco-like format and the trap contents will include:

```
ifDescr and ifType
```

Examine and edit */usr/share/snmp/snmpd.conf* appropriately for your configuration. Information in */usr/share/snmp/snmpd.conf* is only read at startup - or when the daemon is forced to reread its configuration. See the standard Linux man page for snmpd.conf for more details.

4.14.6 SNMP Applications

The switch includes the *snmpget*, *snmpwalk*, and *snmpset* applications you can use these standard Linux utilities to test your SNMP agent. For example:

```
snmpwalk localhost -c public
```

will the entire MIB of the localhost starting at the top of the MIB. See the *Linux Reference Man Pages* for the usage of the SNMP utilities.

MIB values are decoded from their numerical representations into readable text by parsing MIBs located in the */usr/share/snmp/mibs/* directory. If you need to add a MIB, add it to that directory and *zsync* to save across reboots.

4.15 Port Mirroring

Zmirror_4800 sets packet mirroring from a given set of ports to a given port. Turning on packet mirroring causes a copy of the packet to be sent to the *mirror_to* port. There can be only a single *mirror_to* port, but there can be multiple *mirror_from* ports. The *zmirror_4800* command overwrites the *mirror_to* port. The *zmirror_4800* command accumulates the *mirror_from* ports.

Note: There are performance issues when trying to mirror more bandwidth than is available on the *mirror_to* port. Use the *zmirror_4800* command in the following way:

```
zmirror_4800 mirror_from mirror_to
```

After executing the following three commands, packets received on ports 0, 1 and 2 would be mirrored (copied and transmitted) to port 12. This mirroring would be in addition to any Layer 3 or Layer 2 switching.

```
zmirror_4800 0 12
zmirror_4800 1 12
zmirror_4800 2 12
```

To clear the current mirroring use the *-t* option.

```
zmirror_4800 -t 2 12
```

The *-e* option can be used to indicate that packets being sent on a given port should be copied to the *mirror_to* port. For example if the *-e* option is used as follows, the packets transmitted, as opposed to received, on ports 0, 1 or 2 would be mirrored to port 12.

```
zmirror_4800 -e 0 12
zmirror_4800 -e 1 12
zmirror_4800 -e 2 12
```

The *-l* option enables Layer 3 Packet mirroring. Layer 3 packet mirroring is off by default, and is enabled on a switch wide basis.

4.16 Link and LED Control

The *zlc* application sets the link speed and state of individual ports of the switch, or display their current state. It can also set or clear the extract led or the internal fault led, or to set a port down or up. To force the link on port 0 down:

```
zlc zre1 down
```

To check the status of a link:

```
zlc zre1 query
```

To check the status of all links:

```
zlc zre1..16 query
```

4.17 Link Event Monitoring

The *zlmd* application is intended to run as a daemon, waiting for a configured event to occur and then running the program configured for that event. The events monitored are changes in the link status at any of the 16 ports of the switch, the start of removal of the switch from the cPCI backplane, or the cancellation of the removal before it actually takes place. The program can be a shell script that initiates appropriate actions to respond to the event.

4.18 Web Interface (thttpd)

Web management is provided for basic configuration and monitoring of the switch. By default, the *thttpd* daemon is not started. Documentation for this program can be found online at http://www.acme.com.

4.18.1 Starting thttpd

To start the daemon, enter the following command exactly as shown or the web interface will not function properly.

- Change directories to root

  ```
  cd /
  ```

- Enter the following command

  ```
  /usr/sbin/thttpd -r -u root -c "\/http/cgi-bin/*"
  ```

By adding the previous command to /etc/init.d/rcS, the web server will start automatically when the switch is booted. Once *thttpd* is running, you can connect to OpenArchitect from your browser.

4.18.2 Restricting Access

By default, there is no password protection for the web interface. To setup password protection, follow these steps.

- Change directories to root
  ```
  cd /
  ```
- Create a password file and a new user
  ```
  htpasswd -c .htpasswd root
  ```

Notice the "." in front of the file name. Enter and confirm the password for the new user when prompted.

The password protection will only work properly when the password file is located in the root directory. Once a new user is created, reconnect to the web interface through your browser. You will be prompted for a username and password.

4.18.3 Web Interface

There are five buttons available with the following function: From left to right they are System, Ports, Vlan, IP, and Command. The default page loaded is the System page. By clicking on any of the buttons, a page will be displayed.

System	This is the default screen. The system page displays general information about the switch. All information except the switch version is obtained from SNMP. Because the information is gathered via SNMP, this page takes a few seconds to load.
Ports	Displays a graphical representation of the link LEDs. It shows the current link status for all ports. Link status is indicated by a green or red square. This page is automatically updated every five seconds. To view more information about any given link, simply click a link and an information table will be displayed to the right. The zre, duplex setting, duplex state, and transmit and receive information is all displayed. This page is also automatically updated every five seconds.
Vlan	Displays the current VLAN configuration on the switch. It displays which ports are assigned to which VLANs, and which ports are tagged or untagged. It also displays any trunks that have been created.
IP	Displays the current ip addresses assigned to interfaces. This information is obtained via SNMP.
Command	Allows the user to enter any commands as if they were at the console. On the left, there is a list of links that are the most common commands. By clicking one of these, the information will be displayed on the right. A command can also be entered manually. To keep someone from executing a command that might cause harm, all commands are checked against a list on the switch. To add more commands, they can be added to the */http/html/comcheck.html* file. Only then can they be executed in the web interface.

4.18.4 Customizing the Web Interface

Web pages can easily be added, modified, or deleted from the switch web interface. All files are located under the /http directory.

Note: If you are managing the switch via the web interface and you shutdown the switch port that is being used for the web interface connection, the management session will be lost.

High Availability (HA) Networking 5

High availability networking is achieved by eliminating any single point of failure through redundant connectivity. Redundant cables, switches and network interfaces for hardware, combined with HA software solutions on both the hosts and switches to control the HA hardware and maintain connectivity. Intel provides a complete solution by using Intel® Advanced Network Services (ANS) or Zynx's Rainlink drivers on the host and surviving partner on the switch.

Note: **The High Availability Networking feature set only applies to the ZT 8102HA Switch.** You must have an OpenArchitect/HA Suite RTU (Run Time User's) license for the chassis to use any OpenArchitect High Availability functionality or component on the switch within the chassis. The steps to installing the RTU are included in the sections that follow regarding configuration of Surviving Partner.

Please contact your Intel Sales Representative for more information.

5.1 Advanced Network Services (ANS) for Hosts

Intel ANS software allows you to setup adapter teaming for redundant LAN connections and load balancing network traffic across multiple server connections on System Master Processor Boards that use Intel LAN adapters. For example, Intel ANS can be enabled and configured on the Ethernet interfaces on the ZT 5524 and ZT 5515. Intel ANS supports the following Teaming Modes: Adapter Fault Tolerance, Switch Fault Tolerance, Adaptive Load Balancing, Receive Load Balancing, and Intel Link Aggregation.

For detailed information see http://www.intel.com/support/network/adapter/ans/teaming.htm.

5.1.1 RAINlink for Hosts

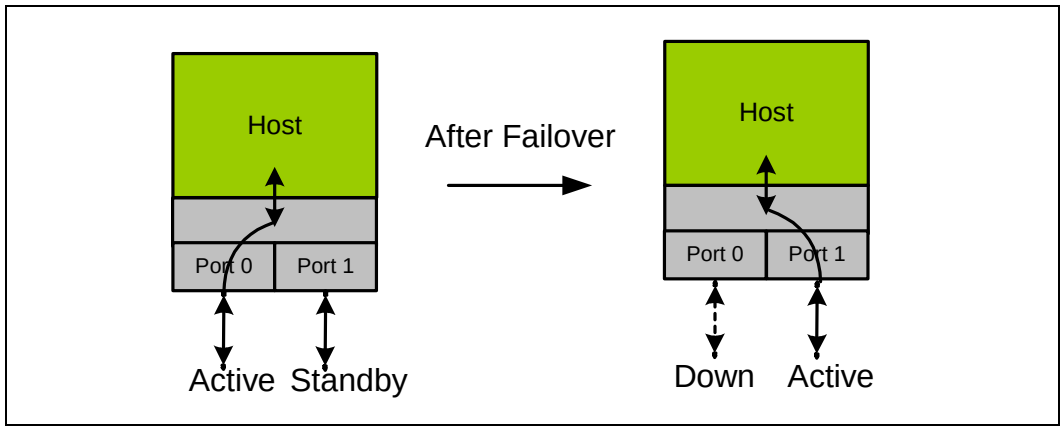
Zynx RAINlink provides IP Transparent Failover between two or more ports running in hosts that use ZNYX drivers. RAINlink presents a single, virtual interface to the protocol stack while managing multiple physical links. With RAINlink a failover between physical links can be made very quickly without requiring change to the IP or MAC address of the virtual interface, effectively transparent from the applications point of view. With redundant links from a switch (or switches) to the host, one link is maintained as the ACTIVE link and the other as STANDBY. If the ACTIVE link were to go down, RAINlink switches to the STANDBY, while presenting the same virtual interface to the host.

To contact your ZNYX Sales Representative, call (800) 724-0911, or send email to **sales@znyx.com**, or **support@znyx.com**.

5.1.2 Redundant Connections Using ANS or RAINlink

Redundant host connections using Intel ANS or Zynx RAINlink can provide an ACTIVE and STANDBY link to a switch, or more likely to provide redundancy between more than one switch.

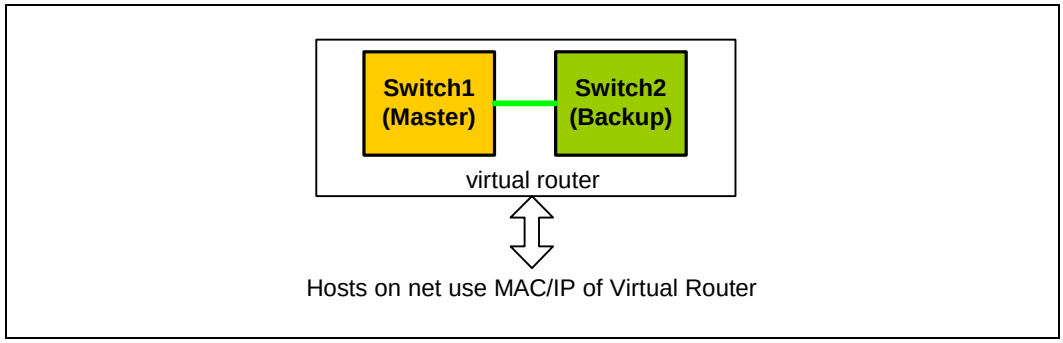
Figure 9. Failover Example



5.1.3 VRRP

Since most hosts use default router addresses (default gateway), the change of the default router address during a switch failover would require the end nodes to reconfigure. Layer 3 switches that failover must maintain the default router address to maintain the end node's IP transparent failover. The Virtual Router Redundancy Protocol (VRRP, RFC 2338) running in the Surviving Partner switch provides transparent movement of the default router address. VRRP maintains the notion of a Master switch and one or more Backup switches. This group of switches presents a virtual router IP address that can be used by hosts on that network as their default route, as shown in Figure 10.

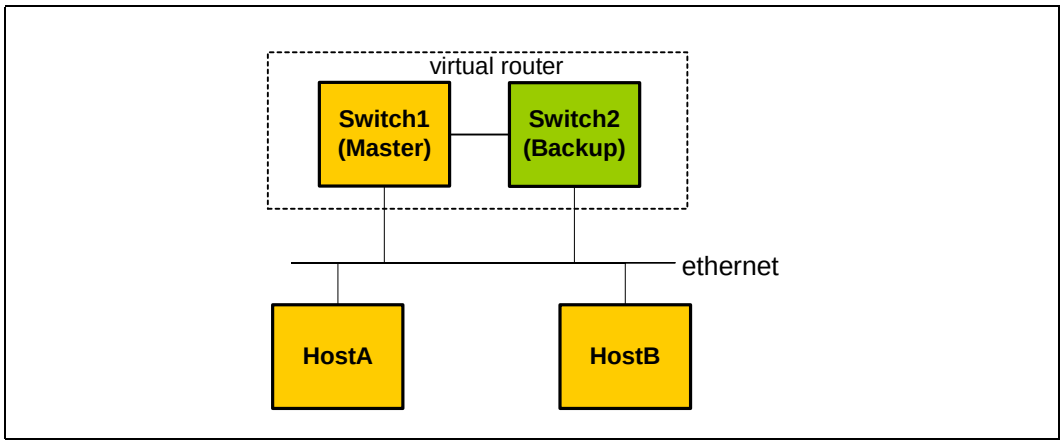
Figure 10. VRRP



If a Backup switch determines the Master switch is no longer available, one of the Backup switches will assume the role as Master. Physically, each switch maintains a link to the local network. Only the Master switch answers to the default gateway, and the hosts on that network have no need to relearn the router address.

In an HA configuration, the goal is to avoid any single point of failure. VRRP provides a mechanism to provide a static route for a local network, but a true HA configuration must also provide redundant connections for the host. Providing a virtual router for the local network is not enough. Take the simple case of two hosts on the local network with a connection to the virtual router. Each host needs a connection to each physical switch participating in VRRP. In the simplest configuration, each host would have one connection to the network as shown in Figure 11.

Figure 11. VRRP with Single Point of Failure on LA

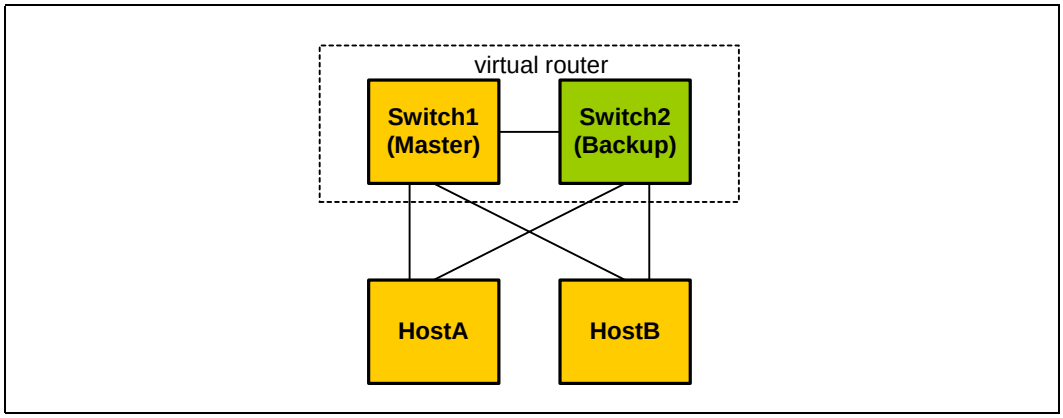


5.2 Switch Surviving Partner

Surviving Partner is a switch-based HA solution. Surviving Partner runs on the switches to provide transition of Layer 2 and Layer 3 switching functionality between two or more switches. Surviving Partner is comprised of many interactive protocols and processes including *VRRP*, *zlmd*, *zlc*, and others.

The problem with Figure 5-3 is that it assumes that each host has a single connection into a hub or switch that is then connected to the redundant virtual switch. A true HA solution would include redundant connections from each host to each switch in the virtual router, as shown in Figure 12.

Figure 12. VRRP with HA on LAN



Combining the features of Surviving Partner on the switches and Intel® Advanced Network Services (ANS) or Zynx RAINlink on the hosts allows implementation of this true HA configuration.

5.2.1 ZLMD

In addition to complete switch failover, single link failure must be properly handled. The Link Monitor Daemon *zlmd*, monitors the link status of each port. If a link goes down, *zlmd* communicates with the VRRP daemon (*vrrpd*) to change its priority. Changing the VRRP priority results in movement of switching functionality.

By combining *zlmd* with the *zlc* application, links connected to hosts that have not failed can be deterministically moved to the new master switch if desired. Supported modes:

switch The switch with the greatest number of UP links becomes the Master for all VLANs under HA management.

vlan The switch with the greatest number of UP links in that particular VLAN becomes the Master for that particular VLAN. If the switch has additional VLANs, they each change independently.

port The Master will remain the Master for that particular VLAN until all ports in that VLAN are down. The Backup then becomes the new Master for that VLAN. Failed links move their connectivity through the Backup Switch and the switch interconnect to reach the Master Switch. This option alleviates the need to move all nodes to a new switch just because a single link goes down

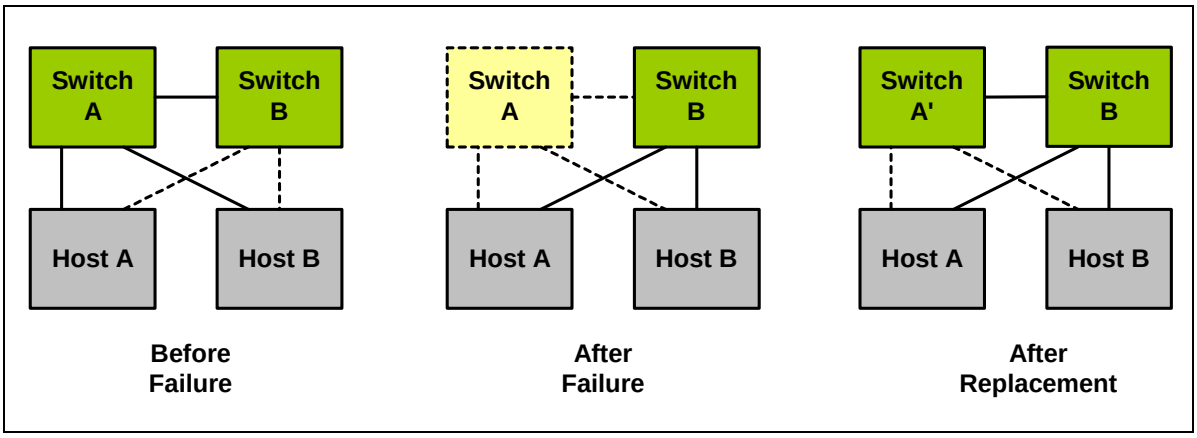
Note: Both VLAN Mode and Port Mode require inclusion of the interconnect in the VLAN. For the ZT 8102, when placed in a PICMG 2.16 chassis, the connection between the two fabric slots is an out of band port. VLAN Mode and Port Mode will not work over an OOB port.

5.2.2 Switch Replacement and Reconfiguration

When a switch fails, it must be replaced. The replacement switch will likely require proper configuration. For transparent switch replacement, the newly replaced switch must learn its configuration from its Surviving Partner.

Figure 13 illustrates a simple failover scenario. Host A and Host B are configured with Intel ANS providing IP transparent failover between two host ports, one port connected to Switch A and the other connected to Switch B. Assume Switch A provides connectivity between Host A and Host B. If Switch A fails, Intel ANS moves the active link on each host over to the port connected to Switch B. Surviving Partner software on Switch B recognizes that Switch A has failed, and assumes the role of switching traffic between Host A and Host B. When the failed Switch A is replaced with a new Switch A', Switch A' will learn its network configuration from the surviving partner Switch B. Switch A' is now ready as a backup to Switch B in case of failure of Switch B.

Figure 13. Failover Scenario



This is achieved through the use of DHCP. When a switch becomes a VRRP Master, a DHCP server is started with a pointer to a configuration file that contains configuration information for its partners. The replacement switch comes up running DHCP client to retrieve its configuration.

5.3 Configuring Surviving Partner

The *S60SP_startup* script is useful in setting up proper switch replacement. By factory installing the *S60SP_startup* script in replacement switches, the replacement switches will boot looking for a Master switch configuration. The *S60SP_startup* script works as follows:

1. It first looks for a local file /etc/rcZ.d/surviving_partner/zsp.primary.conf. If this file exists, it is used to configure the switch. Only the originally configured switch or *Central Authority* should contain this file. See Section 5.4, "Central Authority" later in this chapter for more information.
2. Next it uses *zspconfig –u* to attempt to contact a running Master switch to retrieve the proper configuration. This is the normal case for a replacement switch.
3. Finally, it lets the currently saved S70Surviving_Partner script execute. This case would be the case of a power up of an already configured backup switch when the other HA switch is unavailable. This case could occur after losing power to the entire chassis.

Proper configuration of Surviving Partner requires coordinated configuration of many different processes, including *vrrpd*, *zlmd*, *zlc*, and *dhcpd*. The daemon processes run scripts to perform their actions. Because these scripts are complex and inter-dependent, a configuration application called *zspconfig* is used to build them.

The basic steps to configuring Surviving Partner are:

1. Determine your desired configuration
2. Modify the configuration file (/etc/rcZ.d/surviving_partner/zsp.conf is the default) to use as input to the configuration utility (*zspconfig*)
3. Configure startup scripts or other scripts such as *gated* routing scripts and *vrrp* configuration scripts.
4. Run zspconfig on the Master system.

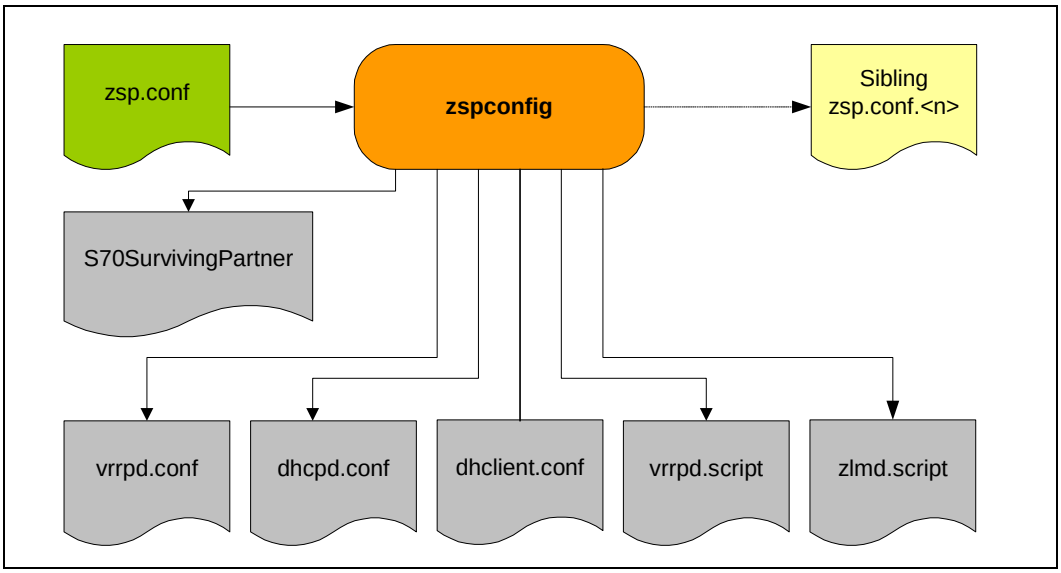
5. Run zspconfig –u on the Backup/Sibling system.

5.3.1 zspconfig

Zspconfig performs the job of building the scripts based on a provided input file locally, or from a remote machine. The result of *zspconfig* is to create several configuration files and runtime shell scripts, and optionally start the Surviving Partner processes. Scripts are generated for configuring VLANs, starting the network, and starting the *vrrpd* and *zlmd* daemons.

Zspconfig can also used by sibling backup switches to retrieve configuration from the Surviving Partner and start the *vrrpd* and *zlmd* daemons. *Zspconfig* is generally only run once to configure Surviving Partner. Figure 14 shows the scripts generated by *zspconfig*,

Figure 14. zspconfig Configuration and Runtime Scripts



The configuration and runtime scripts created are as follows:

S70Surviving_partner

Switch initialization script that is run at boot time. This script will restart the switch with the original configuration given to *zspconfig*. Optionally, *zspconfig* will run this script from the initial invocation.

zsp.conf.<n>

zspconfig configuration file that contains the configuration of the sibling backup switches. The <n> is used to distinguish potentially more than one backup switch. This configuration file is placed in */tftpboot*, and is retrieved via DHCP by a replacement switch on boot up.

vrrpd.conf

Configuration script for the VRRP daemon. This configuration is used when the *S70Surviving_partner* script launches vrrpd. There is a line in this file for each virtual router address vrrpd will manage.

dhcpd.conf

Configuration script used by *dhcpd* when the switch becomes master. *dhcpd* is used to serve replacement switches their configuration scripts. Namely a zsp.conf file that can be input to *zspconfig* with the -u flag.

dhclient.conf

If zspconfig is executed with the -u flag, a dhclient.conf file is created, and then dhclient is used to retrieve a *zspconfig* configuration file from the */tftpboot* area of the Master switch.

vrrpd.script

Runtime script that executes each time the *vrrpd* changes state. This script starts and stops *dhcpd*, and toggles down Intel ANS ports to force the Intel ANS nodes to a new Master switch.

zlmd.script

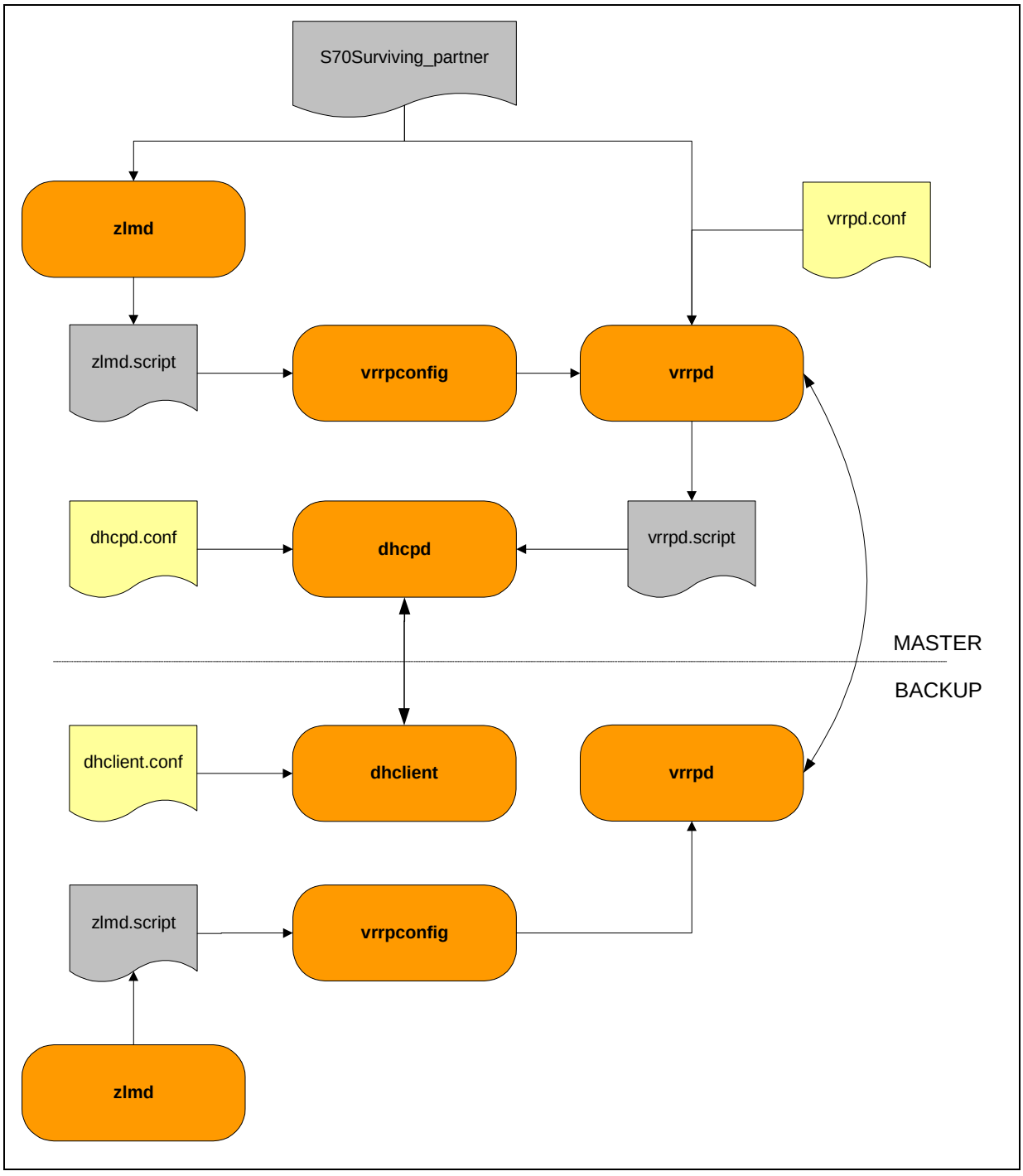
Runtime script executed by *zlmd* when a link goes up or down. This script modifies the priority of the *vrrpd* that in turn may cause the VRRP Master to move from one sibling switch to another.

After the scripts are created, *zspconfig* may run the *S70Surviving_partner* script to start the Surviving Partner tasks. The tasks started are *vrrpd*, *zlmd*, and *dhcpd*.

The *vrrpd* and *zlmd* daemons run scripts to perform their actions. When *vrrpd* changes state between Master and Backup, it runs a script that starts and stops *dhcpd*. When *zlmd* sees a link go up or down, it runs a script that communicates with *vrrpd* via *vrrpconfig*.

Figure 15 illustrates the tasks running on a pair of switches configured in a Surviving Partner setup. In this illustration, the upper switch is the Master and the lower is Backup.

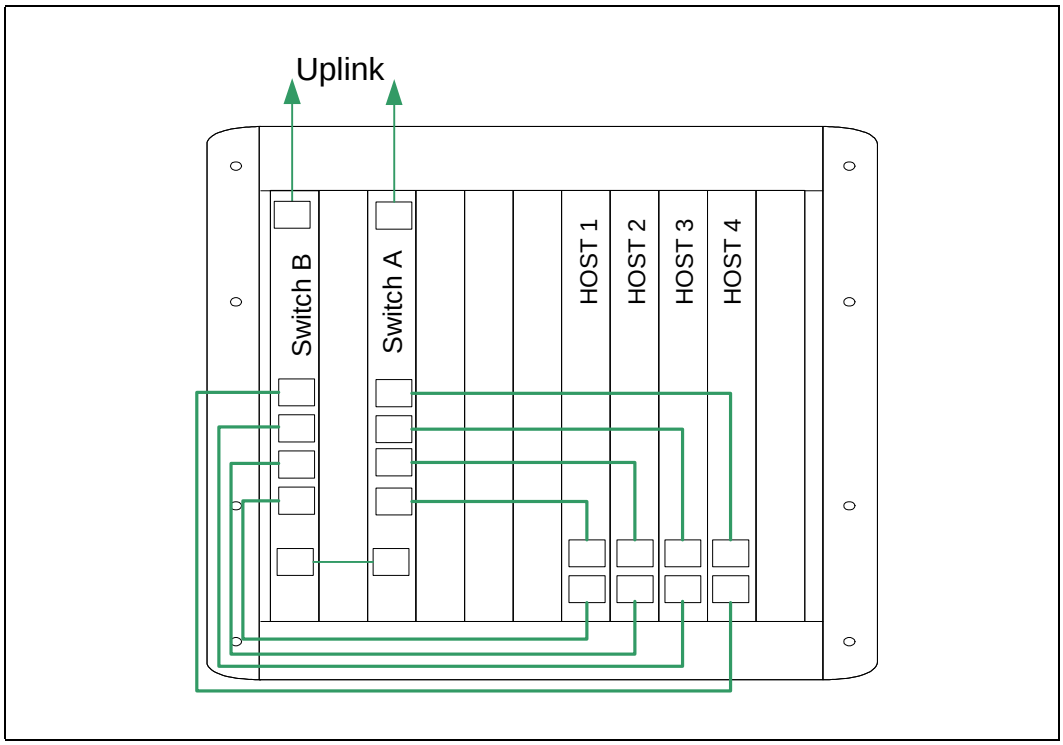
Figure 15. Surviving Partner Overview



5.3.2 Example HA Chassis Configuration

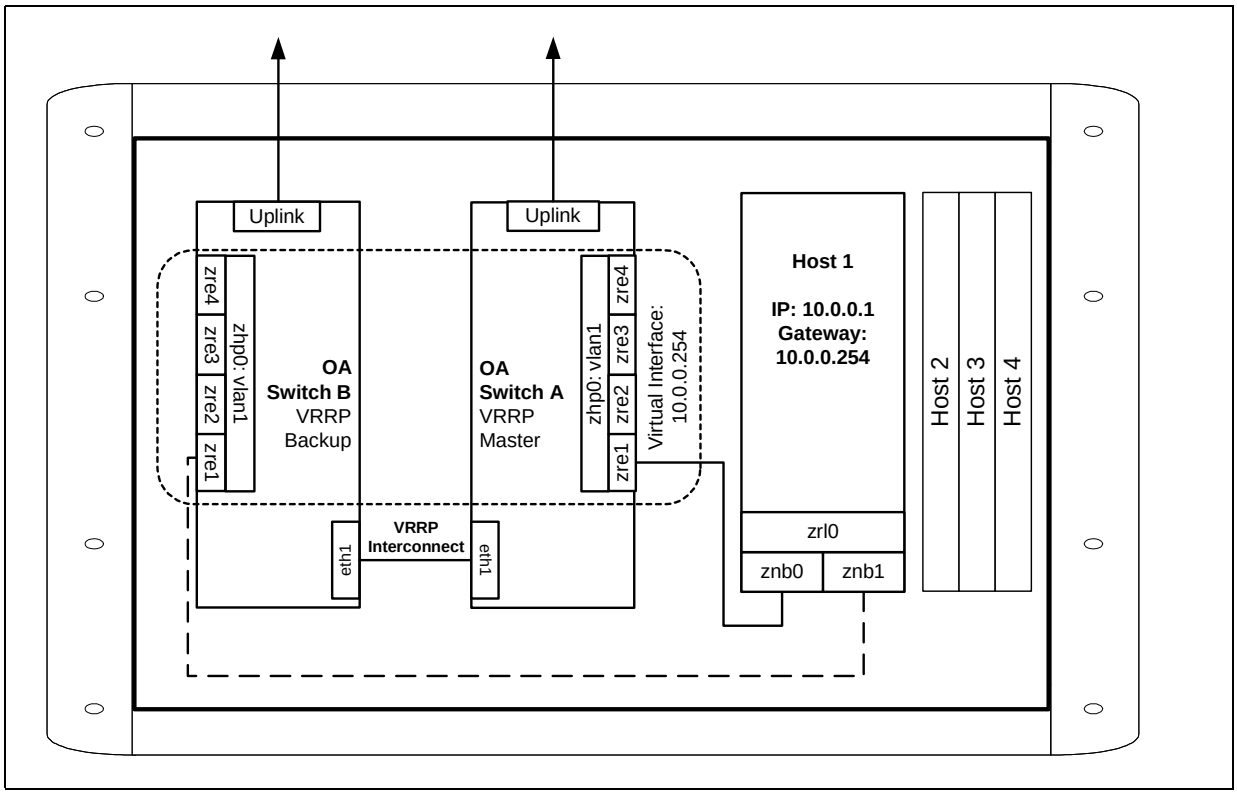
The following walks through a basic Surviving Partner configuration typical for an HA setup. Consider an HA chassis with four hosts, such as single-board CPUs, and two switches configured for Surviving Partner. Each of the hosts has two Ethernet ports providing a link to each of the switches. Each switch furnishes an uplink from the HA chassis out of the box.

Figure 16. HA Chassis



Each host runs Intel ANS or Zynx RAINlink with the two ports configured for failover. The Switch node provides one virtual interface (zrl0) to the host protocol stack. One physical port (znb) is connected to the Master Switch, and one physical port is connected to the Backup Switch. An interlink provides communication between the switches. Figure 17 shows detail of the two switches and a representative host configuration.

Figure 17. HA Example



In the above configuration, the hosts are all configured on the same VLAN. The switches are configured with ports 1-4 in a single VLAN, and the two switches are configured as Surviving Partners using VRRP to form a single virtual interface to the hosts. The LPF interconnect (*eth1*) has it's own VLAN. The Uplink ports can be configured many different ways, either as part of the same VLAN, a different VLAN, or a different VLAN running VRRP.

5.3.2.1 Modifying zsp.conf

An example file for setting up zspconfig is */etc/rcZ.d/surviving_partner/zsp.conf*. The following will document the settings to prepare Switch A and Switch B shown in Figure 17.

On Switch A (Master), make a backup copy of zsp.conf, and edit zsp.conf:

```
cd /etc/rcZ.d/surviving_partner
cp zsp.conf zsp.conf.save
vi zsp.conf
```

The first thing you need to do is add your Run Time User's (RTU) license required to utilize the HA features to zsp.conf. Replace the "xxxx" value with your license, and uncomment the line by removing the "#" sign.

```
# Copyright 2002 Intel, Inc.
# All Rights Reserved.
#
```

```
# You must have an OpenArchitect/HA Suite RTU license for the chassis
# to use any OA high availability functionality or component on the
# switches within the chassis or use OpenArchitect/Node on any of the
# CPU host cards. This includes executing this utility script.

# Please uncomment the line below and replace 'xxxx' with your
# OpenArchitect/HA license key. It must be terminated with a ';

# OAHA_RTU: xxxx;
```

The first section uses *zconfig* to create the VLANs. Create VLAN1 for the Hosts1-4 in the HA chassis, and VLAN100 for the VRRP Interconnect. This example uses untagged frames, so untag the ports. If you knew the configuration of the Uplink, you would treat that accordingly. For now, we'll ignore the Uplink:

```
zconfig zhp0: vlan1=zre1,zre2,zre3,zre4;
zconfig zre1,zre2,zre3,zre4=untag1;
```

Notice that the interconnect port (eth1) is the LPF port, which is an out of band (OOB) port. This prevents the use of *vlan* or *port* switching modes because for those modes, each VLAN that presents a virtual address must include the interconnect port.

The next section sets up the physical IP addresses to use for the Master and each of the Sibling (Backup) switches. The Master provides the addresses to the Siblings on a first come, first serve basis. Note that the physical IP address should be different from the virtual IP address that spans the entire Sibling (Virtual Router) group. Once configured, the Sibling group appears as one connection point to other hosts on the VLAN. You need to supply an IP address for each interface on each switch, including the interconnect.

```
sibling_addresses: zhp0 = 10.0.0.100, 10.0.0.101;
sibling_addresses: eth0 = 192.168.0.100, 192.168.0.101;
```

Now configure the virtual address for each sibling group. In our diagram, we are going to create a virtual interface across VLAN1 of each switch. To Hosts 1-4 on that VLAN, this provides a single point to connect to, or route through depending on your Uplink. For the setup above,

```
vrrp_virtual_addresses: zhp0 = 10.0.0.254;
```

Next come port definitions, as defined on the *zspconfig* man page. Since our hosts are connected using Intel ANS, we will want to choose Intel ANS on each of the ports in VLAN1 on each switch, and interconnect for the interconnect port on each switch,

```
interconnect: eth1;
Intel ANS: zhp0;
```

The next sections are optional. The first determines the failover mode between the Surviving Partner switches. There are three modes:

```
switch    The switch with the most links becomes the Master (default)
```

vlan The switch with the most up links in the VLAN becomes the Master. *NOTE: This requires an in-band port as the interconnect.*

port The Master will remain the Master until all ports in the VLAN are down. The Backup then becomes the new Master. *NOTE: This requires an in-band port as the interconnect.*

The remaining entries provide a mechanism to propagate files and/or startup scripts to sibling switches. An example might be startup scripts or scripts to configure gated. Example scripts are included to start gated with RIP1, RIP2, or OSPF setup. You must use absolute path names.

Once the configuration files are complete, run the *zspconfig* utility on the Master to configure all the scripts:

```
zspconfig –f zsp.conf
```

Once configuration is complete, insure there are no superfluous S-type startup scripts *in /etc/rcZ.d*, and *zsync* your switch to save your configuration.

Now go to the backup switch and run *zspconfig –u* to get the appropriate configuration information from the Master,

```
zspconfig –u eth1
```

5.4 Central Authority

Modifications can be made to the *S60SP_startup* script to use a third machine running DHCP that is not part of the Surviving Partner pair. The third machine is referred to as the *Central Authority* which is usually a Linux server.

Setup requires a DHCP daemon configuration file on the Central Authority and a *dhclient* configuration file for each of the two Surviving Partner switches in the pair. The format of the DHCP daemon configuration file is dependent on the machine and operating system being used. An example can be obtained from the Surviving Partner primary switch in the location */etc/rcZ.d/surviving_partner/dhcpd.conf*.

This configuration will contain configuration for only one of the two Surviving Partner switches. It must be edited. For example:

```
subnet 100.0.0.0 netmask 255.0.0.0 {
        option broadcast-address 100.255.255.255;

        host INTEL1 {
                fixed-address 100.0.0.31;
                option dhcp-client-identifier “Intel”;
                option vendor-encapsulated-options “zsp.conf.1”;
        }
}
```

A second host entry must be added with unique information.

```
subnet 100.0.0.0 netmask 255.0.0.0 {
        option broadcast-address 100.255.255.255;

        host PRIMARY {
                fixed-address 100.0.0.30;
                option dhcp-client-identifier “PRIMARY”;
                option vendor-encapsulated-options “zsp.pri-
    mary.conf”;
        }
        host SECONDARY {
                fixed-address 100.0.0.31;
                option dhcp-client-identifier “SECONDARY”;
                option vendor-encapsulated-options “zsp.second-
    ary.conf”;
        }
}
```

The *zsp.primary.conf* and *zsp.secondary.conf* files must be placed in the *tftp* location on the machine, often */tftpboot*. The *zsp.primary.conf* and *zsp.secondary.conf* files can be retrieved from the Surviving Partner switches. This is the configuration that will be given to the switches. It is recommended that the zsp.conf be taken from the primary as follows:

1. The *zsp.conf* file created by hand on the primary is moved to */tftpboot/zsp.primary.conf* on the Central Authority.
2. Move */tftpboot/zsp.conf.1* file on the primary created by *zspconfig* to */tftpboot/ zsp.secondary.conf* on the Central Authority.
3. Create *dhclient.conf* files on the Surviving Partner switches. Examples can be found in */etc/ rcZ.d/surviving_partner/dhclient.conf*. As an example:

   ```
   send dhcp-client-identifier “Intel”;
   request vendor-encapsulated-options;
   require vendor-encapsulated-options;
   ```

4. Modify the dhclient.conf file on the primary switch as follows:

   ```
   send dhcp-client-identifier “PRIMARY”;
   request vendor-encapsulated-options;
   require vendor-encapsulated-options;
   ```

5. Modify the dhclient.conf file on the secondary switch as follows:

   ```
   send dhcp-client-identifier “SECONDARY”;
   request vendor-encapsulated-options;
   require vendor-encapsulated-options;
   ```

The last step is to modify the startup scripts that run *zspconfig* to use the -c option. The -c option allows you to provide a dhclient.conf script rather then having *zspconfig* create a default. For example, the *S60SP_startup* script line that reads:

```
echo y y n | zspconfig -t 10 -su zhp0 > /dev/null 2>&1
```

Can be modified to

```
echo y y n | zspconfig -c /etc/rcZ.d/surviving_partner/dhclient.new.conf -t 10 -su
    zhp0 > /dev/null 2>&1
```

If you use *S60SP_startup*, the */etc/rcZ.d/surviving_partner/zsp.primary.conf* file should not exist. This way the *S60SP_startup* script will first look at the Central Authority. If the Central Authority is down, then it will use its current configuration.

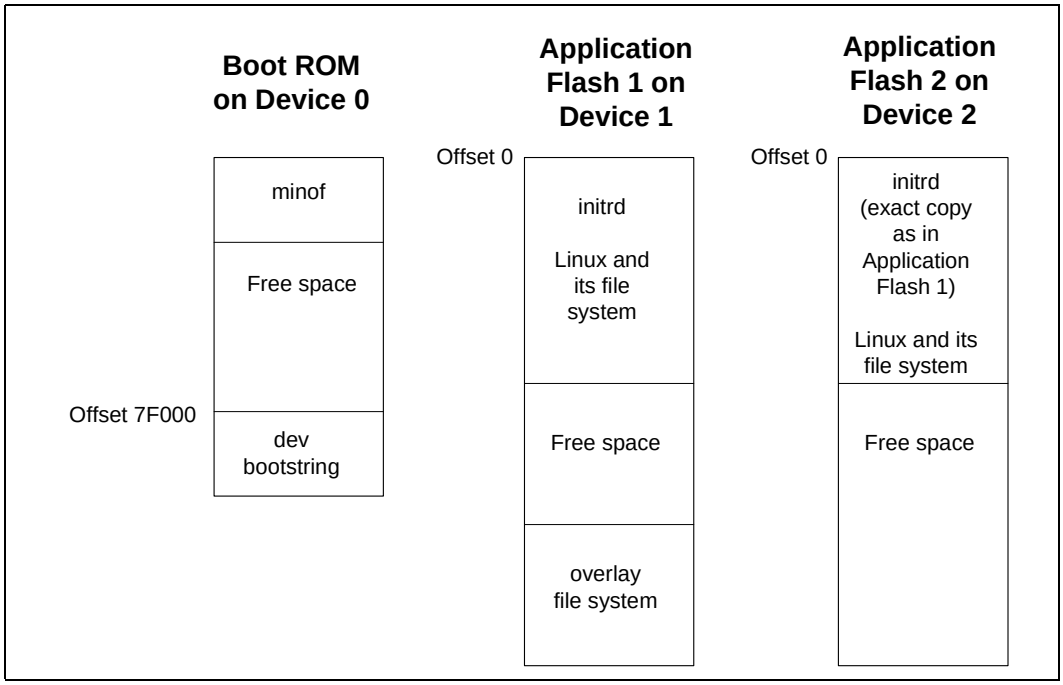
Switch Maintenance 6

This chapter includes basic information about the ZT 8102 switch environment including an overview of the file system structure, modifying and updating switch files, upgrading the switch driver and kernel, and implementing a system recovery.

6.1 Overview of the OpenArchitect switch boot process

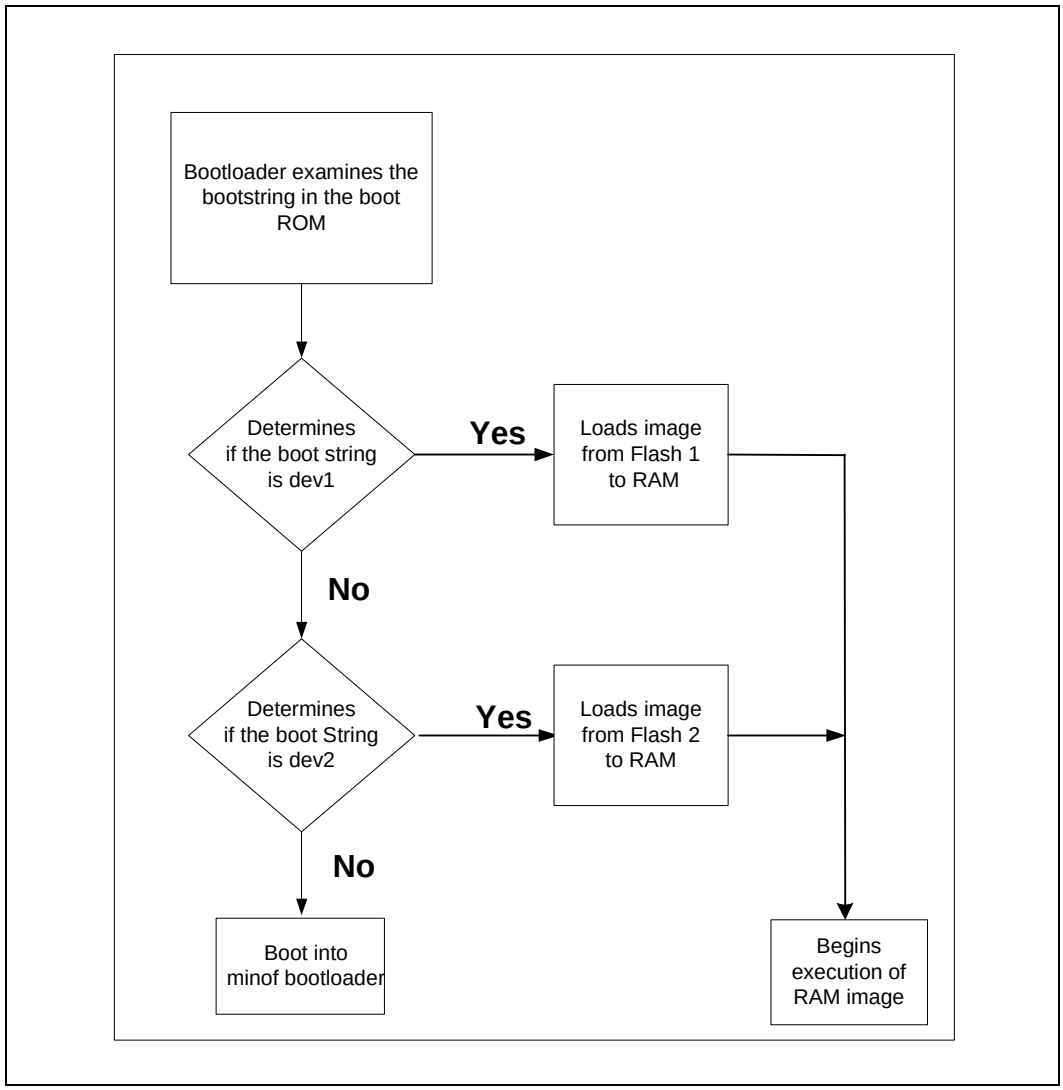
The switch is equipped with a Random Access Memory (RAM) disk and three Read Only Memory (ROM) devices, including, a boot ROM and two application flash.

Figure 18. ROM Devices in OpenArchitect



The boot ROM is located on device 0 and contains the OpenArchitect minof application that operates as a boot loader and includes a device bootstring. Device 1 contains the application flash 1 image of the Linux operating system and the switch overlay file system. Application flash 1 is the primary working image for the switch. Device 2 contains the application flash 2 that is an exact copy of application flash 1. You would only boot from this device if application flash 1 is corrupted and you need to restore the switch to the factory-shipped configuration.

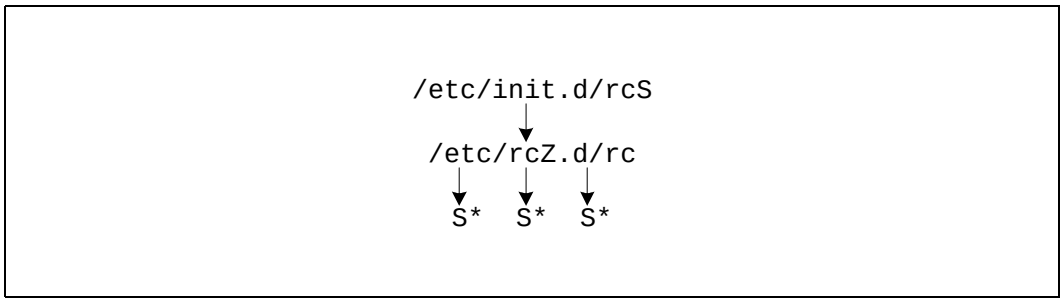
Figure 19. Boot Process Flow



Under normal circumstances, the booting up process follows the process outlined in Figure 6-2. During boot up, the *minof* bootloader reads the device bootstring to locate and validate the correct application image to load. The bootstring command is in the following format:

devX | [<options>] where X represents the device value 1 or 2

Figure 20. Init Script Flow



6.2 Saving Changes

Any modifications made to the scripts for your particular configuration must be properly saved or your changes are lost when you reboot. The file system for the switch only exists in memory. A rewriteable overlay is contained within the upper four megabytes of the first application flash.

6.3 Modifying Files and Updating the Switch

Any file in OpenArchitect can be added, deleted or modified, with the exception of /sbin/init, /usr/sbin/zmnt, /lib/modules/zfm_c.o, and the /tmp directory. Files are saved across a system reboot by running the script zsync.

A directory /.zsync contains database files used by *zsync* for managing the file system overlaying process. The user should not modify the files in this directory or unpredictable results may occur.

6.4 Recovering from a System Failure

If the switch does not function after you initially change or reconfigure the image, you have several options for recovering from an error. First, try to *telnet* into the switch. If you are successful, remember to run *zsync* after fixing your problem. If you cannot *telnet* to the switch, attach a console cable to the switch and manage the via the RS-232 Console Management Port.

6.4.1 System Boots with a Console Cable

After attaching the system console cable, if the system boots, fix the problem that does not allow you to *telnet* to the box, run *zsync*, and reboot. The problem is likely to be in the configuration files contained in /etc/rcZ.d. In order to telnet into the box, there must be a configured interface with a proper IP address. For example, *zhp0* is configured with the IP address 10.90.90.90 in the factory default configuration.

6.5 Booting with the –i option

If you cannot *telnet* into the switch and Linux fails to boot, it is likely that a change saved by *zsync* has left the switch in an inaccessible state. To allow users to recover from mistakes saved in the overlay file system, a boot argument of –i passed to the init process will stop the un-tarring of the saved overlay files. As a result, the system boots to the factory shipped configuration.

- Connect through the console port. During boot up, the system displays the Linux boot string. "Linux/PPC load: "for five seconds. During the five second pause, enter the boot option "-i" and press Enter:

```
Linux/PPC load: root=/dev/ram init=/sbin/init -i
```

- Initiating the *–i* option of *zbootcfg*.

```
zbootcfg-d 1-i
```

- Reboot the system. After the reboot, clear the –i option from the boot string. Enter the following command:

```
zbootcfg-d 1
```

The reboot command will also take "-i" as an option and pass it to the Linux boot,

```
reboot -i
```

- When the system boots, the overlay file system is returned to the factory-installed configuration. At this point, you have a few options.
- Run *zsync* and the factory-installed system will be restored to your flash. NOTE: All changes you have made and saved prior to the *zsync* command will be lost.
- Restore particular files from the existing overlay. Use the *zmnt* command to mount the overlay in a designated directory and copy back just the changes you want to keep from the existing overlay. For example, if you wanted to recover your /etc/hosts file from the existing overlay, use *zmnt* to mount the overlay in a designated directory, like /tmp, then copy /tmp/etc/hosts to /etc/hosts. Lastly, use *zsync* to save your changes.

```
zmnt /tmp
cp /tmp/etc/hosts /etc/hosts
zsync /etc/hosts
```

- Reboot the system.

6.6 System Hangs During Boot

After attaching the system console cable, if the system hangs during boot, try booting with the *–i* option as described in the previous section. It is possible that important Linux system files became corrupted and incorrectly saved in the flash overlay. Use *zmnt* as described in the previous section to fix or remove the problem files from the overlay. If the system will not boot with the *–i* option, then proceed to the *Booting the Duplicate Flash Image* section in this chapter.

6.7 Booting the Duplicate Flash Image

Another recovery method, if Linux fails to boot, is to temporarily boot the factory-installed duplicate image located in the second flash device.

1. Connect through the console port.

2. When you see the following, hit ESC to break into the boot loader. Depending on your terminal emulation application, you may have the enter ESC twice.

```
OpenArchitect Firmware  v1.1.x
Copyright (c) 2001, 2002 Intel Networks

Testing memory...... ok.
Copying loader image to 0x01000000

Press ESC to interrupt auto-boot.
```

3. At the monitor prompt, type:

```
boot dev2
```

4. The system should boot from the image in the second device.
5. At this point, follow the steps in the *Upgrading the OpenArchitect Image* section in this chapter to put a new RAM disk image in the application flash 1.

Warning: Be sure not to program flash 2, since this is your only bootable image at this time.

6. The command to program flash 1 should be similar to the following command. The image name may be slightly different depending on the version of the image:

```
zflash -d 1 rdr4800.zImage.initrd
```

6.8 Upgrading the OpenArchitect Image

1. Use *telnet*, or preferably, attach a console cable to the switch, and login to the switch. If you are connecting via telnet, be aware that the upgrade process will reset the switch to the default IP address of 10.90.90.90, so you will have to be able to reach 10.90.90.90.
2. Download the OpenArchitect image to a local system.
3. The OpenArchitect image is very close to the limit of free space available on a default system so you may need to clear some space prior to downloading the OpenArchitect image to the switch. Check for free space with the “df” command.
   One of the easiest ways to create free space is to remove /usr/sbin/gated. The application will be replaced during the update procedure. Once you have enough free space, proceed.
4. From the switch console, *ftp* the new OpenArchitect (rdr) image from the local system to your switch.
5. The switch has two flash devices available device 1 and device 2. Use the *zflash* command to write the new OpenArchitect image into the first flash device.

Note: Make sure that Surviving Partner is not running before using zflash. The delays incurred while zflash writes the flash can cause the Surviving Partner daemons to think there is a failure, resulting in link oscillation.

```
zflash -d 1 <image_file>
```

6. The image file will be something named similar to the following,

```
zflash -d 1 rdr4800.zImage.initrd
```

6.9 Upgrading or Adding Files

Follow the procedure below to upgrade or add a new file to the switch. Place the file you are adding or upgrading into the appropriate location in the file system. Save the file in the overlay directory area on the application flash by running *zsync*.

```
zsync
```

After running *zsync*, the file is saved to the flash for future reboots.

6.10 Excluding Saving Files to Flash

Specific files or directories can be excluded from saving to flash by zsync by including an entry in /etc/exclude. Likewise, existing entries in /etc/exclude such as /tmp can be removed in order to save those files to flash with zsync.

6.11 Using Apt-Get

The utility apt-get was created by the Debian Linux community to allow remote fetching and installation of software stored in a repository in Debian package format. It allows users to keep their software up-to-date with the latest binaries, and install new software without the need to recompile.

A repository of updated and add-on Debian packages for OpenArchitect-based switches resides at openarchitect.znyx.com and is accessible through either FTP or HTTP. The file /etc/apt/sources.list contains a line-by-line listing of all apt-get repositories. Users may also create their own repositories and add entries in /etc/apt/sources.list for their private access methods.

For example, assume there is a small utility that allows a user to modify the L2 table included in the repository called mac-address. The utility requires a library called znyx-lib to function properly. The Debian packages have been created with the Debian dpkg and debhelper utilities. To use apt-get to retrieve the utility, first download the list of available packages from the repositories listed in /etc/apt/sources.list,

```
apt-get update
```

Now that apt-get knows about all the packages currently in the repository, use apt-get to install the package,

```
apt-get install mac-address
```

The process also determines that mac-address relies on znyx-lib and installs the library. Once finished, the mac-address utility can be run on the switch. Use zsync to make the changes permanent.

Removing a package will also remove it's dependencies, For instance if you were to remove the library:

```
apt-get remove znyx-lib
```

The mac-address utility would also be removed. Finally, because space is at a premium on the switch, it is a good idea to cleanup after managing packages:

```
apt-get clean
```



Switch Commands 7

OpenArchitect* applications are implemented above the OpenArchitect libraries and the RMAPI interface. OpenArchitect applications are used for normal operation of the switch, for runtime status and diagnostics, and for prototyping new applications development.

For runtime operation, the OpenArchitect applications perform initialization and configuration, and real-time control and maintenance of the switching tables in the switch silicon. Protocol support is performed by the Linux* operating system. In turn the OpenArchitect applications communicate with Linux to determine the appropriate switch table setup.

The initialization of the switch is completed by the *zconfig* application. Through configuration scripts, the user can setup any combination of Layer 2 and Layer 3 switching configurations with VLAN support. Running the *zconfig* command causes network interfaces to be presented to the Linux operating system. These interfaces can be setup for Layer 2 bridging functions such as Spanning Tree Protocol, or Layer 3 routing through the Linux operating system.

zl2d is run as a daemon to monitor the Linux operating system bridging function and update the switch silicon accordingly.

zl3d is run as a daemon to monitor the Linux operating system routing table information and update the switch silicon switching tables accordingly.

For gathering statistics or prototyping applications, there are OpenArchitect applications that allow any register or table in the switch to be read or written. These applications include *zreg*, *zstats*, and *zarl* and all of the different table equivalents.

vrrpconfig

NAME

vrrpconfig – Configure and control the running vrrpd

SYNOPSIS

```
vrrpconfig [-d <level>] -- <vrrpd parameters>

vrrpconfig [-d <level>]  [-a]  [-k] [-p] [-s <vid>]
```

DESCRIPTION

vrrpconfig provides communication with a running *vrrpd* daemon. The -- option for *vrrpconfig* will pass all parameters to *vrrpd* as would be done when starting the *vrrpd*. Any output generated by *vrrpd* is displayed on the *vrrpconfig* controlling tty. Any action normally taken by *vrrpd* for the given parameter is done so by *vrrpd*. Reference *vrrpd* for *vrrpd* parameters and their usage.

OPTIONS

vrrpconfig also has a set of local options that are not passed to *vrrpd* directly. Many do, however, retrieve information from the running *vrrpd*. The local options are as follows:

-d <level>	Set the debug level. The default debug level is 1. The higher the level, the more debugging output is produced. Debugging output is sent to the controlling tty. This debugging output is from *vrrpconfig*. To set the debug level of *vrrpd*, one would use the *vrrpd* debug level setting option placed after the -- in the *vrrpconfig* command line.
-a	Display in a user readable format, information about the current state of all the Virtual Routers controlled by *vrrpd*.
-k	Kill *vrrpd*. The entire daemon is killed. Running this command will require that *vrrpd* be restarted.
-p	Display relevant SNMP table values.
`-s <vid>`	`Print a numeric representation of the state of the Virtual Router associated with the Virtual Router Identifier <vid>. The numeric representations are 0 = INIT, 1 = BACKUP, and 2 = MASTER`

EXAMPLES

Here is an example of using the -- invocation method that changes the priority to 99 for the Virtual Router associated with the Virtual Router Identifier 1:

```
vrrpconfig -- -v 1 –p 99
```

SEE ALSO

vrrpd

vrrpd

NAME

vrrpd – Virtual Router Redundancy Protocol Daemon

SYNOPSIS

```
vrrpd -i ifname -v vrid [-f piddir] [-s] [-a auth] [-p prio]
   [-nhbe] [-I ifname] [-d delay] [-m address] [-S script]
   [-c conf_file] [-D level] ipaddr
```

DESCRIPTION

vrrpd is an implementation of Virtual Redundant Routing Protocol (VRRPv2) as specified in RFC2338. It runs in Linux user space. In

short, VRRP is a protocol that elects a Master server on a LAN to which the Master answers to a *virtual IP address*. If it fails, a Backup server takes over the IP address.

VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN. The VRRP router controlling the IP address(es) associated with a virtual router is called the Master, and forwards packets sent to these IP addresses. The election process provides dynamic fail over in the forwarding responsibility should the Master become unavailable. This allows any of the virtual router IP addresses on the LAN to be used as the default first hop router by end-hosts. The advantage gained from using VRRP is a higher availability default path without requiring configuration of dynamic routing or router discovery protocols on every end-host.

OPTIONS

The following options are supported by vrrpd:

-h	display the usage line
-n	Don’t use the virtual MAC address
-b	Run vrrpd in foreground
-i <ifname>	the interface name on which to run the Virtual Router. Machines connected with the named interface will see the Virtual Router Address move with the Master switch
-I <ifname>	the interface name on which to communicate with other VRRP routers for management of the Virtual Router (default is the -i interface)
-v <vrid>	the id of the Virtual Router Identifier [1-255]. This value must be a unique value, one per Virtual Router. In other words there is a unique vrid to ifname associated with the –i option.
-s	Toggle preemption mode (Enabled by default). Preemption means that a Master switch will go to Backup if a current Backup has higher priority.
-M	Become MASTER when priority is equal. Be sure it is only set on one host or the switches will oscillate. Must set -B option on other hosts (requires preemption mode ! -s)
-B	Become BACKUP when priority is equal. See -M option
-S <script>	script to be called with state change occurs
-a <auth>	(not yet implemented) set the authentication type auth=(none|pass/hexkey|ah/hexkey) hexkey=0x[0-9a-fA-F]+

-p <prio>	Set the priority of this host in the virtual server (default is 100)
-f <piddir>	specify the directory where the pid file is stored (default is /var/run)

-c <conf>	Configuration read from conf file (required when managing multiple Virtual Routers). The contents of the conf file are lines of command line options. Each line represents a Virtual Router. Parameters given on the command line apply to all Virtual Routers defined by the conf file. So for example, if the command line reads:
	
	`vrrpd –d 50 –c vrrpd.conf`
	
	And the vrrpd.conf file contains:
	
	`-v 1 –i zhp0 –I zhp3 10.0.0.42`
	`-v 2 –i zhp1 –I zhp3 11.0.0.42`
	
	vrrpd would be started controlling two Virtual Routers; one for 10.0.0.42 and the other for 11.0.0.42. They would both get a –d 50 option.

-d <delay>	Set the advertisement interval. Default is one second. By default, time is specified in seconds. If the delay value ends in a lower case ‘m’ the time is specified in milliseconds. The millisecond specification results in a proprietary use of the VRRP Adver Int field.
-m <address>	Change the virtual MAC address from 00:00:5E:00:01:<vid> to the provided addr. The addr should be input as 6 two digit hex numbers that are colon delimited with no spaces. The –n option overrides the change made with –m. The result of which to use the native MAC address of the interface. Using the –n option is not recommended.
-D <level>	Set debugging output to the supplied level
<ipaddr>	the ip address(es) of the virtual server

SEE ALSO

vrrpconfig

zal

NAME

zal, zhgtable, zhostmac, zipf, zipmc, zl2mcvlanbc, zl4table, zlinkagg, zmac, zmirror, zpttable, zrule, zvmc, zvrp, zvrpout, - Formatted display of OpenArchitect switch tables.

SYNOPSIS

```
zal [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zhgtable [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zhostmac [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zipf [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zipmc [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zl2mcvlanbc [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zl4table [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zlinkagg [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zmac [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zmirror [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zpttable [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zrule [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zvmc [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zvrp [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]

zvrpout [-p <ppa>][-i <index>][-m <mac_address>][-a][-c]
    [-v <vlan_id>][-h <host_name>] [-d <level>]
```

DESCRIPTION

The switch table display functions produce formatted output of the switch tables for display on the user console. The format of the output is table-dependent. Headers describing the column being displayed are printed after every 22 lines of output, which makes it easy to pipe through *more*. The switch has keyed and non-keyed tables. Keyed tables grow and shrink as entries are added and deleted. The non-keyed tables are always the same size, and entries are

written into specified locations. By default, the table functions print all entries for non-keyed tables and all active entries for the keyed tables. Several options are available which enable the user to display only selected entries. Additionally, there is an option that clears entries in the table. The zal functions are not mapped. The tables reflect the true display of the 5416 functions, and bypass mapping. Use zgr to display the tables with the ports mapped to the order indicated to other applications and on the front panel.

OPTIONS

-h <host_name> Specifies which hostname to connect. By default, zal connects to the locally connected OpenArchitect switch (i.e., the one that is on the local PCI bus).

-d <level> Sets the level of debugging output required by zal. The default level is one (1). Setting the debug level higher produces more output. Four (4) is currently the maximum output level.

-p <ppa> Specifies the Physical Point of Attachment (PPA). Each switch that is controlled by the CPU and run by zreg is a unique PPA. If there is only one switch, such as when zreg is running on the embedded processor, the PPA is zero (0). The default PPA is 0.

-m <mac_address>
Displays entries whose MAC address field matches <mac_address>. Only valid for tables that have a MAC address field. Cannot be combined with -v or -i.

-v <vlan_id> Displays entries whose VLAN ID field matches <vlan_id>. Only valid for tables that have a VLAN ID field. Cannot be combined with -m or -i.

-i <index> Displays the entry at the <index> position in the table. Valid for all tables. Cannot be combined with -m or -v.

-a Displays the entire table. Do not stop when all the active entries in the table have been displayed. The -a flag can produce a lot of output for many of the tables.

-c Clears the entries in the table. Any entry that would be displayed if the -c flag had not been used will be cleared. Exercise caution with the –c flag, as clearing entries out of some tables can stop the switch from functioning, particularly when zl3d is running.

EXAMPLES

The following example searches for and displays an entry in the L2 table with the specified MAC address:

```
zmac -m 00:c0:95:45:00:00
```

If there is an entry in the L2 table with the MAC address, 00:c0:95:45:00:00, all the fields of that entry will be displayed.

The following command deletes the above entry:

```
zmac -c -m 00:c0:95:45:00:00
```

```
The following command displays all entries of the PTTABLE:
```

```
zpttable
```

SEE ALSO

zreg

zbootcfg

NAME

zbootcfg – Modifies the boot parameters of the OpenArchitect switch.

SYNOPSIS

```
zbootcfg -a | -d <device number> [<boot_string>]
```

DESCRIPTION

zbootcfg is used to display or modify the boot parameters on the switch. The boot parameters are utilized by the minof boot loader application to indicate on which device to find a boot image. Care should be taken when changing the boot string. Incorrect procedures can result in a switch that cannot boot.

-d <device_number>
: Three ROM boot devices are available in the switch. The factory-shipped boot device is 1. The following describes each boot device:

-d 1
: Boot image located at offset 0 in the application flash 1. This is the factory-shipped location of the primary OpenArchitect image.

-d 2
: Loads an image located at offset 0 in the application flash 2. This is the factory-shipped location for the alternate OpenArchitect image.

Any characters after the -d <dev> parameters are saved in flash memory and passed unchanged to the booting kernel.

OPTIONS

-a
: Displays the current boot string. The default factory shipping string is “dev1.”

-d <dev>
: Specifies the ROM device from which to boot. The <dev> value must be the number 1 or 2 corresponding to applica-

tion flash 1 or application flash 2, respectively.

<boot string> Optionally, a boot string may be provided that is passed to the booting kernel. All characters after the -d <dev> are passed unchanged to the booting kernel.

EXAMPLES

```
The following example illustrates a command for making the im-
age boot from the second application flash. Typically this is
required before updating application flash1. By booting the al-
ternative image, if a failure occurs during the programming of
application flash 1, recovery is easier.

zbootcfg -d 2
```

The next example passes the *-i* option to the booting kernel. This is useful when recovering from a mistake saved to the read-write file system or after updating the application flash 1 and doing the first boot. The *-i* option prevents the read-write file system from overwriting the initial RAM disk image.

```
zbootcfg -d 1 –i
```

SEE ALSO

zflash, reboot

zconfig

NAME

zconfig - Configures the OpenArchitect switch.

SYNOPSIS

```
zconfig [-h <host_name>] [-d <level>] [-a] [-t] [{-f
   <configuration_file>} | <configuration>]
```

DESCRIPTION

zconfig creates Virtual Local Area Network (VLAN) groups of switch ports or trunks. Each VLAN group forms a Layer 2 switching domain. Each VLAN group has a VLAN Identification number (VID) that can be carried in a tag field, located in the header of packets traveling on that VLAN. The configuration of a port determines whether a packet transmitted from that port includes the VLAN tag. A set of up to eight ports may be configured as a trunk, with all links from these ports connect to the same link partner. For each VLAN group created by *zconfig*, a network interface is also created. After the network interface is started by *ifconfig*, the VLAN group performs Layer 2 switching. The network interface can be used for Layer 3 routing between VLAN groups.

A network interface uses the following format:

zhpN (e.g., zhp0)

N-is an integer between 0 and 9999. The value of N is not required to be the same as any of the port(s) that are its members. The range 0-4999 is reserved for network interfaces created by users. The range 5000-9999 is reserved for network interfaces created by switch applications.

A trunk uses the format *zrlK*, where K is an integer between 0 and 5.

OPTIONS

-h <hostname> Specifies the hostname to configure. By default, zconfig configures the local OpenArchitect switch.

-d <level> Sets the level of debugging output produced by zconfig. The default level is 1. Setting the debug level higher produces more output. The maximum output level is currently four(4).

-a Displays the current configuration of the switch.

-t Tears down the entire switch configuration.

{-f <configuration_file>} | <configuration>
Gets configuration information from the specified file. A <configuration_file> name of '+' reads configuration data from standard input. If the -f flag is not used, a single line of configuration data can be entered as parameters to zconfig.

CONFIGURATION SYNTAX

zconfig takes configuration data from standard input or from a file with the -f option. In either case, the configuration syntax is the same. The zconfig configuration data consists of a list of semicolon-delimited statements. Each statement specifies an action to take on an interface. An interface is one of three types: a network interface called Intel host port (zhp); a switch port interface called zre; or a trunk interface called zrl.

Comments, spaces and new lines are ignored. Comments begin with the # character and include characters through the next new line.

Trunk Interface Statements

A trunk interface statement begins with the trunk name followed by an equals sign and an action. Trunk interface statements are used to create or tear down trunks or define the rule to be used to determine which member of the trunk should be used to transmit a packet based on the following rules.

Protocol	Data
L2 Multicast	DA_{MAC}
TCP/IP	$SA_{MA}C + DA_{MAC} + SA_{IP} + DA_{IP} + SPort_{TCP} + DPort_{TCP}$
UCP/IP	$SA_{MA}C + DA_{MAC} + SA_{IP} + DA_{IP} + SPort_{UDP} + DPort_{UCP}$
Other IP	$SA_{MAC} + DA_{MAC} + SA_{IP} + DA_{IP}$
Other	$SA_{MAC} + DA_{MAC}$

Note: For the ZT 8102 the rules are set on a global basis, not per trunk. Setting rules on one trunk will set the rules for all trunks in the switch.

```
For example:

zrl0 = <Trunk Interface Action>;
```

There are three possible trunk interface actions:

List of ports — Creates a trunk interface with the specified port members. All of the ports specified must not be a part of any other trunk, or be individually included in any network interface. Up to eight ports can be included in a trunk.

A port member is identified with the zre<X> format, where x represents a port number between 0-23 for 10/100 FX ports, and 24 or 25 for 1000FX ports (A or B, respectively).

teardown — Removes the trunk interface, making the ports which were part of the trunk available for configuration in other trunks or VLANs.

mac | ip {source_address | destination_address | source_address destination_address}

Specifies the criterion for selecting which port in the trunk a packet should be transmitted out of. The last three bits of the address specified, or the exclusive OR of the last three bits of the two addresses specified, are used to index the currently active members of the trunk to select the port. If ip addresses are specified, and the packet is not an ip packet, the mac addresses are used instead.

Examples of trunk interface statements:

This statement creates a trunk containing three ports:

```
zrl5 = zre11,zre15,zre17;
```

```
The following statement specifies that packets will be sent out
over this trunk using the exclusive OR of the last three bits
of their mac source and destination addresses to select the
port:
```

```
zrl5 = mac source_address destination_address;

The teardown statement uses a colon instead of an equals sign:

zrl5: teardown
```

Note: For the ZT8102, the rules are set on a global basis, not per trunk. Setting rules on one trunk will set the rules for all trunks in the switch.

Network Interface Statements

The syntax for a network interface statement is the interface name followed by a colon and an action. Network interface statements are used to create or tear down a VLAN group and can consist of one or a list of network interface names; followed by a colon and then an action. For example:

```
zhp0: <Network Interface Action>;
```

Network interface actions may include:

vlan<N> = list of ports or trunks
: Creates a network interface and a VLAN group with a VLAN identification number (VID) consisting of specified port members.

<N>
: is an integer between 1-4095.

list of ports or trunks
: A port member is identified with the zre<X> format, where x represents a port number between 1-16 for 10/100/1000 ports, and 15 or 16 for 1000FX ports. A trunk is identified with the zrl<Y> format, where Y represents the port number between 0-5. . The Out of Band ports, port 17 and LPF cannot be included in the list of ports.

  If the network interface and VLAN group already exist, the specified ports or trunks are added to the network interface and VLAN group.

teardown
: Deletes the network interface and the associated VLAN group.

zre_list = multicast <mac_address>
: Register the multicast <mac_address> on the zre_list ports associated with the given VLAN

multicast_clear
: Clear all registered multicast address on all the ports in the VLAN

<list of ports or trunks> teardown
: Deletes the specified ports or trunks from the network interface and the VLAN group associated with it. If there are no remaining port or trunk members, then also deletes the network interface and VLAN group.

Examples of Network Interface Statements:

The statement below creates a VLAN group with the VID number 1 and the network interface named zhp5. This VLAN includes a single switch port, zre1.

```
zhp5: vlan1=zre1;
```

The next statement creates a VLAN group with the VID number 100 and the network interface named zhp1. This VLAN includes four switch ports, zre1, zre10, zre11, zre13.

```
zhp0: vlan100 = zre1,zre10,zre11,zre13;
```

The next statement adds two switch ports, zre1, zre2 and zre3, to an existing network interface and VLAN.

```
zhp0: vlan100 = zre1..3;
```

The next statement deletes two switch ports, zre1 and zre2, to an existing network interface and VLAN.

```
zhp0: zre1..2 teardown;
```

The final example is a teardown action that deletes the VLAN group defined in the previous example, including the network interface.

```
zhp1: teardown;
```

Port Interface Statements

Port interface statements specify a port or trunk name or a list of such names; followed by an equal sign (=) and then the action. Port interface actions may include:

SYNTAX

```
zconfig <zre_list>=untag<n>
```

untag<N> Packets sent from this portor trunk for VLAN <N> are transmitted without a VLAN tag. The port or trunk specified must have previously been included in the VLAN group with VID<N>.

zconfig <zre_list> multicast=<forward_type>

<forward_type> Set the ports in the specified VLAN group to act as defined by the forward_type for multicast traffic. Possible <forward_types> are:

forward_unregistered(default)
forward_all

filter_unregistered

Examples of Port Interface Statements:

Assuming that zre1 has been assigned to VLAN 1, to specify that packets sent from port 1 for VLAN 1 are transmitted without a VLAN tag, and packets arriving on this port without a VLAN tag are given the VLAN tag with the VID number 1, enter:

```
zre1=untag1;
```

If port 0 is also a member of VLAN 100, packets for VLAN 100 are sent from this port with a VLAN tag as part of their header.

In the next example, the switch ports 10, 11, and trunk 2 are configured as untagged members of VLAN 100.

```
zre10,zre11,zrl2=untag100;
```

This statement is equivalent to the following three lines:

```
zre10=untag100;
zre11=untag100;
zrl2=untag100;
```

In the examples above, since port interfaces can only be untagged for one VLAN group, zre1 cannot also be untagged for VLAN 100. A port or trunk can be a member of multiple VLANs but can only be designated untagged on one VLAN.

WILDCARDS

Wild card characters can be included to simplify the process of creating larger, more complex configurations. Wild card characters for *zconfig* include:

, (comma)	Use for creating lists
.. (dot-dot)	Specifies an inclusive range
+ (plus)	Specifies auto-incrementing

Below are some examples for the correct usage of the comma (,) and dot-dot (..). Each line below produces the same results:

```
zhp0: vlan1 = zre1, zre2, zre3, zre4;
zhp0: vlan1 = zre1..4;
zhp0: vlan1 = zre1, zre2..4;
zhp0: vlan1 = zre1..2, zre3..4;
```

The following examples create multiple VLAN groups using a single

statement. A list of network interface names are followed by a colon (:) and a list of VLAN actions which are followed by an equal sign (=) and a port list. Each VLAN group is created in turn, along with the corresponding network interface, and all ports listed after the equal sign are included in each group.

The following statement creates 14 VLAN groups with VID numbers 1-14. Each VLAN contains the same switch port, port 1, represented as zre1.

```
zhp0..13: vlan1..14 = zre1;
```

The plus (+) wildcard can be used with the last port listed to auto-increment that port number before each VLAN group is created. The following network interface statement creates 14 VLAN groups, with the first group containing port 1, the second group port 2, and so on. The second statement configures all ports as untagged in their respective VLANs.

```
zhp0..13: vlan1..14=zre1+;
zre1..13=untag1+;
```

This is equivalent to:

```
zhp0: vlan1=zre1;
zhp1: vlan2=zre2;
zhp2: vlan3=zre3;
    .
    .
zhp13: vlan14=zre14;

zre1=untag1;
zre2=untag2;
zre3=untag3;
    .
    .
zre14=untag14;
```

The previous configuration can be used for creating a 14 port Layer 3 switch, with each port assigned to its own VLAN.

In the next example, one VLAN group, with VID number 1, is created that contains 14 ports. The second statement designates the 14 ports as untagged for the VLAN 1 group.

```
zhp0: vlan1 = zre1..13;
zre1..13 = untag1;
```

The previous configuration can be used for creating a 14 port Layer 2 switch, all 14 ports assigned to the same VLAN.

SEE ALSO

zl3d

zcos

NAME

zcos – class of service queue control

SYNOPSIS

```
zcos [-p <ppa>][-h <hostname>] [-d <level>]
     [-m q0,q1,q2,q3,q4,q5,q6,q7 [block]]
     [-c {1|2|4} [-s {32|64}] <port list>]|
         [-n <max packets> [-t <max time>] <queue list>]
     [-q <[all] | [ {qmap|qinfo} [port list]] | [scheduler
         [queue#]]>]
```

DESCRIPTION

The OpenArchitect switch supports one, two or four queues of packets to be sent out each of the 16 ports or forwarded to the CPU. There can be a different number of COS queues for each port and the CPU. Packets have an 802.1p priority of 0-7. The 802.1p priority for a packet is determined by one of the following:

1. The priority value in the tag control field for tagged packets.
2. The COS_SRC field in the ARL entry for the source address of the packet.
3. The COS_DST field in the ARL entry for the destination address of the packet.
4. The COS_DST field in the MARL entry for the multicast destination address of the packet.
5. The CPU_PRI field in the CONFIG register for BPDUs and other control messages sent to the CPU.
6. The PRI field from an IRULE entry which is matched.
7. The COS field in the DMA control block for a packet sent by the CPU. Priority is usually determined by applying the rules in order, although COS_SRC may have priority over COS_DST if SCP is set in the matched source entry in the L2 table. Except for the COS field in the DMA control block, the value obtained is the 802.1p priority value; it is mapped to one of the four COS queue values to determine which queue the packet will go into for each port it is sent out. If there are only two COS queues for a port, only the least significant bit of the mapped COS queue value is used. When there is only one queue, all packets for the port go into that queue. The COS field in the DMA control block directly specifies the COS queue to be used, bypassing any mapping.

   Each port also has a final egress queue that holds either 32 or 64 packets. Packets are moved from the COS queues to the output queue whenever there is room and transmitted in order from the output queue. Packets are taken from the highest numbered queue first, then lower numbers until queue 0 is used. The maximum number of packets to be taken from a queue before moving on to the next is set separately for each of the four COS values. It

defaults to 0, which means to exhaust the packets in a queue before moving on to the next. It is also possible to specify a maximum allowable latency time of from 16 to 4095 microseconds for each of the four COS values. If no packet from a queue has been processed for the specified time, a single packet is transmitted from that queue before continuing with the current queue.

The *zcos* application sets the number of class of service queues for each port in the switch and provides the mapping between packet priorities and COS queues. It is also used to configure the algorithm used for selection of packets from each queue class for egress.

OPTIONS

-p <ppa> Specifies the physical point of attachment (PPA). This is used to uniquely identify the switch being used. By default, this parameter is set to 0 to imply the current OpenArchitect switch where you are running this command. Ignore this option if you are configuring on the OpenArchitect switch on which the zcos command is being run.

-h <hostname> Specifies the hostname of the OpenArchitect switch to be configured. Ignore this option if you are configuring on the OpenArchitect switch on which the zcos command is being run.

-d <level> Sets the debug level. The default is 1. The maximum is 4.

-m q0,..,q7 Specifies the COS to queue map for incoming packets. The first parameter maps COS0 to queue number q0, the second maps COS1 to queue number q1, etc. (valid range is 0 to 3). The default behavior is that all ports use the specified map. Optionally, a COS to queue map may be specified for a grouping of ports called a “block.” The blocks are numbered as follows:

Block	Ports
0	0-7
1	8-15
2	16-23
3	Gigabit Port A
4	Gigabit Port B

-c [{ 1|2|4} [-s {32|64} <port list>]

sets egress port queue properties.

-c defines the number of queues (one of 1, 2 or 4 queues)

-s defines the size of the egress queue in packets (either 32 or 64).

port list — is a mandatory entry that is a comma separated list of ports specified as zre<n> (where n is 1-16 for in band ports; and cpu for the CPU.

[-n <max packets> [-t <max time>] <queue list>]
sets scheduler parameters where:

- max packets — defines the weight for the specified queue(s). Valid range is 0 to 255.
- max time — defines the maxlatency timer. This value is in microseconds, although the timer granularity is 16 microseconds. Therefore, the values entered are rounded to the nearest 16 microseconds. The maximum value is 4080 microseconds. A value of zero (anything less than 8) turns off the timer.
- queue list — defines a list of queues, comma separated.

[-q <[all] | [{qmap|qinfo} [port list]] | [scheduler [queue#]]>]
queries currently configured parameters.

- all — displays all of the queue mappings, queue configuration parameters, and scheduler configuration for all ports and queues.
- qmap — displays the COS to queue mappings. If port list is not specified, then details for all ports will be displayed.
- qinfo — displays configured details for the egress queues. If port list is not specified, then details for all ports will be displayed.
- port list — is a comma separated list of ports specified as zre<n>, where n is 1-16 for Ethernet ports 1-16 and cpu for the CPU.
- scheduler — displays configured scheduler information. If queue number (from the range 0-3) is not specified, then parameters for all the queues are displayed.

EXAMPLES

To set all 16 10/100/1000 Ethernet ports to use 2 COS queues, with a 32 entry Egress queue:

```
zcos –c 2 –s 32 zre1..16
```

To set the CPU port to use 4 COS queues, with a 32 entry Egress queue:

```
zcos –c 4 –s 32 zre27
```

To map 802.1p priorities 0-3 to COS0, 4-5 to COS1, 6 to COS2 and 7 to COS3:

```
zcos -m 0,0,0,0,1,2,3
```

To set scheduling algorithm for COS3 to strict priority with minimum latency and all other COS queues to round robin:

```
zcos -n 0 -t 1 3
zcos -n 1 0,1,2
```

SEE ALSO

zfilterd, zirule, iptables

zfilterd

NAME

zfilterd - A daemon to use the fast filter processor (FFP) of the OpenArchitect switch for filtering based on *iptables* rules.

SYNOPSIS

```
zfilterd [-h <hostname>][-d <level>] [-p <ppa>] [-r <usecs>]
   [-c][-D <port>]
```

DESCRIPTION

zfilterd is a daemon that intercepts filtering rules entered by the user, checks them for validity and then enters them into the FFP. By default, *zfilterd* performs these actions every 30 seconds, although this may be overridden with the -r <usec> option.

OPTIONS

-h <hostname>	Specifies which hostname to connect. By default, zconfig connects to the locally connected OpenArchitect switch (i.e., the one that is on the local PCI bus).
-d <level>	Sets the level of debugging output required by zconfig. The default level is one (1). Setting the debug level higher produces more output. Four (4) is currently the maximum output level.
-p <ppa>	Specifies the Physical Point of Attachment (PPA). Each switch that is controlled by the CPU and run by zreg is a unique PPA. If there is only one switch, such as when zreg is running on the embedded processor, the PPA is zero (0). The default PPA is 0.
-r <usec>	Sets the refresh interval to <usec> microseconds.
-c	Clears the filter table and disables filtering.

`-D <port>`
: `Sets the destination port for DROP action to <port>.`

SEE ALSO

zrule, iptables

zflash

NAME

zflash Loads images into the flash ROMs on the OpenArchitect switch.

SYNOPSIS

```
zflash -d <dev> [-o|-O <offset>] <image_file>
```

DESCRIPTION

Zflash enables you to program the flash ROMs on the switch. The switch contains 3 flash ROM devices: the boot ROM flash, application flash 1 and application flash 2. Care should be taken when flashing new images into the switch. Incorrect procedures can result in a switch that cannot boot; especially when flashing the boot ROM referred to as device 0. See *Switch Maintenance* for updating the flash ROM images.

OPTIONS

-d <dev>
: Specifies the ROM device being programmed. Dev must be a number 0, 1, or 2 corresponding to the boot ROM, application flash 1, or application flash 2 respectively.

-o <offset>
: Specifies the offset into the device or memory where the image should be loaded. The offset is specified in megabytes. To specify an offset not on a megabyte boundary, use -O. Default offset is 0

-O <offset>
: Specifies the offset in bytes. Default offset is 0

EXAMPLES

The following example loads a new initial RAM image into application flash 1 at the default offset of 0:

```
zflash -d 1 rdr4800.zImage.initrd
```

The following example loads a new boot image into the boot rom at the default offset of 0:

```
zflash -d 0 zx4800.img
```

```
Exercise caution when using this command, as an error can ren-
der your switch inoperable. Do not interrupt this process until
complete.
```

SEE ALSO

zbootcfg

zgmrpd

NAME

zgmrpd - GARP Multicast Registration Protocol (GMRP) daemon for the OpenArchitect switch.

SYNOPSIS

```
zgmrpd [-d <level>] [-f] [-h <hostname>] [-p <ppa>] [-t <tar-
   get>]
```

DESCRIPTION

zgmrpd is run after the network interfaces are created and initialized with *zconfig*, and started with *ifconfig*. *zgmrpd* starts a background task that implements the GARP Multicast Registration Protocol (GMRP) protocol for a specified interface, either a *zhp* or *bzhp*. GMRP provides a Layer-2 mechanism for determining which ports in a VLAN are listening to which multicast addresses. *zgmrpd* updates the multicast table in silicon with this information, so that the multicast traffic is forwarded only to the ports in a VLAN to which listening hosts are attached. This optimizes packet flow through the switch. The background task started by *zgmrpd* continues throughout the life of the Layer 2 network. GMRP is specified in *ANSI/IEEE Std 802.1D, 1998 Edition*.

zgmrpd manages the switch multicast table (MARL). Based on GMRP packets received on the target interface, *zgmrpd* creates, updates, or deletes an entry in the MARL. The key to each MARL entry is a source Ethernet multicast address combined with a VLAN ID. Two port bitmaps are maintained: one that identifies the untagged members of the VLAN, and one which identifies which ports of the VLAN have listening hosts attached.

NOTE: OpenArchitect does not tag BPDU packets. This means that if a port belongs to multiple VLANs, exactly which Multicast MAC/VLAN entry to create, modify, or delete cannot be determined. GMRP should not be used when ports belong to multiple VLANs. If the target is a zhp, it’s recommended that the zhp contain all the ports on the switch.

When the maximum number of entries in the MARL is reached, *zgmrpd* deletes a random entry prior to adding the next entry.

By default, while GMRP is running, unregistered multicast traffic is filtered on the target interface. The GMRP protocol provides a mechanism for changing the filtering behavior on a per-port basis. Filtering behavior can be set to filter unregistered multicast traffic, forward unregistered multicast traffic, or forward all traffic. *zconfig* can be used to set the filtering mode of each port. See *zconfig*. Ports connected to routers should be set to "Forward All" filtering mode manually with zconfig after zgmrpd is up and running. When *zgmrpd* is terminated, filtering behavior is set to forward unregistered multicast traffic on each port in the target interface.

Only the GARP normal registration mode is currently supported.

Multiple instances of *zgmrpd* may run concurrently provided the targets are unique. However, *zgmrpd* cannot run concurrently with *zsnoopd*. See *zsnoopd*.

OPTIONS

-d <level>	Sets the level of debugging output required by zgmrpd. The default level is zero (0). Setting the debug level higher produces more output. Five (5) is currently the maximum output level.
-f	Run zgmrpd in foreground. Default is to run it in background.
-h <hostname>	Connect to remote host <hostname>.
-p <ppa>	Start zgmrpd on switch <ppa>. Default is 0.
-t <target>	Enable GMRP on the set of ports specified by the target, either a zhp or bzhp interface. There is no default. A target must be specified.

EXAMPLES

```
In the following example, zgmrpd starts a background task that
enables the GMRP protocol for the ports in the zhp0 interface.
zgmrpd receives and sends GMRP packets, and updates the Multi-
cast Table (MARL) accordingly. This background task continues
throughout the life of the Layer 2 network, or until manually
terminated.

zgmrpd –t zhp0

Once you run zgmrpd, use zmarl to display the contents of the
MARL. zgmrpd deletes all entries in the MARL when starting up,
and when shutting down. Manual changes to the MARL can be made
while zgmrpd is running. However, all such changes will be de-
leted when zgmrpd is terminated.
```

SEE ALSO

zconfig, zsnoopd

zgr

NAME

zl2, zl2mc, zl3host, zl3net, zvlan –Formatted display of OpenArchitect generic tables.

zl2 displays the abstraction API layer 2 table.

zl2mc displays the abstraction API layer 2 multicast table.

zl3host displays the abstraction API layer 3 host route table.

zl3net displays the abstraction API layer 3 network route table.

zvlan displays the abstraction API VLAN table.

SYNOPSIS

```
zl2 [-p <ppa>] [-i <index>] [-m <mac_address>] [-a] [-v
   <vlan_id>] [-P port] [-h <host_name>] [-d <level>]

zl2mc [-p <ppa>] [-i <index>] [-m <mac_address>] [-a] [-v
   <vlan_id>] [-P port] [-h <host_name>] [-d <level>]

zl3host [-p <ppa>] [-i <index>] [-m <mac_address>] [-a] [-P
   port] [-h <host_name>] [-d <level>]

zl3net [-p <ppa>] [-i <index>] [-m <mac_address>] [-a] [-P
   port] [-h <host_name>] [-d <level>]

zvlan [-p <ppa>] [-i <index>] [-a] [-v <vlan_id>] [-h
   <host_name>] [-d <level>]
```

DESCRIPTION

The generic table display functions produce formatted output of the abstraction API tables for display on the user console. The format of the output is table-dependent.

Headers describing the column being displayed are printed after every 22 lines of output, which makes it easy to pipe through *more*. The abstraction layer tables grow and shrink as entries are added and deleted.

Several options are available which enable the user to display only selected entries. Additionally, there is an option that clears user-specified entries in the table.

OPTIONS

-p <ppa> Specifies the Physical Point of Attachment (PPA). Each switch that is controlled by the CPU and run by zreg is a unique PPA. If there is only one switch, such as when zreg is running on the embedded processor, the PPA is zero (0). The default PPA is 0.

-i <index> Displays the entry at the <index> position in the table. Valid for all tables. Cannot be combined with -m, -P or -v.

-m <mac_address> Displays entries whose MAC address field matches <mac_address>. Only valid for tables that have a MAC address field. Cannot be combined with -i, -P, or -v.

-a Displays the entire table.

-v <vlan_id> Displays entries whose VLAN ID field matches <vlan_id>. Only valid for tables that have a VLAN ID field. Cannot be combined with -i, -m, or -P.

-P <port> Displays the entries whose port field matches <port>. Only valid for tables that have a PORT ID field. Cannot be combined with -i, -m, or -v.

-h <host_name> Specifies which hostname to connect. By default, zgr connects to the locally connected OpenArchitect switch (i.e., the one that is on the local PCI bus).

-d <level> Sets the level of debugging output required by zgr. The default level is one (1). Setting the debug level higher produces more output. Four (4) is currently the maximum output level.

EXAMPLES

The following example searches for and displays an entry in the *zl2* table with the specified MAC address:

```
zl2 -m 00:c0:95:45:00:00
```

If there is an entry in the ZL2 table with the MAC address, 00:c0:95:45:00:00, all the fields of that entry will be displayed.

The following command deletes the above entry:

```
zl2 -c -m 00:c0:95:45:00:00
```

The following command displays all entries of the *zl2* table:

```
zl2
```

Be careful, the *-c* option does not ask. The following command deletes all entries in the *zl2* table:

```
zl2 -c
```

SEE ALSO

zal

zgvrpd

NAME

zgvrpd - GARP VLAN Registration Protocol (GVRP) daemon for the OpenArchitect switch.

SYNOPSIS

```
zgvrpd [-d <level>] [-f] [-h <hostname>] [-p <ppa>] [-t <tar-
   get>]
```

DESCRIPTION

zgvrpd is run after the network interfaces are created and initialized with *zconfig*, and started with *ifconfig*. *zgvrpd* starts a background task that implements the GARP VLAN Registration Protocol (GVRP) protocol for a specified *zhp* interface. GVRP provides a Layer-2 mechanism for dynamically managing port membership in VLANs, including adding and deleting ports, and creating and deleting VLANs. The background task started by *zgvrpd* continues throughout the life of the Layer 2 network. GVRP is specified in *ANSI/IEEE Std 802.1Q, 1998 Edition*. The GARP protocol on which GVRP is based is specified in *ANSI/IEEE Std 802.1D, 1998 Edition.*

zgvrpd updates the switch's VLAN configuration, based on GVRP packets received on the target interface. Specifically, it adds a port to or deletes a port from the VLAN specified in the GVRP packet. If the VLAN does not exist, *zgvrpd* creates it. If *zgvrpd* deletes the last port from a dynamically-created VLAN, it also deletes the VLAN.

When a VLAN is dynamically created, a corresponding zhpN interface is also created, where N is an integer between 5000 and 9999. The value of N is equal to 5000 plus the VLAN identification number (VID). For example, if zgvrpd creates vlan 5, it also creates a zhp5005.

zgvrpd learns the existing (static) VLAN configuration of its target when starting up. When shutting down, *zgvrpd* deletes only the dynamic changes to the VLAN configuration that it has made. Manual changes to the VLAN configuration can be made while *zgvrpd* is running. However, all such changes will be deleted when *zgvrpd* is terminated.

Only the GARP normal registration mode is currently supported. Multiple instances of *zgvrpd* may run concurrently provided the targets are unique. If zgvprd's target is a zhp, it's recommended that the zhp contain all the ports on the switch.

zgvrpd cannot be run concurrently with zsnoopd, because zsnoopd assumes static VLAN membership.

OPTIONS

-d <level>	Sets the level of debugging output required by zgvrpd. The default level is zero (0). Setting the debug level higher produces more output. Five (5) is currently the maximum output level.
-f	Run zgvrpd in foreground. Default is to run it in background.
-h <hostname>	Connect to remote host <hostname>.
-p <ppa>	Start zgvrpd on switch <ppa>. Default is 0.
-t <target>	Enable GVRP on the set of ports specified by the target zhp interface. There is no default. A target must be specified.

EXAMPLES

In the following example, zgvrpd starts a background task that enables the GVRP protocol for the ports in the zhp0 interface. zgvrpd receives and sends GVRP packets, and updates the VLAN configuration accordingly. This background task continues throughout the life of the Layer 2 network, or until manually terminated.

```
zgvrpd –t zhp0
```

Once you run *zgvrpd*, use *zconfig -a* to display the current VLAN configuration.

SEE ALSO

zconfig, *zsnoopd*

zl2d

NAME

zl2d - Layer 2 daemon for the OpenArchitect switch.

SYNOPSIS

```
zl2d [start | stop] [-t <msecs>] [-d <level>] [-f]
        [-p <priority>] <interface..>
```

DESCRIPTION

zl2d is run after the network interfaces are created and initialized with zcon-

fig. zl2d creates a Linux bridge for each interface using brctl. The bridge name is the interface name with a ‘b’ pre-pended to it. This command is primarily used for Spanning Tree Protocol (STP). Each port associated with the interface is included within the bridge. zl2d starts a background task that continues throughout the life of the Layer 2 network.

OPTIONS

start | stop	Starts or stops the zl2d daemon.
-t <msec>	Cause zl2d to monitor the Spanning Tree state of each port on each bridge every <msec> milliseconds. If unspecified, the default is 500 milliseconds.
-f	Enables Fast Forward on bridge(s) using 0x4000 (16384) as the dynamic root priority.
-d <level>	Sets the level of debugging output required by zl2d. The default level is one (1). Setting the debug level higher produces more output. Four (4) is currently the maximum output level.
-p <priority>	Sets the dynamic root priority. <priority> should be specified as a decimal number. A priority of 0 disables root priority change.
<interface...>	The network interfaces on which zl2d should operate. These network interfaces must first be created by zconfig. Zl2d does not operate with standard network interface cards. It only works on switch network interfaces created by zconfig.

OPERATIONS

zl2d manages the Spanning Tree state fields in the switch of each port within the bridge(s). Based on a timer, zl2d reads the port information for each Linux bridge and updates the switch when necessary.

EXAMPLES

In the following example, zl2d creates a Linux bridge named `bzhp0` which includes all of the `zre<n>` devices previously associated with the `zhp0` device. zl2d then starts a background task that monitors the port information of the Linux bridge every 500 ms. and updates the Spanning Tree state fields in the hardware when necessary.

```
zl2d -t 500  zhp0
```

Once you run zl2d, use brctl to display and alter your Spanning Tree settings.

SEE ALSO

zconfig, brctl

zl3d

NAME

zl3d - Layer 3 daemon for the OpenArchitect switch.

SYNOPSIS

```
zl3d [-h <host_name>] [-t <msecs>] [-b] [-e] [-i I|D] [-c] [-
    l] [-n] [-d <level>] <interface ..>
```

DESCRIPTION

zl3d is run after the network interfaces are created and initialized with zconfig, and started with ifconfig. Zl3d listens for Netlink messages from the kernel and monitors the Linux network routing tables for routing updates. When an update occurs, if appropriate, zl3d updates the corresponding routing tables in the silicon so that Layer3 forwarding is done in the switch at line speed. Zl3d also ages the switch host route entries.

zl3d attempts to keep the Linux route cache and the switch host route table consistent. When a host route table entry is in use, zl3d updates the Linux route cache. Host route table entries are deleted when Linux removes the corresponding entry from the route cache. Similarly, network route entries are removed when the corresponding Linux FIB table entry is deleted.

OPTIONS

-h <hostname>	Specifies which host to monitor. By default, zl3d monitors the OpenArchitect switch that is locally connected (i.e., the one that is on the local PCI bus).
-t <msec>	Sets the timeout value. By default, zl3d wakes up every 15 seconds (15000 ms) to look for updates to the Linux routing tables and to do aging processing of route table entries on the switch.
-b	Do not background the zl3d process.
-e	Enables additions of a default route to the L3 network route table.
-i I | D	Turn off L2 insert(I)/delete(D) notification processing
-l	Leave hardware tables intact on exit.
-n	Do not write the proc table entry. By default, zl3d writes 3/2 of it's timeout value to /proc/net/sys/ipv4/route/ route_expires. The Linux kernel uses this value as an expiration time for cache routes updated by zl3d.

-d <level>	Sets the level of debugging output required by zl3d. The default level is one (1). Setting the debug level higher produces more output. Four (4) is currently the maximum output level.
<interface...>	The network interfaces on which zl3d should operate. These network interfaces must first be created by zconfig. Zl3d does not operate with standard network interface cards. It only works on switch network interfaces created by zconfig. It uses the same syntax as zconfig.

OPERATIONS

Zl3d manages the host route and network route tables located in the switch. Based on a trigger condition, *zl3d* reads the Linux FIB table and the Linux route cache. Each table entry is filtered by *zl3d* according to the following:

- If an entry from the route cache is not a broadcast address or a local address, and *zl3d* is able to resolve the MAC address of the destination, then *zl3d* inserts the entry into the switch host route table.
- If an entry in the Linux FIB table is a host entry and *zl3d* is able to resolve the MAC address of the destination host, then the entry is inserted into the switch host route table. If an entry from the FIB table is a gateway entry, is not local, and *zl3d* is able to resolve the MAC address of the gateway, then the entry is inserted into the switch network route table.

EXAMPLES

Normally, *zl3d* is run as a background task that continues throughout the life of the Layer 3 network.

In the following example, for the three interfaces specified, *zl3d* continuously monitors the Linux FIB and route cache tables looking for updates every three seconds. Entries in the switch host route table are checked for activity every 15 seconds and aged accordingly.

```
zl3d zhp1 zhp2 zhp3
```

SEE ALSO

zconfig

zlc

NAME

zlc – link and LED control

SYNOPSIS

zlc [-p <ppa>][-h <hostname>][-d <level>][-x] <port_list> <action> [on | off]

zlc [-p <ppa>][-h <hostname>][-d <level>][-x] <action> [on | off |clear]

zlc [-p <ppa>][-h <hostname>][-d <level>][-x] [state|query]

DESCRIPTION

The *zlc* application sets the link speed and state of individual ports of the switch, or displays the current state. It can also turn on or off the extract LED or the internal fault LED.

OPTIONS

-p <ppa>
: Specify ppa to connect to switch (default 0).

-h <hostname>
: Connect to remote host <hostname>.

-d <level>
: set debug level to <level>

-x
: expected query value. Creates no output, exit code only. If the port_list contains more than one port, returns the number of ports that match the option.

<port_list>
: port or list of ports on which to take action. Port lists are supplied in zconfig syntax (e.g., zre1, zre2..4, etc.)

<action>
: Set link speed or state to `up`, `down`, `auto`, `100fd`, `100hd`, `10fd` or `10hd`. The interface must be down to change the port speed. Set `intfault` or `extfault`. Must supply <option>.
Use `query` to return present settings.

on | off
: Turn `on` or `off` the specified LED. Only valid for actions `intfault`, `extfault` or `extract.`

clear
: No longer globally control the specified LED

state
: Return the list of LEDs currently illuminated.

query
: Return the list of LEDs currently controlled globally.

EXAMPLES

In the following example, *zlc* forces the line speed of port 1 to 100 Full duplex. The interface must be down to change the speed. Assuming zre1 is part of interface zhp0,

```
ifconfig zhp0 down
zlc zre1 100fd
```

The external fault, internal fault, and ok LEDs can be set on a per port

basis or *globally.* To set the external fault LED for a particular port,

```
zlc zre1 extfault on
```

To query the settings of a particular port,

```
zlc zre1 query
```

Global Settings

The external fault, internal fault, and extract LEDs are set as a logical OR of all the ports. The LEDs can also be set globally to on, off, or other. If globally set to on or off, the LED will not change when links go up or down ,or interfaces are configured. If set to other, the LED resumes its normal operation.

The next example globally turns on the Pull (extract) LED.

```
zlc extract on
```

Additional capabilities are also available by supplying an additional led action:

```
zlc <led_name> on
```

The <led_name> LED is turned on. The LED will not change when links go up or down or interfaces are configured. If other is used, the LED resumes its normal operation. <led_name> can be intfault, extfault, extract, or ok.

```
zlc query
zlc state
```

"query" lists the LEDs which have been set globally "on" or "off." "state" shows which LEDs are on at the moment. All LEDs are shown, including the clk LED.

SEE ALSO

ifconfig

zlmd

NAME

zlmd – monitor link changes or hot swap events.

SYNOPSIS

```
zlmd [-h <hostname>] [-b] [-d <level>] [-p <ppa>]
   [{-f <configuration_file>} | <configuration>]
```

DESCRIPTION

The *zlmd* application is intended to run as a daemon, waiting for a configured event to occur and then running the program configured for that event. The events monitored are changes in the link status of any of the 16 ports of the switch, the start of removal of the switch from the back plane, or the cancellation of the removal before it actually takes place. The program can be a shell script that initiates appropriate actions to respond to the event.

Configuration commands:

<port-list> = <program>
Run <program> when a fault occurs or clears on a port.

hot swap = <program>
Run <program> when a hot-swap extraction or insertion event occurs.

<port-list> is a list of ports in the same forms supported by zconfig, e.g. zre1,zre2 or zre10..14

<program> Path to an executable program or script to be run when the event occurs. Note: An absolute path to <program> is required. The program will be called with the following parameters:

For Link Changes:

<program> <ppa> <port> {external(0)|internal(1)} {off(0)|on(1)}

For Hot-swap Events:

<program> <ppa> {extraction(1)|insertion(2)}

OPTIONS

-h <host_name> Connect to remote host <host_name>

-b Do not background the zlmd process.

-d <level> Set debug level to <level>.

-p <ppa> Monitor switch <ppa>. Default is 0.

{-f <configuration_file>} | <configuration>

Read configuration from <file>. If file is a '+', configuration is read from stdin. Without the -f option, a single line of input can precede the last flag. (i.e., <Configuration>)

EXAMPLES

In the following example, *zlmd* monitors ports 1 through 4 and runs a

script called *prt_change* upon a link change event.

```
zlmd zre1..4=/usr/sbin/prt_change
```

```
Suppose port 2 were up and you disconnected the cable, zlmd
would call prt_change with the following parameters:
```

```
/usr/sbin/prt_change 0 2 0 1
```

where 0 is the ppa, 2 is the port, 0 is an external fault, 1 is on.

SEE ALSO

zconfig

zlogrotate

NAME

zlogrotate – Rotates log files.

SYNOPSIS

```
zlogrotate [-b] [-t time] [-s segment size] [-n # of files] [-
   f file to rotate]
```

DESCRIPTION

```
zlogrotate rotates the selected file every [time] seconds if
the file is larger than [segment size]. It will keep only the
number of files selected.
```
zlogrotate is called from /etc/init.d/rcS by default with no parameters.

OPTIONS

-b	Do not background the process - i.e. run in foreground.
-t <time>	the time between logfile checks in seconds (default 60)
-s <size>	the targeted file segment size, in kilobytes (default 256)
-n <# of files>	The number of segments kept on the system (default 4)
-f <file>	The file to rotate (default /var/log/messages)

EXAMPLES

To start zlogrotate with the default values,

```
zlogrotate
```

zmirror_4800

NAME

zmirror_4800 - Set packet mirroring on an ingress or egress port

SYNOPSIS

```
zmirror_4800  -a | -t
zmirror_4800  [-e] [-l] <from> <to>
```

DESCRIPTION

zmirror_4800 sets packet mirroring from a given set of ports to a given port. Turning on packet mirroring causes a copy of the packet to be sent to the *to* port. There can be only a single *to* port, but there can be multiple *from* ports. The *zmirror_4800* command overwrites the *to* port and accumulates the *from* ports. Note: there are performance issues when trying to mirror more bandwidth than is available on the *to* port.

After executing the following three commands, packets received on ports 0, 1 and 2 would be mirrored (copied and transmitted) to port 12. This mirroring would be in addition to any Layer 3 or Layer 2 switching.

```
zmirror_4800 0 12
zmirror_4800 1 12
zmirror_4800 2 12
```

To clear the current mirroring, use the *-t* option. The *-e* option can be used to indicate that packets being sent on a given port should be copied to the *to* port. For example if the -e option is used as follows, the packets transmitted, as opposed to received, on ports 0, 1 or 2 would be mirrored to port 12.

```
zmirror_4800 -e 0 12
zmirror_4800 -e 1 12
zmirror_4800 -e 2 12
```

The *-l* option toggles enabling of Layer 3 Packet mirroring. Layer 3 packet mirroring is off by default, and is enabled on a switch-wide basis. To mirror to the host port, use port 27.

OPTIONS

-a	Display the current mirroring setup
-e	Set egress port mirroring for the specified from port
-t	Teardown or disable the mirroring

-l	Toggle on and off Layer 3 packet mirroring

SEE ALSO

tcpdump

zmnt

NAME

zmnt – Expands the read/write files onto the RAM disk.

SYNOPSIS

```
zmnt [-c] <directory>
zmnt [-c] -t <file>
zmnt [-c] –l
```

DESCRIPTION

zmnt expands files from flash onto the RAM disk that have been previously saved with *zsync*. The *init* process runs *zmnt* to expand the files in flash onto RAM file system. The user may use *zmnt* to expand the files at anytime and may place them in any directory. For example:

```
zmnt /tmp
```

will expand the files under the directory /tmp.

Booting the kernel with `-i` instructs the init process to skip the *zmnt* stage After booting in this way, *zmnt* can be used for correcting a problem file in the read-write file system.

The *-t* option can be used to save the configuration of a switch to a tar file. A tar file can be copied to another switch and saved with *zsync -t*. As a result, the configuration of a switch may be cloned to other switches.

The *-c* option is used to mount the custom overlay. See *zsync* for a description of custom verses dynamic overlay.

OPTIONS

-c	Read saved files from the custom overlay
-t <file>	Save the overlay into the specified <file> in tar format.
<directory>	The directory under which the overlay files are expanded, or the file to which the tar image is saved.
-l	List files in selected overlay, do not unpack.

EXAMPLES

In the following example, *zmnt* the current overlay into a tar file called `overlay.tar`

```
zmnt –t overlay.tar
```

The resulting tar file can now be saved on a different host as a snapshot of the overlay at that point in time. Use zsync to restore the overlay on the switch:

```
zsync –t overlay.tar
```

The restored overlay will be used upon the next reboot.

SEE ALSO

zsync

zrc

NAME

zrc - Packet rate control

SYNOPSIS

```
zrc  -a | -t | [-b] [-m] [-d] [-p <port>] [-v <vlan_id>] [-g
   <group>] [-M <address>] [-T <msecs>] [-D <level>]
   <rate_limit>
```

DESCRIPTION

zrc sets rate control on Broadcast, Multicast and/or Destination Lookup Failure (DLF) packets. The rate is measured in the number of packets per time period. If the number of packets received of the specified type exceeds the specified rate limit, packets are discarded at the ingress port.

OPTIONS

-a	Display the current rate control settings
-t	Teardown or disable all rate control
-b	Enable rate control for Broadcast packets
-m	Enable rate control for Multicast packets
-d	Enable rate control for DLF packets
-p	Enable rate control on this <port>

-v	Enable rate control for this <vlan_id>
-g	Enable rate control for this Multicast <group>
-M	Enable rate control for this Mac <address>
-T	Set time period to <msecs>. Default is 1000 (one second).
-D <level>	Set debugging output to <level> when running the program.
<rate_limit>	The number of packets per time period above which Broadcasts, Multicasts and/or DLFs will be discarded. Valid rate limits are any number between 0 and 1048560 inclusive.

SEE ALSO

zstats

zreg

NAME

zreg - Read and write registers and tables on the switch.

SYNOPSIS

```
zreg [-h <hostname>] [-d <level>] [-p <ppa>] [-i <index>]
   [-t <index>] [-k] [-w] [-r 10] <reg>
```

DESCRIPTION

zreg allows a user to read and write direct and indirect registers and tables on the switch chip resident on the Intel OpenArchitect switch product. *zreg* is commonly used for debug, or for prototyping when creating applications that control the OpenArchitect switch. It is also useful when put into shell scripts for displaying hardware status or statistics. Although the -t option allows tables to be displayed, one might find the formatted output of the table functions more useful. See *zal*.

OPTIONS

-p <ppa>	Specify the Physical Point of Attachment (PPA). Each OpenArchitect switch that is controlled by the CPU on which zreg is running is a unique PPA. If there is only one OpenArchitect switch, as would be the case when zreg is running on the embedded processor the PPA would be 0. The default PPA is 0.
-w	Causes zreg to write to the register or table. Data to be written is read from standard input.
-i <index>	Causes zreg to access at the indexed register specified by



<reg>. See OPERANDS for usage of <reg> with indexed registers. The <index> parameter is used to determine which entry.

-t <index> — Causes zreg to access the memory specified by <reg>. The <index> parameter is used as the index into the specified table. Only content addressable memories are accessed using the –t option. All other tables and memories are accessed with the –i option.

-k — Causes zreg to access the memory specified by <reg>. The entry accessed is determined using the data from standard input as the search key. Enter 0 for fields that are not part of the search key.

-h <hostname> — Specify the <hostname> to configure. By default zreg configures the OpenArchitect switch that is locally connected (i.e., the one that is on the local PCI bus).

-r 10 — Sets numeric radix for registers to 10. Default is 16.

-d <level> — Set the level of debugging output produced by zreg. The default level is 1. Setting the debug level higher produces more output. The maximum level of output is currently 4.

OPERANDS

<reg> — If no –i, -t, or –k option is specified, <reg> is chosen from the enum ix54_reg_e. With the –i option, <reg> is chosen from the enum ix54_xreg_e. With the –t and –k options, <reg> is chosen from ix54_mem_e.

EXAMPLES

There are three types of accesses performed by *zreg*; scaler register, indexed register, and table. For each of theses access types, values can be read or written. Content addressable memory register access is the default. The following is an example of reading the CONFIG Register:

```
zreg 1
```

When running *zreg* on the embedded processor of the OpenArchitect switch, the <ppa> is always 0, since the embedded CPU processor only controls the directly attached switch chip. Since the default <ppa> is 0 the *-p* option is not needed.

To write a value to a register the *-w* flag is used and the data is read from standard input. The following example writes 0x800000640 to the Aging Time Register:

```
echo 0x80000640 | zreg -w 49
```

If the *zreg* command is typed at the prompt, it waits for input from the user. You may also use File I/O or shell scripts. The following example

reads the MAC Packet Length Register for port 7:

```
zreg -i 7 2
```

SEE ALSO

zal, zstats

zsnoopd

NAME

zsnoopd - IGMP Snooping daemon for the OpenArchitect switch.

SYNOPSIS

```
zsnoopd [-d <level>] [-f] [-h <hostname>] [-p <ppa>]
   [-r <sec>] [-t <sec>] [-u <sec>] [-v <vlan_id>]
```

DESCRIPTION

zsnoopd is run after the network interfaces are created and initialized with *zconfig*, and started with *ifconfig*. *zsnoopd* starts a background task that monitors incoming IGMP traffic in order to learn which hosts in a VLAN are listening to which IP multicast addresses. *zsnoopd* updates the multicast table in silicon with this information, so that IP multicast traffic is forwarded only to the ports in a VLAN to which listening hosts are attached. This optimizes packet flow through the switch. Traffic on all VLANs is monitored by default. IGMP snooping can be restricted to specific VLANs using the *–v* option. The background task started by *zsnoopd* continues throughout the life of the Layer 2 network.

zsnoopd manages the switch multicast table (MARL). Based on monitored IGMP traffic, *zsnoopd* creates or updates an entry in the MARL. The key to each MARL entry is a source Ethernet multicast address combined with a VLAN ID. Two port bitmaps are maintained: one that indentifies the untagged members of the VLAN, and one which identifies which ports of the VLAN have listening hosts attached.

When the maximum number of entries in the MARL is reached, *zsnoopd* deletes a random entry prior to adding the next entry.

Traffic from IP multicast addresses not found in the multicast table is forwarded to all ports within a VLAN. Traffic from reserved IP multicast addresses (224.0.0.X) is forwarded on all ports within a VLAN.

To operate correctly, traffic from unregistered IP multicast addresses should be forwarded on all ports in a VLAN. To do this, the multicast port filtering mode must be set to FORWARD_UNREGISTERED. See *zconfig*.

```
Neither zgmrpd nor zgvrpd can run concurrently with zsnoopd.
```

OPTIONS

-d <level>	Sets the level of debugging output required by zsnoopd. The default level is zero (0). Setting the debug level higher produces more output. Four (4) is currently the maximum output level.
-f	Run zsnoopd in foreground. Default is to run it in background.
-h <hostname>	Connect to remote host <hostname>.
-p <ppa>	Start zsnoopd on switch <ppa>. Default is 0.
-r <sec>	Time to wait, in seconds, before removing a port with no router multicast traffic. Default is 260 seconds.
-t <sec>	Time to wait, in seconds, before removing a port with no host multicast traffic. Default is 260 seconds.
-u <sec>	Time to wait, in seconds, before checking port timeouts.
-v <vlan_id>	Enable zsnoopd for VLAN <vlan_id>. Default is to enable zsnoopd on all VLANs. This option may be entered more than once.

EXAMPLES

In the following example, *zsnoopd* starts a background task that monitors incoming IGMP packets and updates the Multicast Table (MARL) accordingly. This background task continues throughout the life of the Layer 2 network.

```
zsnoopd
```

Once you run *zsnoopd*, use *zmarl* to display the contents of the MARL. *zsnoopd* deletes all entries in the MARL when starting up, and when shutting down. Manual changes to the MARL are not recommended while *zsnoopd* is running.

SEE ALSO

```
zconfig, zgmrpd
```

zstats

NAME

zstats – Display statistics and information about switch

SYNOPSIS

```
zstats  [-d <level>] [-i <unit>]|[-m <port>]|
   [-v <vlan id>]|[-V]|[-t <tgid>]
```

DESCRIPTION

zstats displays MIB counters for a selected physical port, trunk group or VLAN. It can also display information about the configuration of the switch and bridge to the PCI bus or the Vital Product Data memory. All output is formatted.

OPTIONS

-m <port>	MIB statistics for specified <port>
-v <vlan id>	MIB statistics for specified <vlan id>
-i <unit>	Information for specified <unit>, 0 is switch, 1 is bridge
-d <level>	Set debug level to <level>
-p <ppa>	Specify <ppa> of switch (default 0).
-t <tgid>	MAC layer statistics for all ports in trunk <tgid>.
-V	Vital Product Data (not currently supported in ZT 8102)

EXAMPLES

To display statistics for a particular port on the switch, such as port 0

```
zstats –m 0
```

SEE ALSO

zreg, zal

zspconfig

NAME

zspconfig - configure and start surviving partner

SYNOPSIS

```
zspconfig [-d <level>] [-p <path>] [-u <dhcp_interface>] [-c
   <dhclient.conf>] [-t <timeout>] [-s] [-v] -f <file>
```

DESCRIPTION

zspconfig is part of the High Availability feature set, which is a separate product from the standard OA. *zspconfig* is used to configure and start the Surviving Partner software. With the *-f* option a configuration file is provided that completely describes the network setup and desired behavior of all of the switches participating in the Surviving Partner. With the *-u* option the interface on which to run *dhclient* to retrieve a configuration file must be provided. The configuration file format retrieved by a *-u* is identical to that supplied with *-f*. It is envisioned that the *-f* option is used for initial configuration, and all subordinate and replacement switches run *zspconfig* with the *-u* option. The *-v* option prints the current version of *zspconfig* and performs no actions.

OPTIONS

-d <level> Set the debug level. The default debug level is 1. The higher the level, the more debugging output is produced. Debugging output is sent to the console.

-p <path> Set the directory path for where zspconfig places the scripts it generates. The default location is `/etc/rcZ.d/surviving_partner`.

-u < dhcp_interface>
Come up as an unconfigured slave. Use the specified <dhcp_interface> to retrieve the configuration. User confirmation is not required unless the –s option is also used.

-c <dhclient.conf>
Use file <dhclient.conf> as the configuration file to dhclient when retrieving configuration information. If –c is not used, a default configuration file is created and used. Only valid with the –u option.

-t <timeout> Time to wait in seconds before giving up on finding a Surviving Partner to retrieve configuration information from. Only valid with the –u option.

-s Do not ask for confirmation. Run from a script.

-v Prints the current version of zspconfig.

-f <file>
The provided <file> is used as input to configure the Surviving Partner. See the next section on CONFIGURATION FILE for the syntax of the configuration file.

CONFIGURATION FILE

```
The configuration file contains commands for controlling the
Surviving Partner setup. Commands are single lines delimited
ending with a semicolon. Comments, spaces and new lines are ig-
nored. Comments begin with the # character and include charac-
```

ters through the next new line. Comments may be placed on the same line as a command after the semicolon.

All configurations must include the VLAN configuration first by use of zconfig commands. zconfig commands can be put in the configuration file and will be passed directly to the zconfig application. zconfig commands start with the keyword zconfig and are of the same format as described in the zconfig manual page. Here is an example of zconfig commands in a zspconfig configuration file.

```
zconfig zhp0: vlan1 = zre1..4;
zconfig zhp1: vlan2 = zre5..8;
zconfig zhp2: vlan100 = zre14;
```

In the above example, three VLANs are created: *zhp0* and *zhp1* will be used as connections to high availability nodes; *zhp2* will be used as the inter-connect between two Surviving Partner switches to run the VRRP heartbeat. All VLANs must be created before other *zspconfig* commands may operate on them.

The next section of a zspconfig configuration file sets up the IP addresses of the created VLANs. There are two types of addresses in a Surviving Partner setup: Physical Addresses, and Virtual Addresses. The Virtual Addresses are those that VRRP manages and moves to the current Master switch. The Physical Addresses are the real addresses of the switch, and are used for management only. Physical Addresses are setup using the `sibling_addresses` command. Virtual addresses are setup using the `virtual_address` command. The sibling addresses name comes from the fact that we are setting up the addresses for all of the sibings in the Surviving Partner group. So if we have two Surviving Partner switches, the `sibling_addresses` statement might look like this:

```
sibling_addresses: zhp0 = 10.0.0.30, 10.0.0.31;
sibling_addresses: zhp1 = 11.0.0.30, 11.0.0.31;
sibling_addresses: zhp100 = 100.0.0.30, 100.0.0.31;
```

A `sibling_addresses` statement is required for each VLAN created with the *zconfig* commands. The two addresses in the list indicate there are two switches in the Surviving Partner group. The first address 10.0.0.30 and 11.0.0.30 are assigned to the switch on which the configuration is being run. The remaining addresses are distributed to the switches that run ***zspconfig -u*** on a first come, first serve basis.

The `sibling_addresses` command may also take a *netmask* operand similar to that given to *ifconfig*. For example:

```
sibling_addresses: zhp0 = 10.0.0.30, 10.0.0.31 netmask
    255.255.255.0;
sibling_addresses: zhp1 = 11.0.0.30, 11.0.0.31 netmask
    255.255.255.0;
```

Virtual addresses should be setup for all VLANs that connect to *High Availability* nodes. Usually this would include all VLANs except the interconnect VLAN or VLANs connected to upstream routers. It is possible that a setup would have VLANs that are used for management only. Virtual addresses are setup as follows:

```
virtual_address: zhp0 = 10.0.0.42 netmask 255.255.255.0;
virtual_address: zhp1 = 11.0.0.42 netmask 255.255.255.0;
```

Only a single address per VLAN is provided because this single address will move with the current Master switch, and the *netmask* must be the same as that provided in the **`sibling_addresses`** statement.

The last required section for the configuration is description of the ports. Particularly we need to specify one of the following for all of the ports participating in the high availability setup. The possible port types are:

interconnect	Ports connected between groups of Surviving Partner switches. VRRP heartbeat messages are sent on the interconnect ports.
crossconnect	Crossconnect ports are ports that are connected to other Surviving Partner switches, that are not part of this Surviving Partner group. Crossconnect ports behave differently then Intel ANS ports. The links are not brought down temporarily, and VRRP runs with the native MAC addresses to avoid MAC address duplication with the other VRRP group.
Intel ANS	Ports connected to Intel ANS enabled nodes. These ports contain virtual addresses managed by VRRP. And during a failover event, the links are toggled down to force Intel ANS to failover to the Master switch.
route	Ports connected to upstream routers. VRRP does not manage virtual IP addresses for these links. Routing protocols must be used to instruct up stream routers of a different path to get to the VRRP managed networks. These port types are currently not implemented.
monitor_only	Ports that are monitored but do not have a virtual address managed on them. They will not have their links brought down temporarily during a failover scenario. These ports are only monitored. If a problem occurs on this type of link it will cause a failover scenario.
configure_only	Ports are configured as per the zconfig commands, but do not participate in the high availability network. Problems on these links will not cause a switch failover.

Each port that is setup in a VLAN by the *zconfig* commands must have its port type specified. The port type is specified on a physical port bases. That is on a zre basis, but zhp names can be used as a quick way to setup the port type for all ports that are a member of that VLAN. It is possible to make a port a member of more then one VLAN. That is

a *zre* can be a member of more then one *zhp*. In such cases, configuring the *zhp*'s as different port types would cause a conflict, and will not work. To handle this setup the individual *zre* commands would be used to setup the port types. Here is an example of setting up the port types as a continuation of our current configuration:

```
interconnect: zhp2;# Could also use zre14

Intel ANS:    zhp0, zhp1;# Could also use zhp0..1,
               # or list the zre's
```

The ".." wild card is supported as in *zconfig* to indicate a range of numbers. The comma is used to indicate a list. The *zre's* that are part of the *zhp* could also be used. Here is a more complex setup. The zconfig commands are also shown to understand the VLAN setup:

```
zconfig zhp0: vlan1 = zre1..4, zre14;
zconfig zhp1: vlan2 = zre5..8, zre14;
zconfig zhp100: vlan100 = zre14;
zconfig zhp0: vlan1 = zre1..4;
zconfig zhp1: vlan2 = zre5..8;
zconfig zhp2: vlan100 = zre14;

#  sibling and virtual address setup omitted
interconnect:zhp100;
Intel ANS:zre1..8;# Use zre definition to
 # exclude zre14
```

If *zhp0* and *zhp100* are setup as different port types, there would be a conflict for port *zre14*. In the particular example above, the *zre14* is shared. It is used to pass VRRP interconnect traffic and as a means to pass VLAN 1 and 2 traffic between switches. Since *zre14* is an interconnect, it is not a Intel ANS enabled port, and therefore should be setup as an interconnect port type. To accomplish this, the zre ports are listed to avoid conflicting port types.

Note that a single line cannot contain both *zhp* and *zre* definitions. Therefore

```
Intel ANS:zhp1, zre1..4
```

does not work and the definition *zre1..8* **is equivalent.**

Optional *zspconfig* commands are listed below.

```
vrrp_msg_rate:  1000;# Time in milliseconds.
 # Default is 1 second
```

The message rate is the interval between VRRP messages sent over the interconnect link. The time given is in milliseconds. It takes the lack of 3 VRRP messages for the Backup to assume the role as Master. It is recommended with faster message rates to increase the default priority to 254. The higher priority will decrease the latency of the failover.

```
vrrp_def_priority:  254;# default is 100.
 Value from 1 to 254
```

The default virtual MAC address for VRRP is 00:00:5E:00:01:<vid>, where <vid> the virtual router ID. Using this virtual address is problematic in network environments where multiple VRRP instances might be running that are using the same <vid>. To overcome this problem, *zspconfig*, uses a default MAC address derived from the physical address of the switch on which it is running. For the slave switch, the **vrrp_virtual_mac_addr** command is used to set the MAC address to the same as the Master. This statement is typically not used within the Master switch's configuration. It is used in the *zspconfig* generated Slave switch configuration. And is retrieved by the Slave switch with *zspconfig* using the *–u* option. If in doubt, don't use this command.

```
vrrp_virtual_mac_addr: 00:01:02:03:04:05
```

There are three failover_mode's supported; *switch*, *vlan* and *port*. For *switch* failover, if any managed link fails, the entire switch is failed over to the backup switch. In *vlan* failover, if a link fails in a VLAN, only the links associated with that VLAN are failed over. In *port* failover, only the port that fails is moved to the backup switch. For both *vlan* and *port* failover, the interconnect link will need to be used to maintain connectivity between ports that have failed over and those that have not. For *vlan* and *port* failover modes, the interconnect link must be an inband port, and must be included in the managed VLANs running with VLAN tagging on.

```
failover_mode:  switch;
failover_mode:  vlan;
failover_mode:  port;
```

Additional startup scripts may be included in the configuration using the **start_script** command. The files in the **start_script** command will be placed in a location for *tftp* transfer to sibling switches that initialize using the *–u* option. A common use of the **start_script** command might be to propagate *gated* configurations to all members of the Surviving Partner group. Absolute path names must be used. Using multiple commands allows inclusion of multiple scripts. For example:

```
start_script: /etc/rcZ.d/S75gated;
start_script: /etc/rcZ.d/S80static_routes;
```

The **vrrpd_script** command allows a user defined script to be run when *vrrpd* changes state. This script is called at the end of the *zspconfig* created **vrrpd.script**. See *vrrpd –S* for a description of when the scripts are called. *zspconfig* sets up *vrrpd* to call **vrrpd.script**. The **vrrpd_script** command in *zspconfig* places a call to the user-defined script at the end of **vrrpd.script** file. The following example would call the **my_vrrpd_script** each time *vrrpd* calls its *–S* provided script:

```
vrrpd_script:  /etc/rcZ.d/surviving_partner/my_vrrpd_script;
```

NOTE: the **my_vrrpd_script** is not called from a different process thread. Therefore if **my_vrrpd_script** crashes or has long delays, it will crash the *vrrpd*, or cause delays in the Surviving Partner failover. To protect against this, write the script to launch a second script in a background shell. The advantage to calling the user provided script in the same process thread is that it gives synchronized control over the failover process for those who want it.

OUTPUT FILES

The output of *zspconfig* is a set of configuration and script files. The configuration files configure *vrrpd* and *zlmd* daemons. The *vrrpd* and *zlmd* daemons combined with the script files run the Surviving Partner. This is a list of all configuration and script files:

/etc/rcZ.d/S70Surviving_partner
: The main startup script that starts the Surviving Partner by running zconfig, ifconfig, zlmd and vrrpd. zspconfig prompts the user to run this script. This file can be saved with zsync to automatically start the Surviving Partner at switch boot.

/etc/rcZ.d/surviving_partner/vrrpd.conf
: Configuration script for the VRRP daemon. This configuration is used when the S70Surviving_partner script launches vrrpd. There is a line in this file for each router address vrrpd will manage. Or stated another way, each virtual_address command in the zspconfig configuration file results in a line in `vrrpd.conf`.

/tftpboot/zsp.conf<n>
: zspconfig configuration file that contains the configuration of the sibling backup switches. The <n> is used to distinguish potentially more than one backup switch. This configuration file is placed in /tftpboot, and is retrieved via DHCP by a replacement switch on boot up.

/etc/rcZ.d/surviving_partner/dhcpd.conf
: Configuration script used by dhcpd when the switch becomes master. dhcpd is used to serve replacement switches their configuration scripts. Namely a zsp.conf file that can be input to the zspconfig with the -u flag.

/etc/rcZ.d/surviving_partner/dhclient.conf
: If zspconfig is executed with the -u flag, a dhclient.conf file is created, and then dhclient is used to retrieve a zspconfig configuration file from the /tftpboot area of the Master switch.

/etc/rcZ.d/surviving_partner/vrrpd.script
: Runtime script that executes each time the vrrpd changes state. This script starts and stops dhcpd, and toggles down Intel ANS ports to force the Intel ANS nodes to a new Master switch.

/etc/rcZ.d/surviving_partner/zlmd.script
Runtime script executed by zlmd when a link goes up or down. This script modifies the priority of vrrpd, which in turn may cause the VRRP Master to move from one sibling switch to another.

SEE ALSO

zconfig, ifconfig, vrrpd, dhclient, dhcpd

zsync

NAME

zsync – Saves changes to the flash.

SYNOPSIS

```
zsync [-c][-f][<dir_or_file>]
zsync [-c][-f][-t <file>]
zsync [-c][-f][-z]
zsync [-c][-l]
```

DESCRIPTION

zsync is used to save a snapshot of the current file system to flash ROM. By default, *zsync* creates a compressed tar image of the files that have changed and saves the image in the flash ROM. The saved image is expanded on reboot. The saved compressed tar image is called an “overlay.”

If a directory parameter is given to *zsync*, the contents of the directory are saved instead of searching for updated files. The specific purpose of the <directory> parameter is for saving files that have been mounted with *zmnt*. Using the *-t* option allows a tar image created by `zmnt -t` to be saved.

To correct a corrupted file that is saved to flash ROM with *zsync*, first reboot with the *-i* option (*see Switch Maintenance*). Use *zmnt* to put the corrupted file in the /mnt directory, open and correct the file, then *zsync* to the **`/mnt`** directory to save your changes and reboot.

There are two overlay areas: *dynamic* and *custom*. The dynamic overlay is the default. The custom overlay is used to create base configurations that are different then the Intel base configuration. The intent of the custom overlay is to support OEMs and other special customers that require an overlay different from the standard Intel overlay. The custom overlay is written by using the *-c* option.

The *-z* option zeroes an overlay area, returning the switch to the factor configuration.

Specific files or directories can be excluded from saving to flash by zsync by including an entry in /etc/exclude. Likewise, existing entries in /etc/exclude such as /tmp can be removed in order to save those files to flash with zsync.

OPTIONS

-c	Save files to the custom overlay
-t <file>	Read files to be saved from a tar file.
-z	Zero the overlay area.
-f	Do not confirm with user and do not warn if saving failed. Exit code can be examined to determine success or failure.
<dir_or_file>	Save only the named file, or save the named directory to the overlay. Contents of directories must be created with zmnt.
-l	List files that would be written. Do not flash.

EXAMPLES

To *zsync* only the hosts file:

```
cd /etc
zsync hosts
```

If you previously created a snapshot of an overlay to a tar file using *zmnt*,

```
zmnt –t overlay.tar
```

You can use *zsync* to restore the overlay on the switch directly from the tar file,

```
zsync –t overlay.tar
```

```
The restored overlay will be loaded upon the next reboot.
```

SEE ALSO

zmnt

brctl

NAME

brctl - Bridge and Spanning Tree Protocol administration.

SYNOPSIS

```
brctl [options]
```

DESCRIPTION

brctl is used to set up, maintain, and display the bridge configuration in the Linux kernel. brctl is a standard command included with Linux bridge support including Spanning Tree Protocol (STP), which has been modified to include Intel Fast Forward Bridge (ffbridge), and Fast Forward Port (ffport) options.

A bridge is a device commonly used to connect different networks together, so that these networks will appear as one network to the participants.

Each of the networks being connected corresponds to one physical interface, or port in the bridge. These individual networks are bundled into one bigger ('logical') network. This bigger network corresponds to the bridge network interface.

Multiple bridges can work together to create even larger networks using the IEEE 802.1d Spanning Tree Protocol. This protocol is used for finding the shortest path between two networks as well as eliminating loops from the topology. Bridges communicate with each other by sending and receiving Bridge Protocol Data Units (BPDUs).

brctl can be used for configuring certain spanning tree protocol parameters. For an explanation of these parameters, see the IEEE 802.1d specification for detailed information.

OPTIONS

addbr <bridge>
: creates a new instance of a bridge. The network interface corresponding to the bridge will be called <bridge>. For the OpenArchitect switch, bridges are named bzhp0, bzhp1, etc.

  NOTE: This option must only be executed by zl2d.

delbr <bridge>
: deletes the instance <bridge> of an ethernet bridge. The network interface corresponding to the bridge must be down before it can be deleted.

  NOTE: This option must only be executed by zl2d.

show
: shows all current bridges.

```
addif  <bridge>  <interface>
            makes the interface a port of the bridge. This
            means that all frames received on the interface
            will be processed as if destined for the bridge.
            Also, when sending frames on the bridge, the in-
            terface will be considered as a potential output
            interface. For the OpenArchitect switch, <inter-
            face> is zhp0, zhp1, Ö

            NOTE: This option must only be executed by zl2d.

delif  <bridge>  <interface>
            detaches the interface from the bridge.

            NOTE: This option must only be executed by zl2d.

showbr  <bridge>
            shows information for the bridge and it's at-
            tached ports.

showmacs  <bridge>
            shows a list of learned MAC addresses for the
            bridge.

setageingtime  <bridge>  <time>
            sets the ethernet (MAC) address aging time, in
            seconds. After <time> seconds of not having seen
            a frame coming from a certain address, the bridge
            will time out (delete) that address from the For-
            warding DataBase (fdb).

setgcint  <bridge>  <time>
            sets the garbage collection interval for the
            bridge to <time> seconds. This means that the
            bridge will check the forwarding database for
            timed out entries every <time> seconds.

stp  <bridge>  <state>
            controls this bridge's participation in the Span-
            ning Tree Protocol. When turned off, the bridge
            will not send or receive BPDUs, and will thus not
            participate in the Spanning Tree Protocol. If
            your bridge isn't the only bridge on the LAN, or
            if there are loops in the LAN's topology, DO NOT
            turn this option off. If you turn this option
            off, please know what you are doing.

setbridgeprio  <bridge>  <priority>
            sets the bridge's priority to <priority>. The
            priority value is an unsigned 16-bit quantity (a
            number between 0 and 65535), and has no dimen-
            sion. Lower priority values are 'better'. The
            bridge with the lowest priority will be elected
            'Root Bridge'.
```

```
setfd  <bridge>  <time>
            sets the bridge's 'bridge forward delay' to
            <time> seconds.

sethello <bridge> <time>
            sets the bridge's 'bridge hello time' to <time>
            seconds.

setmaxage  <bridge>  <time>
            sets the bridge's 'maximum message age' to <time>
            seconds.

setpathcost  <bridge>  <port>  <cost>
            sets the port cost of the port to <cost>. This is
            a dimensionless metric. The path cost is set to
            100 for all OpenArchitect switch ports by de-
            fault. IEEE 802.d recommends the following:
```

Link Speed	Recommended Value	Recommended Range
10 Mb/s	100	50-600
100 Mb/s	19	10-60
1 Gb/s	4	3-10

```
setportprio  <bridge>  <port>  <priority>
            sets the port's priority to <priority>. The pri-
            ority value is an unsigned 8-bit quantity (a num-
            ber between 0 and 255), and has no dimension.
            This metric is used in the designated port and
            root port selection algorithms. For the OpenAr-
            chitect switch a port is zre1, zre2, Ö

ffbridge  <bridge>  <state>
            sets the bridge's Fast Forward state to enabled
            or disabled. If Fast Forward is enabled for the
            bridge, then whenever STP picks a new Root Port,
            that port will bypass the Listening and Learning
            states and transition immediately to Forwarding.

ffport  <bridge>  <port>  <state>
            sets the port's Fast Forward state to enabled or
            disabled. If Fast Forward is enabled for this
            port, then this port will always bypass the Lis-
            tening and Learning states and transition imme-
            diately to Forwarding. Fast Forward should ONLY
            be enabled on ports connected to end-stations.
```

NOTES

brctl replaces the older *brcfg* tool.

SEE ALSO

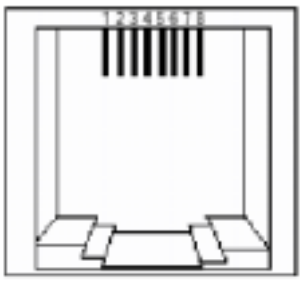
zconfig, zl2d

RS-232 Serial Cable A

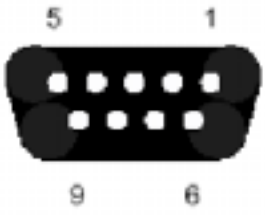
The serial port on the front panel of the ZT 8102 switch requires a serial cable to connect the switch to a terminal or a PC running terminal emulation software. This RJ-45 to DB-9 conversion cable is a standard Ethernet cable with the serial signal assignments listed below.

A.1 Pin Assignments

RJ-45 Pin	Signal
1	RTS
2	DTR
3	TxD
4	GND
5	GND
6	RxD
7	DSR
8	CTS



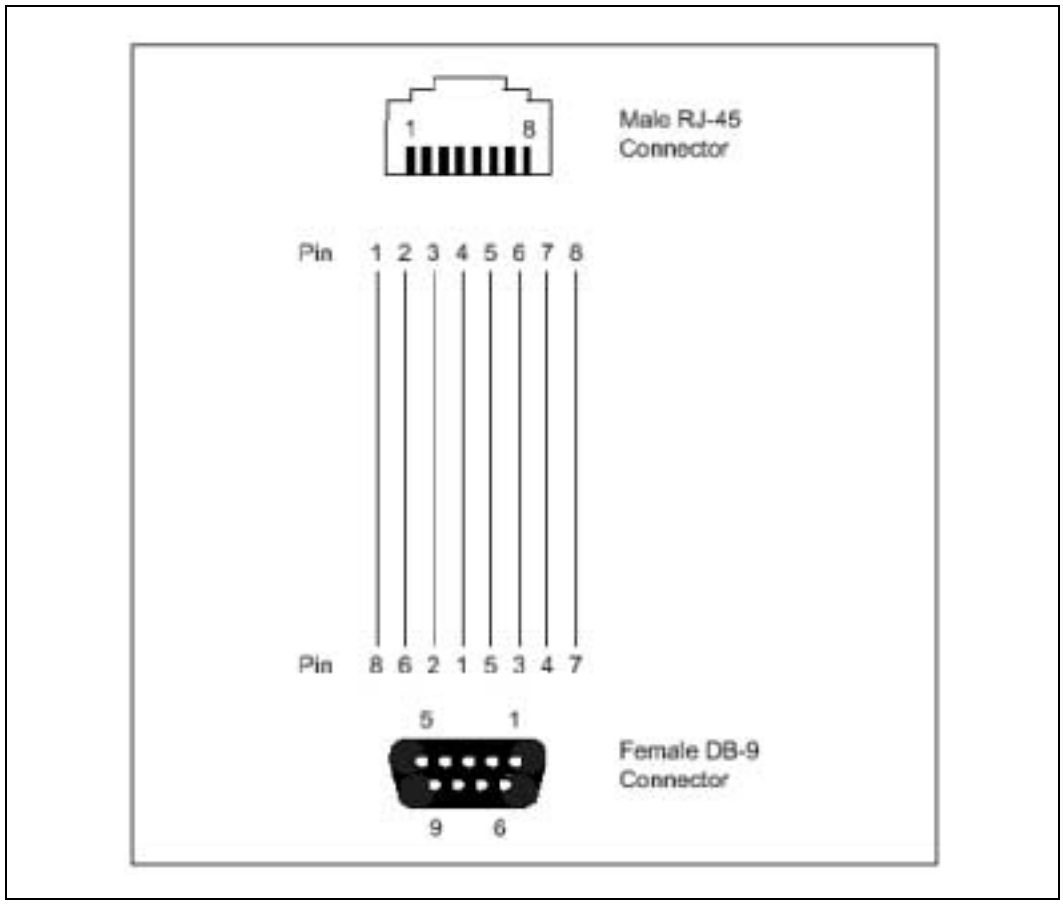
DB-9 Pin	Signal
1	GND Shield
2	RxD
3	TxD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	NC



A.2 Building the Cable

The serial cable allows you to manage the switch from the serial port on the front panel. To build this cable, use a RJ-45 connector for the switch, a DB-9 connector for the terminal, and the following diagram as a guide for the wiring.

Figure 21. Serial Cable Diagram



Real Panel Pinouts B

The ZT 8102 rear panel uses the same connectors as a CompactPCI* 6U Module. The following tables outlines the pins used in J1 through J5. J1 is keyed as code number 3456. J4 is keyed as code number 1235.

Table 5. J1 Connector Pinout

	Z	A	B	C	D	E	F
25	GND	EARLY_5V	PreCharge	ENUM#	EARLY_3.3V	EARLY_5V	GND
24	GND	PreCharge	EARLY_5V	-NC-	PreCharge	PreCharge	GND
23	GND	EARLY_3.3V	PreCharge	PreCharge	EARLY_5V	PreCharge	GND
22	GND	PreCharge	GND	EARLY_3.3V	PreCharge	-NC-	GND
21	GND	EARLY_3.3V	PreCharge	PreCharge	-NC-	PreCharge	GND
20	GND	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
19	GND	EARLY_3.3V	PreCharge	PreCharge	GND	PreCharge	GND
18	GND	PreCharge	-NC-	EARLY_3.3V	PreCharge	-NC-	GND
17	GND	EARLY_3.3V	IPMB_SCL	IPMB_SDA	GND	PreCharge	GND
16	GND	PreCharge	GND	-NC-	PreCharge	-NC-	GND
15	GND	EARLY_3.3V	PreCharge	PreCharge	-NC-	PreCharge	GND
12-14							
11	GND	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
10	GND	PreCharge	GND	EARLY_3.3V	PreCharge	PreCharge	GND
9	GND	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
8	GND	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
7	GND	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
6	GND	PreCharge	-NC-	EARLY_3.3V	-NC-	PreCharge	GND
5	GND	-NC-	-NC-	PreCharge	GND	PreCharge	GND
4	GND	IPMB_PWR	HEALTHY#	-NC-	-NC-	-NC-	GND
3	GND	PreCharge	PreCharge	PreCharge -	EARLY_5V	PreCharge	GND
2	GND	-NC-	EARLY_5V	-NC-	-NC-	-NC-	GND
1	GND	EARLY_5V	-NC-	-NC-	EARLY_12V	EARLY_5V	GND
	Z	A	B	C	D	E	F

Table 6. Backplane Pin Descriptions (Sheet 1 of 2)

Name	Count	Type	Description
EARLY_5 V	8	Power	Early 5 V and 5 V power supply pins
EARLY_3.3 V	10	Power	Early 3.3 V and 3.3 V power supply pins
IPMB_PWR	1	Power	IPMB power supply pin

Table 6. Backplane Pin Descriptions (Sheet 2 of 2)

Name	Count	Type	Description
HEALTHY#	1	Output	Healthy status
IPMB_SCL	1	Output	IPMB Clock
IPMB_SDA	1	I/O	IPMB Address/Data
ENUM#	1	O	To indicate Switch blade is freshly inserted.
EARLY_12V	1	Power	Early 12V power supply pin
GND	76	Power	GND pins. There are more GND pin at J3~J5.
-NC-			No connected pins.
TOTAL	220		

Table 7. J2 Connector Pinout

	A	B	C	D	E	F
22	GA4	GA3	GA2	GA1	GA0	GND
21	-NC-	GND	-NC-	-NC-	-NC-	GND
20	-NC-	-NC-	-NC-	GND	-NC-	GND
19	-NC-	GND	-NC-	-NC-	-NC-	GND
18	-NC-	-NC-	-NC-	GND	-NC-	GND
17	-NC-	GND	-NC-	-NC-	-NC-	GND
16	-NC-	-NC-	-NC-	GND	-NC-	GND
15	-NC-	GND	-NC-	-NC-	-NC-	GND
14	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
13	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
12	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
11	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
10	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
9	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
8	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
7	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
6	PreCharge	PreCharge	PreCharge	GND	PreCharge	GND
5	PreCharge	GND	-NC-	PreCharge	PreCharge	GND
4	-NC-	-NC-	PreCharge	GND	PreCharge	GND
3	-NC-	GND	-NC-	-NC-	-NC-	GND
2	-NC-	-NC-	-NC-	-NC-	-NC-	GND
1	-NC-	GND	-NC-	-NC-	-NC-	GND

Table 8. J3 Connector Pinout

	A	B	C	D	E	F
19	SGA4	SGA3	SGA2	SGA1	SGA0	GND
18	TX1+F	TX1-F	GND	-NC-	-NC-	GND
17	TX2+F	TX2-F	GND	-NC-	-NC-	GND
16	TX1+8	TX1-8	GND	TX3+8	TX3-8	GND
15	TX2+8	TX2-8	GND	TX4+8	TX4-8	GND
14	TX1+7	TX1-7	GND	TX3+7	TX3-7	GND
13	TX2+7	TX2-7	GND	TX4+7	TX4-7	GND
12	TX1+6	TX1-6	GND	TX3+6	TX3-6	GND
11	TX2+6	TX2-6	GND	TX4+6	TX4-6	GND
10	TX1+5	TX1-5	GND	TX3+5	TX3-5	GND
9	TX2+5	TX2-5	GND	TX4+5	TX4-5	GND
8	TX1+4	TX1-4	GND	TX3+4	TX3-4	GND
7	TX2+4	TX2-4	GND	TX4+4	TX4-4	GND
6	TX1+3	TX1-3	GND	TX3+3	TX3-3	GND
5	TX2+3	TX2-3	GND	TX4+3	TX4-3	GND
4	TX1+2	TX1-2	GND	TX3+2	TX3-2	GND
3	TX2+2	TX2-2	GND	TX4+2	TX4-2	GND
2	TX1+1	TX1-1	GND	TX3+1	TX3-1	GND
1	TX2+1	TX2-1	GND	TX4+1	TX4-1	GND

Table 9. J4 Connector Pinout (Sheet 1 of 2)

	A	B	C	D	E	F
25	-NC-	-NC-	GND	-NC-	-NC-	GND
24	-NC-	-NC-	GND	-NC-	-NC-	GND
23	-NC-	-NC-	GND	-NC-	-NC-	GND
22	-NC-	-NC-	GND	-NC-	-NC-	GND
21	-NC-	-NC-	GND	-NC-	-NC-	GND
20	-NC-	-NC-	GND	-NC-	-NC-	GND
19	-NC-	-NC-	GND	-NC-	-NC-	GND
18	-NC-	-NC-	GND	-NC-	-NC-	GND
17	-NC-	-NC-	GND	-NC-	-NC-	GND
16	-NC-	-NC-	GND	-NC-	-NC-	GND
15	-NC-	-NC-	GND	-NC-	-NC-	GND
14	-NC-	-NC-	-NC-	-NC-	-NC-	GND
13	-NC-	-NC-	-NC-	-NC-	-NC-	GND
12	-NC-	-NC-	-NC-	-NC-	-NC-	GND
11	5V	5V	GND	3.3V	3.3V	GND
10	RTS_BP	CTS_BP	TXD_BP	RXD_BP	SMBD	GND

Table 9. J4 Connector Pinout (Sheet 2 of 2)

	A	B	C	D	E	F
9	DTR_BP	DSR_BP	IPMB_PWR	SMBA	SMBC	GND
8	-NC-	-NC-	-NC-	-NC-	-NC-	GND
7	-NC-	-NC-	-NC-	-NC-	-NC-	GND
6	-NC-	-NC-	-NC-	-NC-	-NC-	GND
5	-NC-	-NC-	-NC-	-NC-	-NC-	GND
4	-NC-	-NC-	-NC-	-NC-	-NC-	GND
3	-NC-	-NC-	-NC-	-NC-	-NC-	GND
2	-NC-	-NC-	-NC-	-NC-	-NC-	GND
1	-NC-	-NC-	-NC-	-NC-	-NC-	GND

Table 10. J5 Connector Pinout

	A	B	C	D	E	F
22	-NC-	-NC-	GND	-NC-	-NC-	GND
21	-NC-	-NC-	GND	-NC-	-NC-	GND
20	-NC-	-NC-	GND	-NC-	-NC-	GND
19	-NC-	-NC-	GND	-NC-	-NC-	GND
18	-NC-	-NC-	GND	-NC-	-NC-	GND
17	-NC-	-NC-	GND	-NC-	-NC-	GND
16	-NC-	-NC-	GND	-NC-	-NC-	GND
15	-NC-	-NC-	GND	-NC-	-NC-	GND
14	-NC-	-NC-	GND	-NC-	-NC-	GND
13	-NC-	-NC-	GND	-NC-	-NC-	GND
12	TX1+14	TX1-14	GND	TX3+14	TX3-14	GND
11	TX2+14	TX2-14	GND	TX4+14	TX4-14	GND
10	TX1+13	TX1-13	GND	TX3+13	TX3-13	GND
9	TX2+13	TX2-13	GND	TX4+13	TX4-13	GND
8	TX1+12	TX1-12	GND	TX3+12	TX3-12	GND
7	TX2+12	TX2-12	GND	TX4+12	TX4-12	GND
6	TX1+11	TX1-11	GND	TX3+11	TX3-11	GND
5	TX2+11	TX2-11	GND	TX4+11	TX4-11	GND
4	TX1+10	TX1-10	GND	TX3+10	TX3-10	GND
3	TX2+10	TX2-10	GND	TX4+10	TX4-10	GND
2	TX1+9	TX1-9	GND	TX3+9	TX3-9	GND
1	TX2+9	TX2-9	GND	TX4+9	TX4-9	GND



Warranty Information C

C.1 Intel® NetStructure™ Compute Boards and Platform Products Limited Warranty

Intel warrants to the original owner that the product delivered in this package will be free from defects in material and workmanship for two (2) year(s) following the latter of: (i) the date of purchase only if you register by returning the registration card as indicated thereon with proof of purchase; or (ii) the date of manufacture; or (iii) the registration date if by electronic means provided such registration occurs within 30 days from purchase. This warranty does not cover the product if it is damaged in the process of being installed. Intel recommends that you have the company from whom you purchased this product install the product.

THE ABOVE WARRANTY IS IN LIEU OF ANY OTHER WARRANTY, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ANY WARRANTY OF INFRINGEMENT OF ANY OTHER PARTY'S INTELLECTUAL PROPERTY RIGHTS, OR ANY WARRANTY ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

This warranty does not cover replacement of products damaged by abuse, accident, misuse, neglect, alteration, repair, disaster, improper installation or improper testing. If the product is found to be otherwise defective, Intel, at its option, will replace or repair the product at no charge except as set forth below, provided that you deliver the product along with a return material authorization (RMA) number (see below) either to the company from whom you purchased it or to Intel. If you ship the product, you must assume the risk of damage or loss in transit. You must use the original container (or the equivalent) and pay the shipping charge. Intel may replace or repair the product with either a new or reconditioned product, and the returned product becomes Intel's property. Intel warrants the repaired or replaced product to be free from defects in material and workmanship for a period of the greater of: (i) ninety (90) days from the return shipping date; or (ii) the period of time remaining on the original two (2) year warranty.

This warranty gives you specific legal rights and you may have other rights which vary from state to state. All parts or components contained in this product are covered by Intel's limited warranty for this product. The product may contain fully tested, recycled parts, warranted as if new.

C.1.1 Returning a Defective Product (RMA)

Before returning any product, contact an Intel Customer Support Group to obtain either a Direct

Return Authorization (DRA) or Return Material Authorization (RMA). Return Material

Authorizations are only available for products purchased within 30 days.

Return contact information by geography:

C.1.2 For the Americas

Return Material Authorization (RMA) credit requests e-mail address: requests.rma@intel.com

Direct Return Authorization (DRA) repair requests e-mail address: uspss.repair@intel.com

DRA on-line form: http://support.intel.com/support/motherboards/draform.htm

Intel Business Link (IBL): http://www.intel.com/ibl

Telephone No.: 1-800-INTEL4U or 480-554-4904

Office Hours: Monday - Friday 0700-1700 MST Winter / PST Summer

C.1.3 For EMEA

Return Material Authorization (RMA) e-mail address - emea.fs@intel.com

Direct Return Authorization (DRA) for repair requests e-mail address: emea.fs@intel.com

Intel Business Link (IBL): http://www.intel.com/ibl

Telephone No.: 00 44 1793 403063

Fax No.: 00 44 1793 403109

Office Hours: Monday - Friday 0900-1700 UK time

C.1.4 For APAC

RMA/DRA requests email address: apac.rma.front-end@intel.com

Telephone No.: 604-859-3111 or 604-859-3325

Fax No.: 604-859-3324

Office Hours: Monday - Friday 0800-1700 Malaysia time

Return Material Authorization (RMA) requests e-mail address: rma.center.jpss@intel.com

Telephone No.: 81-298-47-0993 or 81-298-47-5417

Fax No.: 81-298-47-4264

Direct Return Authorization (DRA) for repair requests, contact the JPSS Repair center.

E-mail address: sugiyamakx@intel.co.jp

Telephone No.: 81-298-47-8920

Fax No.: 81-298-47-5468

Office Hours: Monday - Friday 0830-1730 Japan time

If the Customer Support Group verifies that the product is defective, they will have the Direct Return Authorization/Return Material Authorization Department issue you a DRA/RMA number to place on the outer package of the product. Intel cannot accept any product without a DRA/RMA number on the package. Limitation of Liability and Remedies

INTEL SHALL HAVE NO LIABILITY FOR ANY INDIRECT OR SPECULATIVE DAMAGES (INCLUDING , WITHOUT LIMITING THE FOREGOING, CONSEQUENTIAL, INCIDENTAL AND SPECIAL DAMAGES) ARISING FROM THE USE OF OR INABILITY TO USE THIS PRODUCT, WHETHER ARISING OUT OF CONTRACT, NEGLIGENCE, TORT, OR UNDER ANY WARRANTY, OR FOR INFRINGEMENT OF ANY OTHER PARTY'S INTELLECTUAL PROPERTY RIGHTS, IRRESPECTIVE OF WHETHER INTEL HAS ADVANCE NOTICE OF THE POSSIBILITY OF ANY SUCH DAMAGES, INCLUDING, BUT NOT LIMITED TO LOSS OF USE, BUSINESS INTERRUPTIONS, AND LOSS OF PROFITS. NOTWITHSTANDING THE FOREGOING, INTEL'S TOTAL LIABILITY FOR ALL CLAIMS UNDER THIS AGREEMENT SHALL NOT EXCEED THE PRICE PAID FOR THE PRODUCT. THESE LIMITATIONS ON POTENTIAL LIABILITIES WERE AN ESSENTIAL ELEMENT IN SETTING THE PRODUCT PRICE. INTEL NEITHER ASSUMES NOR AUTHORIZES ANYONE TO ASSUME FOR IT ANY OTHER LIABILITIES.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

Agency Approvals D

D.1 CE Certification

The ZT 8102 meets the intent of Directive 89/336/EEC for Electromagnetic Compatibility & Low-Voltage Directive 73/23/EEC for Product Safety. The ZT 8102 has been designed for NEBS/ETSI compliance.

D.2 Safety

UL/cUL 60950	Safety for Information Technology Equipment (UL File # E139761)
EN/IEC 60950	Safety for Information Technology Equipment
CB Report Scheme	CB Certificate and Report

D.3 Emissions Test Regulations

FCC Part 15, Subpart B

EN 55022

CISPR 22

Bellcore GR-1089

D.3.1 EN 50081-1 Emissions

GR-1089-CORE	Sections 2 and 3
EN 55022	Class A Radiated Emissions
EN 55022	Power Line Conducted Emissions
EN 61000-3-2	Power Line Harmonic Emissions
EN 61000-3-3	Power Line Fluctuation and Flicker

D.3.2 EN 55024 Immunity

GR-1089-CORE	GR-1089-CORE Sections 2 and 3
EN 61000 4-2	Electrostatic Discharge (ESD)
EN 61000 4-3	Radiated Susceptibility
EN 61000 4-4	Electrical Fast Transient Burst
EN 61000 4-5	Power Line Surge
EN 61000 4-6	Frequency Magnetic Fields
EN 61000 4-11	Voltage Dips, Variations, & Short Interruptions

D.4 Regulatory Information

D.4.1 FCC—Federal Communications Commission (USA)

This product has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment.

This product generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Note: This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Caution: If you make any modification to the equipment not expressly approved by Intel, you could void your authority to operate the equipment.

D.4.2 Industry Canada (Canada)

Cet appareil numérique respecte les limites bruits radioélectriques applicables aux appareils numériques de Classe A prescrites dans la norme sur le matériel brouilleur: "Appareils Numériques," NMB-003 édictée par le Ministre Canadien des Communications.

This digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the interference-causing equipment standard entitled: "Digital Apparatus," ICES-003 of the Canadian Department of Communications.

D.4.3 Product Safety Information

Safety Precautions

Review the following precautions to avoid injury and prevent damage to this product, or products to which it is connected. To avoid potential hazards, use the product only as specified.

Read all safety information provided in the component product user manuals and understand the precautions associated with safety symbols, written warnings, and cautions before accessing parts or locations within the unit.

Caution: **To Avoid Electric Overload.** To avoid electrical hazards (heat shock and/or fire hazard), do not make connections to terminals outside the range specified for that terminal.

Caution: **To Avoid the Risk of Electric Shock.** When supplying power to the system, always make connections to a grounded main. Always use a power cable with a grounded plug (third grounding pin). Do not operate in wet, damp, or condensing conditions.

Caution: **System Airflow Requirements.** Platform components such as single board computers, Ethernet switches, etc., are designed to operate with external air flow.

Components can be destroyed if they are operated without external air flow. External air flow is normally provided by chassis fans when components are installed in compatible chassis.

Filler panels must be installed over unused chassis slots so that airflow requirements are met.

Refer to the product data sheet for airflow requirements if you are installing components in custom chassis.

Warning: **Microprocessor Heatsinks May Become Hot During Normal Operation.** To avoid burns, do not allow anything to touch processor heatsinks.

Caution: **Do Not Operate Without Covers.** To avoid electric shock or fire hazard, do not operate this product with any removed enclosure covers or panels.

Caution: **Do Not Operate in an Explosive Atmosphere.** To avoid injury, fire hazard, or explosion, do not operate this product in an explosive atmosphere.

Caution: **If Your System Has Multiple Power Supply Sources.** Disconnect all external power connections before servicing.

Warning: Power Supplies Must Be Replaced by Qualified Service Personnel Only.

Caution: **Lithium batteries are not field-replaceable units.** There is a danger of explosion if a battery is incorrectly replaced or handled. Do not disassemble or recharge the battery. Do not dispose of the battery in fire. When the battery is replaced, the same type or an equivalent type recommended by the manufacturer must be used. Used batteries must be disposed of according to the manufacturer's instructions. Return the unit to Intel for battery service.

Caution: Many components in the system contain sensitive electronic components. Service personnel should follow proper grounding procedures when installing or servicing this equipment.

Customer Support *E*

This appendix offers technical and sales assistance information for this product, and information on returning an Intel® NetStructure™ product for service.

E.1 G.1 Technical Support and Return for Service

For all product returns and support issues, please contact your Intel product distributor or Intel Sales Representative for specific information.

E.2 G.2 Sales Assistance

If you have a sales question, please contact your local Intel NetStructure Sales Representative or the Regional Sales Office for your area. Address, telephone and FAX numbers, and additional information is available at Intel's website, located at http://www.intel.com/network/csp/sales.

Intel Corporation

Telephone (in U.S.) 1-800-755-4444

Telephone (Outside U.S.) 1-973-993-3030

FAX 1-973-967-8780

Glossary

Application Flash ROM	This is code embedded in the Boot ROM which may contain one or more images that are placed there as a value-added operation to generate an appropriate executable, compressed image.
Boot loader	The first executable used to start another program. For the switch, a boot loader is placed within the Boot ROM device and is used to start the Linux software from the internal processor.
Boot ROM Image	Defined code (firmware) that is programmed into the Boot ROM located in the processor (see Boot ROM). In the ZT 8102 switches, the Boot ROM image is also the boot loader.
Boot ROM	A flash ROM device connected to the X bus of the processor. The switches use a MPC8240 Boot ROM.
CarrierClass™	A term used to identify telecom products that are capable of meeting the requirements of the telecom industry.
daemon	A program that is not invoked explicitly, but lies dormant waiting for some condition(s) to occur. The idea is that the perpetrator of the condition need not be aware that a daemon is lurking (though often a program will commit an action only because it knows that it will implicitly invoke a daemon). Unix systems run many daemons, chiefly to handle requests for services from other hosts on a network. Most of these are now started as required by a single real daemon, *inetd*, rather than running continuously. Examples are *cron* (local timed command execution), *rshd* (remote command execution), *rlogind* and *telnetd* (remote login), *ftpd*, *nfsd* (file transfer), *lpd* (printing).
Domain Name System (DNS)	Name resolution software that lets users locate computers on a UNIX network or the Internet (TCP/IP network) by domain name. The DNS server maintains a database of domain names (host names) and their corresponding IP addresses. In the following hypothetical example, if **www.mycompany.com** were presented to a DNS server, the IP address **204.0.8.51** would be returned.
dhclient	Run by a replacement switch at initialization at get its configuration from the Surviving Partner
Dynamic Host Configuration Protocol (DHCP)	Software that automatically assigns IP addresses to client stations logging onto a TCP/IP network. It eliminates having to manually assign permanent IP addresses. DHCP software typically runs in servers and is also found in network devices such as ISDN routers and modem routers that allow multiple users access to the Internet. Newer DHCP servers dynamically update the DNS servers after making assignments.
dhcpd	Runs the DHCP server. Started when a switch is the VRRP Master to configure replacement switches.

ftp — (File Transfer Program) FTP is a command in Linux that sends and receives files interactively from your system hard drive or other remote computer systems. The ftp command includes built-in help.

flash memory — A memory chip that can be rewritten and hold its content without power. It is widely used for digital film and for data and programs for communications and industrial products as well as a variety of handheld devices. Flash chips generally have life spans from 100K to 300K write cycles.

gated — A modular software program consisting of core services, a routing database, and protocol modules supporting multiple routing protocols such as RIP (V1 and V2) and OSPF.

Hot Swap™ — A set of functions that allows you to remove and reinstall a board in a system while the power is on.

Network File System (NFS)
NFS is a file sharing protocol in a UNIX network. A file sharing protocol is a high-level network protocol that provides the structure and language for file requests between clients and servers. It provides the commands for opening, reading, writing and closing files across the network and may also provide access to the directory services. Sometimes called a "client/server protocol," it functions at the application layer (layer 7 of the OSI model).

In order for a client to have access to multiple servers running different operating systems, either the client supports the file sharing protocol of each operating system or the server supports the file sharing protocol of each client.

OpenArchitect* — A registered trademark of ZNYX Networks, this software platform which is based on Linux*, enables system administrators and programmers to configure and customize Ethernet switching directly through a switch.

Open Shortest Path First Protocol (OSPF)
A routing protocol that determines the best path for routing IP traffic over a TCP/IP network. OSPF is an interior gateway protocol (IGP) which is designed to work within an autonomous system.

Peripheral Component Interconnect (PCI) bus
A peripheral bus commonly used in PCs. PCI provides a high-speed data path between the CPU and peripheral devices (video, disk, network, etc.).

power cycle — The process of rebooting the switch by turning the computer system on and off.

Rear Transition Board (RTB)
The rear panel egress board that is used in conjunction with the Compact PCI adapters. By convention, Intel refers to rear panel boards for non "P" models as RTM's. "P" models use rear transition board (RTB).

Rear Transition Module (RTM)
The rear panel egress board that is used in conjunction with the Compact PCI adapters. By convention, Intel refers to rear panel boards for non "P" models as RTM's. "P" models use rear transition board (RTB).

reset button	The button on the OpenArchitect* switch which causes the switch to reboot.
Routing Information Protocol (RIP)	A simple routing protocol that is part of the TCP/IP protocol suite. It determines a route based on the smallest hop count between source and destination. RIP is a distance vector protocol that routinely broadcasts routing information to its neighboring routers.
routing protocol	A formula used by routers to determine the appropriate path onto which data should be forwarded. Routing protocol also specifies how routers report changes and share information with the other routers in the network that they can reach. A routing protocol allows the network to dynamically adjust to changing conditions, otherwise all routing decisions have to be predetermined and remain static. RIP and OSPF are examples of routing protocols.
tagged	Data packets which are tagged with a specific 802.1Q VLAN identifier indicating the VLAN from which the data came.
telnet	A terminal emulation protocol that enables a user to log onto a remote device and run a program.
untagged	Data packets which do not belong to a specific 802.1Q VLAN and does not carry an identifier tag.
vi editor	(visual interface) A full-screen text editor that can be run from a terminal or the system console.
VLAN	(Virtual Local Area Network) A VLAN is a logical subgroup within a local area network that is created with software rather than physically reconfiguring network hubs and cables.
zhp	A virtual interface which represents a VLAN that includes specific OpenArchitect* switch ports.
zre	A virtual interface that represents a port on the OpenArchitect* switch.