24+2 Ethernet Switch

User's Guide

Your complete guide to the 24+2 Ethernet Switch's functions, commands, and unique features.

continuous
COMPUTING CORPORATION

24+2 Ethernet Switch

User's Guide

▶ *Your complete guide to the 24+2 Ethernet Switch's functions, commands, and unique features.*

continuous
COMPUTING CORPORATION

©2001 Continuous Computing Corporation. All rights reserved.

The information contained in this document is provided "as is" without any express representations of warranties. In addition, Continuous Computing Corporation disclaims all implied representations and warranties, including any warranty of merchantability, fitness for a particular purpose, or non-infringement of third party intellectual property rights.

This document contains proprietary information of Continuous Computing Corporation or under license from third parties. No part of this document may be reproduced in any form or by any means or transferred to any third party without the prior written consent of Continuous Computing Corporation.

Continuous Computing, the Continuous Computing Corporation logo, Continuous Control Node, Continuous System Controller and Field Replaceable System are trademarks or registered trademarks of Continuous Computing Corporation in the United States and other countries.

Sun, the Sun logo, SPARCengine, Solaris, and OpenBoot are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

CompactPCI is a registered trademark of PICMG.

The information contained in this document is not designed or intended for use in human life support systems, on-line control of aircraft, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Continuous Computing Corporation disclaims any express or implied warranty of fitness for such uses.

FCC Notices

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.

Table of Contents

1 Introduction

24+2 ETHERNET SWITCH CARD 1-1
SYSTEM BLOCK DIAGRAM 1-2
HIGH-LEVEL SYSTEM SPECIFICATIONS 1-3
PHYSICAL PORTS 1-4
10BASE-T/100BASE-T PORTS 1-4
1000BASE-T PORTS 1-5

LEDS AND CONNECTORS 1-5
FRONT PANEL LEDS 1-5
FRONT PANEL CONNECTORS 1-6
TRANSITION CARD 1-6
OPERATING MECHANICAL AND ENVIRONMENTAL 1-6
STORAGE/TRANSIT ENVIRONMENTAL 1-7
SAFETY COMPLIANCE 1-7
ELECTROMAGNETIC COMPATIBILITY (EMC) 1-7
TELCO COMPLIANCE 1-7
MARKS 1-7
BASIC FUNCTIONS OF LAYER-2 SWITCHING 1-7
UNICAST SWITCHING 1-8
VLAN Classification 1-8
Learning 1-8
Filtering 1-8
Forwarding 1-9
MULTICAST SWITCHING 1-9

VLAN 1-10
BROADCAST CONTAINMENT 1-10
MULTICAST-BASED MULTIMEDIA APPLICATIONS 1-10
ENHANCED SECURITY 1-10
VLAN MEMBERSHIP 1-11
Definitions of VLAN Membership 1-11
VLAN Membership Learning 1-11
Remote VLAN Learning 1-11
VLAN CONFIGURATION 1-12
Intra-VLAN Communication 1-12
Inter-VLAN Communication 1-12

CLASS-OF-SERVICE (COS) SUPPORT 1-12
QUALITY-OF-SERVICE (QOS) SUPPORT 1-12
GVRP 1-14
DHCP 1-14
ICMP ROUTER DISCOVERY 1-15
IGMP SNOOPING AND IP MULTICAST FILTERING 1-15

2 Installation

ELECTROSTATIC DISCHARGE (ESD) 2-1
UNPACKING 2-1

INSTALLING THE ETHERNET SWITCH AND TRANSITION CARD 2-1
INSTALLING THE SWITCH AND CARD IN THE CHASSIS 2-2

POWERING ON THE CPCI 24+2 ETHERNET SWITCH 2-3
CONNECTING THE INPUT/OUTPUT DEVICES 2-5
CONNECTING THE NETWORK DEVICES 2-5
CONNECTING THE CONSOLE PORT 2-6

CONNECTORS, PINOUTS, AND SPECIFICATIONS (PRELIMINARY/EXPECTED) 2-7

CONNECTOR USAGE 2-7

PINOUTS 2-8
SPLITTER RJ45 2-8
STANDARD RJ45 2-8
DB9 FEMALE 2-9

PIN ASSIGNMENTS 2-9
J5/P5 CONNECTOR 2-10
J4/P4 CONNECTOR 2-10
J3/P3 CONNECTOR 2-11
J2/P2 CONNECTOR 2-11
J1 CONNECTOR 2-12

3 CLI Commands

CONSOLE MANAGEMENT 3-1

LOGGING ON TO THE 24+2 ETHERNET SWITCH 3-2
THE 24+2 CLI COMMANDS (FIRMWARE) 3-2
SHOW COMMAND 3-2
UPLOAD COMMAND 3-4
SET COMMAND 3-4
REMOVE COMMAND 3-10
EXTRA COMMANDS 3-10

THE 24+2 CLI COMMANDS (BOOT) 3-11

BOOT [1 | 2] 3-12
RSTCON 3-12
UPLOAD 3-12
HELP 3-13

THE 24+2 CLI OUTPUT (FIRMWARE) 3-13

SHOW COMMAND 3-14
UPLOAD COMMAND 3-20
SET COMMAND 3-22
REMOVE COMMAND 3-37
PING COMMAND 3-38
RESET COMMAND 3-40

Extra Commands 3-41

THE 24+2 CLI OUTPUT (BOOT) 3-42

Help 3-42
Upload 3-42
boot [1 | 2] 3-42
rstcon 3-43

THE 24+2 CLI COMMAND INDEX (FIRMWARE) 3-43

Show Command 3-43
Upload Command........ 3-43
Set Command 3-43
Remove Command 3-44
Ping Command 3-44
Reset Command 3-45
Extra Commands 3-45

THE 24+2 CLI COMMAND INDEX (BOOT) 3-45

4 SNMP and RMON Management

OVERVIEW 4-1
SNMP AGENT AND MIB-2 (RFC1213) 4-2
RMON MIB (RFC 1757) AND BRIDGE MIB (RFC 1493) 4-2
RMON Groups Supported 4-2
Bridge Groups Supported 4-3

VERTEX PRIVATE MIB 4-3

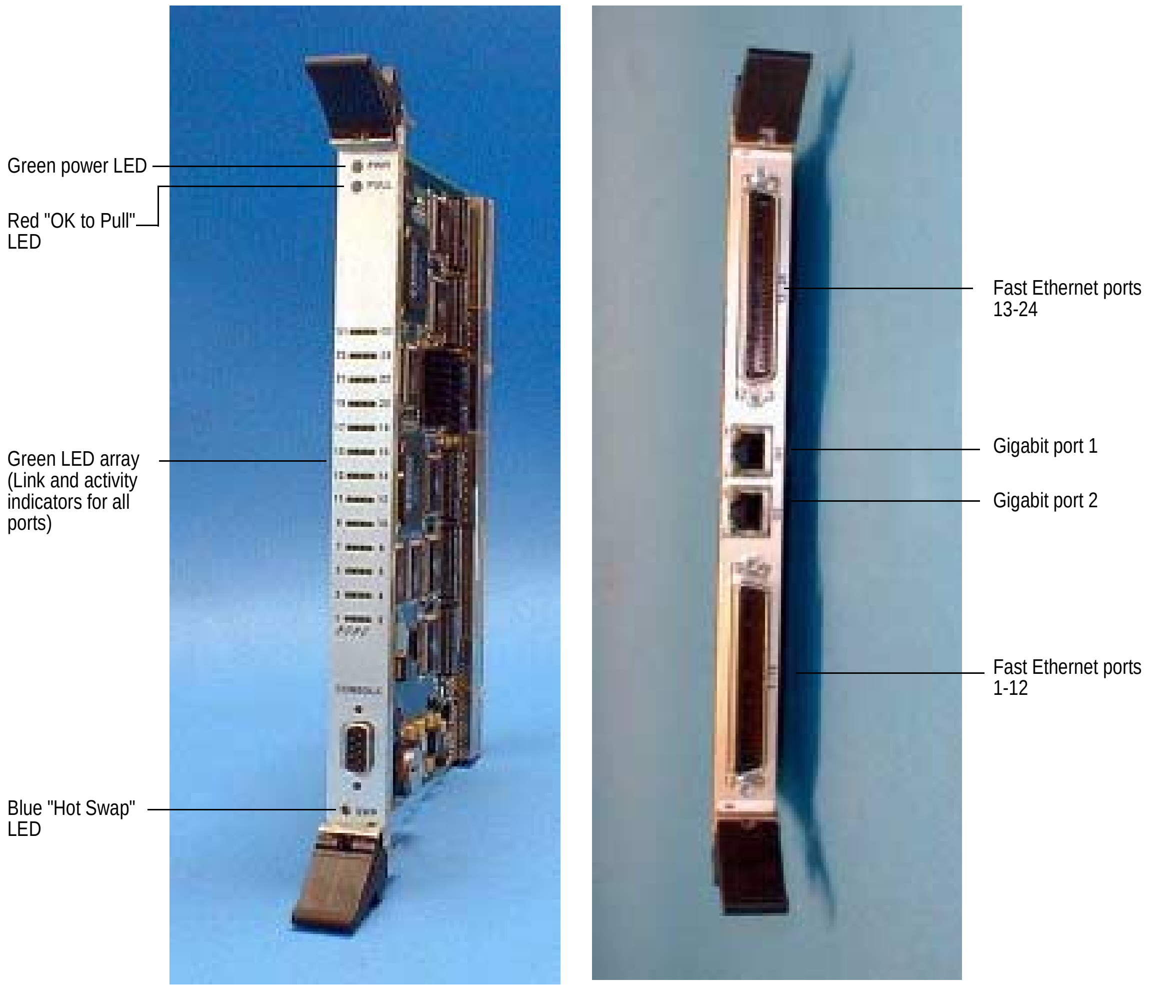
5 Troubleshooting and Technical Support

TROUBLESHOOTING 5-1
CONTACTING TECHNICAL SUPPORT 5-2

1 Introduction

The Continuous Computing 24+2 CompactPCI Ethernet Switch is a flexible, high-speed, state-of-the-art routing switch that offers a combination of Ethernet, Fast Ethernet, and Gigabit Ethernet solutions.

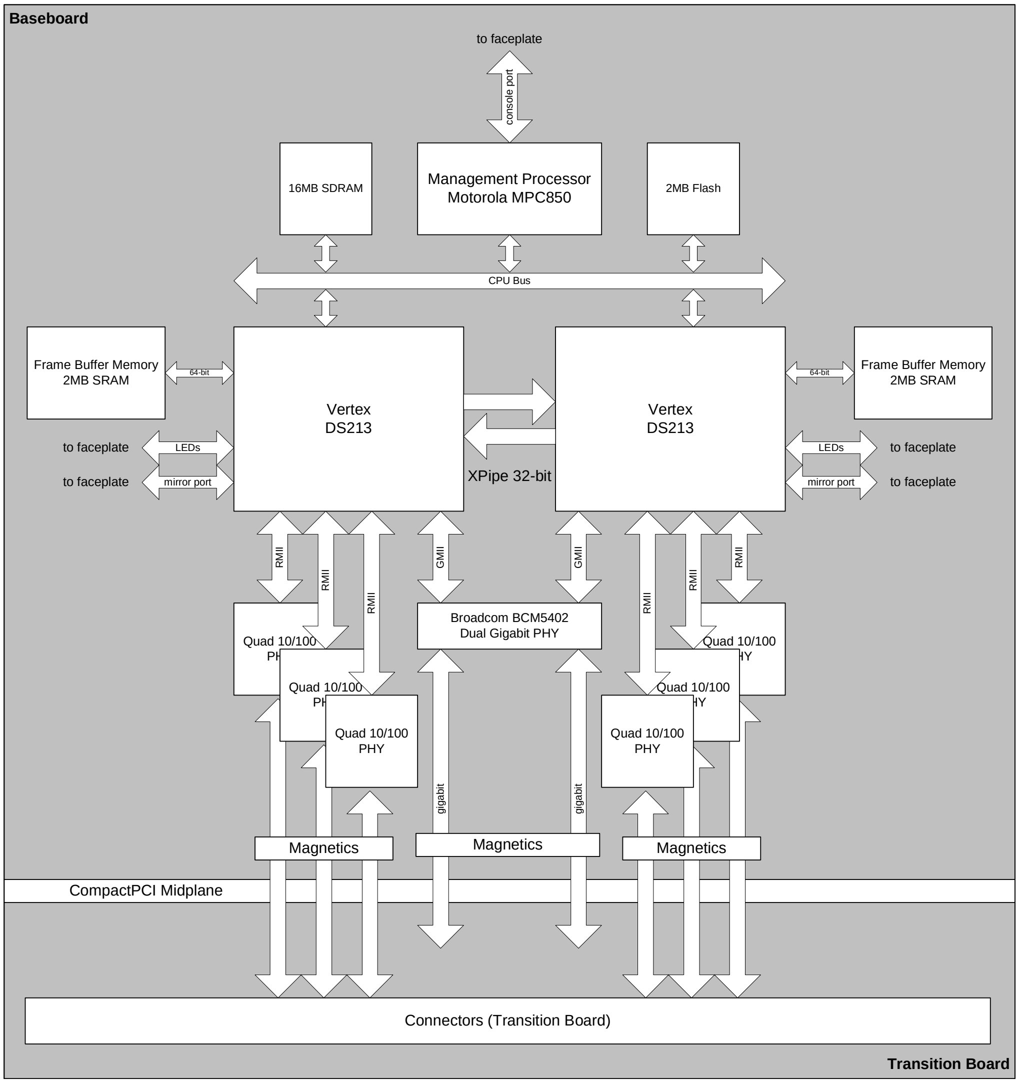
24+2 Ethernet Switch Card



24+2 Ethernet Switch

Transition card

System Block Diagram



System block diagram

High-Level System Specifications

The 24+2 Ethernet Switch provides the following key features:

- Layer-2 switching:
  - 6.54Mpps maximum forwarding rate per system
  - 148.8Kpps on all Fast Ethernet ports
  - 1.488Mpps on both Gigabit ports
  - Up to 8K MAC addresses per system, with a 2K internal cache on the chip and a 14K external table on SRAM sharing the packet buffer memory
- Full-duplex (802.3X) and half-duplex (back pressure) flow control
- VLAN support:
  - Port-based VLAN
  - 802.1Q tagged VLAN
- Hardware-based RMON counters
- Flooding/broadcast storm control
- Two LEDs per port: Link Status and Activity
- Switch manager CPU:

  50 MHz Motorola MPC-850 embedded processor

  16 M bytes DRAM

  1 M byte flash ROM
- Packet filtering of source and destination MAC addresses
- Port security
  - Limit the number of MAC addresses learned per port
  - Static MAC addresses stay in the filtering table
- CoS - 802.1P
  - Four queues per output port
  - Packet transmission scheduled using Weighted Round Robin (WRR)
  - User-defined weights
  - Classification of packet priority can be based on either a VLAN tag on packet or a user-definable port priority
- Port trunking (link aggregation)
  - Supports four groups total, maximum of eight ports per group
  - Load sharing based on source and destination MAC addresses

- Port mirroring provided through dedicated port, Port 0, or Port 13
- Internetworking protocols:

Bridging:	- 802.1D Spanning Tree
	- 802.1P/Q - GARP/GVRP
Routing:	- RIP
	- RIP-2
	- DHCP-Relay
	- ICMP Router Discovery Message
IP Multicast:	- IGMP Snooping
	- IP Multicast Packet Filtering
	- Maximum of 256 VLANs and IP Multicast Sessions

- Network management:

One RS-232 port as local control console

Telnet remote control console

SNMP agent:
- MIB-2
- Bridge MIB (RFC1286)
- RMON MIB (RFC1757) — statistics, history, alarms, and events
- VLAN MIB (802.1Q)
- Vertex MIB

- Java applet-based MIB browser
- TFTP/Kermit software-upgrade capability

Physical Ports

The 24+2 Ethernet Switch supports 10Base-T/100Base-TX, and 1000Base-T ports.

10Base-T/100Base-T Ports

The 10Base-T/100Base-TX ports use RJ-45 connectors and can operate in the following modes:

- 10Base-T full-duplex mode
- 10Base-T half-duplex mode
- 100 Base-TX full-duplex mode
- 100Base-TX half-duplex mode

- Auto-sensing mode

Half-duplex mode uses back pressure flow control to prevent the receiving buffer from being overrun by data from a source node. Full-duplex mode uses the 802.3X flow control standard to prevent fast data traffic from over-running slow data traffic. Auto-sensing mode automatically determines whether full-duplex or half-duplex mode is used after auto-negotiating with the other end of the link.

- When configured for 10Base-T operation, the ports are ideal for connection to single endstations, 10Base-T hubs, or any 10Base-T-compatible device that uses standard 10Base-T adapters and wiring. 10Base-T ports are configured as MDIX and provide a full 10 Mbps bandwidth to attached devices. Maximum segment length is 100 meters (328 feet) over grade 3, 4 or 5 twisted-pair cable.
- When configured for 100Base-TX operation, the ports are ideal for connection to server or network backbones. 100Base-TX ports are configured as MDIX and provide 100 Mbps bandwidth to attached devices. Maximum segment length is 100 meters (328 feet) over grade 5 twisted-pair cable.

1000Base-T Ports

1000Base-T ports use dual RJ-45 connectors to provide links to high-speed network segments or individual workstations.

LEDs and Connectors

Front Panel LEDs

- 24 100Base-TX Link LEDs
- 24 100Base-TX Activity LEDs
- 2 1000BaseT Link LEDs
- 2 1000BaseT Activity LEDs
- Hot Swap LED
- Ok-to-Pull LED (Bellcore GR2914)

Power LED (green)

The green Power (PWR) LED is activated when the Ethernet Switch is recognized 3.3V.

Pull LED (red)

The red Pull (PULL) LED is not yet implemented.

Link LED (green)

The green Link (LNK) LED is activated when a port has established a connection with the host.

Activity LED (green)

The green Activity (ACT) LED indicates receive / transmission activity on that port.

Swap LED (blue)

The blue Hot-swap LED (SWP) is not yet implemented.

Front Panel Connectors

- DB9M (management processor console port)

Transition Card

- Single-slot with two high-density connectors. 24 100Base-T ports accessible through external breakout panel, and two 1000Base-T ports through RJ-45 connectors.

Operating Mechanical and Environmental

Electrical

- 16W of +3.3Vdc maximum power consumption

Mechanical

- CompactPCI 6U, 1 slot (4HP)
- 160mm x 233.35mm x 20mm

Temperature

- -5ºC to 55º (Operating)

Humidity

- 5% to 90% relative humidity, noncondensing

Altitude

- 3000m

Storage/Transit Environmental

Temperature

- -40ºC to 70ºC

Humidity

- 10% to 95% relative humidity, noncondensing

Altitude

- 10000m

Safety Compliance

- UL/cUL 1950 3rd Edition Recognized Component

Electromagnetic Compatibility (EMC)

- FCC Class A

Telco Compliance

- Designed for Bellcore NEBS GR-63-CORE Level 3
- Designed for Bellcore NEBS GR-1089 CORE Level 3

Marks

UL, cUL, CE

Basic Functions of Layer-2 Switching

If the <Destination MAC Address, Source MAC Address> of a receiving frame exists in the Switch's Distributed Flow Cache (DFC), the incoming frame is switched to the output port. Otherwise, the Switch is responsible for switching both VLAN tagged and untagged frames from a receiving port to one or more transmitting ports.

During the switching process, the Switch performs multiple steps, including:

- VLAN classification
- Learning

- Filtering
- Forwarding
- Aging

The following sections provide additional information about the tasks that the Switch performs during unicast and multicast switching.

Unicast Switching

The following sections describe VLAN classification, learning, filtering, and forwarding for unicast switching.

VLAN Classification

When the Switch receives a frame, it classifies the frame in one of two ways:

- If the frame is untagged, the Switch classifies the frame to an associated VLAN.
- If the frame is tagged, the Switch uses the tagged VLAN ID to identify the broadcasting domain of the frame.

Learning

After VLAN classification, the Switch checks the <source MAC address, VLAN> pair in the switching database (SDB) to see whether the <source MAC address, VLAN> pair is known.

- If <source MAC address, VLAN> is unknown, the Switch inserts the <source MAC address, VLAN> into the SDB and learns the <source MAC address, VLAN>.
- If <source MAC address, VLAN> is known, the Switch checks the <source MAC address, VLAN> pair for a mismatched port ID. If the port ID associated with the <source MAC address, VLAN> pair in the SDB is different than the receiving port, the Switch modifies the port ID in the SDB and modifies its management database (MDB) accordingly.

Filtering

After learning the address, the Switch checks:

- Whether the source port or destination port is in the forwarding state.
- The source MAC address or destination MAC address to be filtered.
- That the source port ID is the same as destination port ID.

If any of these conditions are met, the Switch drops the receiving. Otherwise, it continues with the forwarding process described below.

Forwarding

During the forwarding process, the Switch checks whether the <destination MAC address, VLAN> pair is unknown.

- If <destination MAC address, VLAN> is unknown, the Switch floods the receiving frame to all ports in the VLAN, excluding the source port.
- If <destination MAC address, VLAN> is known, the Switch forwards the receiving frame to the port associated with the <destination MAC address, VLAN> pair. At the same time, the Switch ascertains the individual port's VLAN tagging/untaggging configuration and corresponding VLAN ID to render the appropriate frame-tagging decisions when the frame is ready to be transmitted.

Multicast Switching

For a multicast switching, the Switch checks whether the received frame is a BPDU (Bridge Protocol Data Unit). If a BPDU is received, the Switch forwards the frame for processing by the Spanning Tree protocol. Otherwise, the Switch performs the following processes:

- VLAN classification — same as for unicast switching.
- Learning — same as for unicast switching.
- Filtering — after learning, the Switch checks:
  - Whether the source port or destination port is not in the forwarding state.
  - The source MAC address is to be filtered.
  - The source port ID is the same as destination port ID.

  If any of the above conditions are met, the Switch drops the receiving frame. Otherwise, the Switch performs the forwarding process.
- Forwarding — the Switch floods the received multicast frame to all ports within the VLAN, excluding the source port. At the same time, the Switch ascertains the individual port’s VLAN tagging/untaggging configuration and corresponding VLAN ID to render the appropriate frame-tagging decisions when the frame is ready to be transmitted.
- Aging — the Switch performs the aging process for the <MAC addresses, VLAN> pair in the switching database. Once a < MAC address, VLAN> pair is aged out, the SDB is modified.
- Spanning Tree — the Switch supports one Spanning Tree per bridged network.

VLAN

Virtual LANs (VLANs) are logical, independent workgroups within a network. These workgroups communicate as if they had a physical connection to the network. However, VLANs are not limited by the hardware constraints that physically connect traditional LAN segments to a network. As a result, VLANs can define a network into various logical configurations.

For example, VLANs can define a network by application. For instance, a company might create one VLAN for multimedia users and another for e-mail users. VLANs can also define a network by department. For example, a company might have one VLAN for its Engineering Department, another for its Marketing Department, and another for its Accounts Payable Department.

VLANs can also be set up according to the organization structure within a company. For example, the company president might have her own VLAN, her executive staff might have a different VLAN, and the remaining employees might have yet a different VLAN.

As these examples show, VLANs offer unparalleled flexibility. The following sections describe how deploying VLANs can benefit organizations and reduce administration costs.

Broadcast Containment

In traditional networks, traffic broadcasts to all network devices, whether they are the intended recipients or not. However, VLANs can be set up to contain only those devices that need to communicate with each other. As a result, VLANs significantly reduce network congestion In addition, VLANs prevent broadcast storms from causing a network meltdown due to volumes of traffic.

Multicast-Based Multimedia Applications

Multimedia applications, such as interactive training, video conferencing, and news-video transmissions, require large amounts of bandwidth. These applications are also extremely sensitive to variable delays, which are unavoidable on a shared Ethernet network. By defining a VLAN based on the IP multicast address for all subscribing members on the VLAN, sufficient bandwidth will be available for these applications, providing true multimedia on Ethernet.

Enhanced Security

Because VLANs are self-contained, only the devices within the same VLAN can communicate with each other. If a device in one VLAN wants to communicate with a device in another VLAN, the traffic must go through a router.

VLAN Membership

Continuous Computing's VLAN implementation allows:

- Up to 256 VLANs in one switch.
- VLANs across multiple switches by using explicit or implicit tagging and the GARP/GVRP protocol defined in IEEE 802.1p and 802.1Q.
- An end station's network interface card to belong to multiple VLANs.
- A switch port to be associated with multiple VLANs.

Definitions of VLAN Membership

Continuous Computing's VLAN implementation allows VLAN membership to be defined based on ports. Port-based VLANs are organized by physical port number. For example, switch ports 1, 2, 4, and 6 can be one VLAN, while ports 3, 5, 7, and 8 can be another VLAN. Broadcasts from server within each group would only go to the members of its own VLAN. This ensures that broadcast storms cannot cause a network meltdown due to volumes of traffic.

VLAN Membership Learning

Port-based VLAN is defined using a static binding between a VLAN and its associated ports. The 24+2 Ethernet Switch's forwarding decision is based on the destination MAC address and its associated port ID. Therefore, to make valid forwarding and flooding decisions, the 24+2 Ethernet Switch learns the relationship of the MAC address to its related port — and thus to the VLAN — at run-time.

Remote VLAN Learning

In addition to providing network management tools that allow network administrators to statically add and delete VLAN member ports, the 24+2 Ethernet Switch also supports GVRP (GARP VLAN Registration Protocol). GVRP allows for dynamic registration of VLAN port members within a switch and across multiple switches.

In addition to supporting the dynamic updating of registration entries in a switch, GVRP is used to communicate VLAN registration information to other VLAN-aware switches, so that a VLAN member can cover a wide span of switches in a network.

GVRP allows both VLAN-aware workstations and Continuous Computing switches to issue and revoke VLAN memberships. VLAN-aware

Continuous Computing switches register and propagate VLAN membership to all ports that belong to the active topology of the VLAN.

VLAN Configuration

Continuous Computing currently provides a Local/Remote Management Console Interface for VLAN configuration and management. An SNMP-based VLAN MIB is also provided.

Intra-VLAN Communication

The 24+2 Ethernet Switch supports intra-VLAN communication using ASICs.

Inter-VLAN Communication

The 24+2 Ethernet Switch supports inter-VLAN communication using CPU-based routing software.

Class-of-Service (CoS) Support

The 24+2 Ethernet Switch provides four transmit queues on each port, with a weighted round-robin scheme. These functions can be used to provide independent priorities for various types of data including real-time video, real-time voice, and best-effort data.

Priority assignment to a packet in Continuous Computing-based switches is accomplished through explicit assignment by end stations, which have applications that require a higher priority than best-effort data. This mechanism utilizes the IEEE 802.1p and 802.1Q tag structure, which the Switch uses to decide priority assignments for the received packets.

Quality-of-Service (QoS) Support

The Continuous Computing 24+2 Ethernet Switch supports IEEE 802.1p/Quality of Service with four priority transmission queues.

The eight levels of IEEE queues are statically mapped to the four available transmission queues in the following manner:

IEEE Q	24+2 Q	
0	0	(Lowest Priority Level)
1	0	
2	1	

3	1	
4	2	
5	2	
6	3	
7	3	(Highest Priority Level)

Quality of Service (QoS) is normally used to allow traffic prioritization based on traffic type. A typical implementation in a Voice over IP installation might have control traffic tagged with QoS level 7, voice traffic tagged with QoS level 5, web traffic tagged with level 3, and email traffic tagged with level 1. The higher priorities assigned to control and voice traffic ensure that a spike in web or email traffic does not disrupt active voice calls that are travelling across the network.

The Continuous Computing 24+2 Ethernet Switch handles packets tagged with QoS bits in one of two ways, depending on port congestion. When an output port is heavily congested, meaning that the port does not have enough bandwidth to transmit all of the packets that have been sent to it, the switch acts in strict priority QoS mode. This means that the switch will transmit all of the highest priority traffic before moving on to the next QoS priority level.

In practice, this means that a single 100 Mbit stream of high-priority traffic can completely take over an output port and prevent any lower-priority traffic from being transmitted. This is in accordance with the desire to ensure that high priority traffic is not disturbed or limited by lower priority traffic.

The Switch acts in a different manner when a port is not congested. If a port is not congested, by definition it has enough bandwidth to transmit all of the packets that it receives, so no packets are lost or dropped. In this case, the QoS features of the Switch are intended to reduce latency through the Switch for higher priority traffic.

To accomplish this, the Switch uses a weighted round-robin algorithm when selecting packets for transmission. This algorithm ensures that high-priority traffic is transmitted quickly while allowing some lower priority traffic to be scheduled for delivery.

The algorithm provides that for every 4 packets of highest priority (QoS level 6 and 7) traffic, one packet of lower priority traffic (QoS level 4 and 5) may be sent. For every 4 packets of QoS level 4 and 5 traffic, one packet of Qos level 2 and 3 may be sent, and so on.

Future firmware releases will provide the ability for users to customize both the QoS mappings and the weighted round-robin parameters. Currently any changes must be done via a custom firmware image.

GVRP

In addition to network management tools that allow network administrators to statically add and delete VLAN member ports, the Continuous Computing Routing Switch supports GARP VLAN Registration Protocol (GVRP). GVRP supports the dynamic registration of VLAN port members within a switch and across multiple switches.

In addition to dynamically updating registration entries within a switch, GVRP is used to communicate VLAN registration information to other VLAN-aware switches, so that members of a VLAN can cover a wide span of switches in a network.

GVRP allows both VLAN-aware workstations and Continuous Computing switches to issue and revoke VLAN memberships. VLAN-aware Continuous Computing switches register and propagate VLAN membership to all ports that are part of the active topology of the VLAN.

DHCP

Dynamic Host Configuration Protocol (DHCP), described in RFC 1541, is an extension of the Bootstrap Protocol (BOOTP). DHCP allows hosts on a TCP/IP network to dynamically obtain basic configuration information. When a DHCP client starts, it broadcasts a DHCP Request packet, looking for DHCP servers. DHCP servers respond to this packet with a DHCP Response packet. The client then chooses a server to obtain TCP/IP configuration information, such as its own IP address.

Since DHCP uses broadcast mechanism, a DHCP server and its client must physically reside on the same subnet. However, it’s not practical to have one DHCP server on every subnet; in fact in many cases, DHCP/BOOTP clients and their associated DHCP/BOOTP server(s) do not reside on the same IP network or subnet. In such cases, a third-party agent is required to transfer BOOTP messages between clients and servers.

BOOTP/DHCP Relay, described in RFC 1542, enables a host to use a BOOTP or DHCP server to obtain basic TCP/IP configuration information, even if the servers do not reside on the local subnet. When a Continuous Computing Routing Switch with BOOTP/DHCP Relay Agent receives a DHCP Request packet destined for a BOOTP/DHCP server, it inserts its own IP address into the DHCP Request packet so the server knows the subnet where the client is located. Then, depending on the configuration setup, the Switch either:

- Forwards the packet to a specific server as defined in the Switch's configuration using unicast routing, or
- Broadcasts the DHCP Request again to another directly attached IP subnet specified in the Switch configuration for the receiving IP subnet.

When the DHCP server receives the DHCP request, it allocates a free IP address for the DHCP client from its scope in the DHCP client's subnet, and sends a DHCP Response back to the DHCP Relay Agent. The DHCP Relay Agent then broadcasts this DHCP Response packet received from the DHCP server to the appropriate client.

ICMP Router Discovery

Before a host can send IP datagrams beyond its directly attached subnet, the host must discover the address of at least one operational router on that subnet. Typically, this is accomplished by reading a list of one or more router addresses from a configuration file at start-up time. On multicast links, some hosts also discover router addresses by listening to routing protocol traffic.

ICMP Router Discovery message is an alternative router discovery method that use a pair of ICMP messages on multicast links. It eliminates the need to manually configure router addresses and is independent of any specific routing protocol.

ICMP Router Discovery messages are called "Router Advertisements" and "Router Solicitations." Each router periodically multicasts a Router Advertisement from each of its multicast interfaces, announcing the IP address(es) of that interface. Hosts discover the addresses of their neighboring routers simply by listening for advertisements. When a host attached to a multicast link starts up, it may multicast a Router Solicitation to ask for immediate advertisements, rather than waiting for the subsequent, periodic ones to arrive.

Router Discovery messages do not constitute a routing protocol: they enable hosts to discover the existence of neighboring routers, but not which router is best to reach a particular destination. If a host chooses a poor first-hop router for a particular destination, it should receive an ICMP Redirect from that router, identifying a better one.

IGMP Snooping and IP Multicast Filtering

The Internet Group Management Protocol (IGMP) runs between hosts and their immediately neighboring multicast routers. The protocol's mechanisms allow a host to inform its local router that it wants to receive transmissions addressed to a specific multicast group.

Routers periodically query the LAN to determine if known group members are still active. If there is more than one router on the LAN performing IP multicasting, one of the routers is elected “querier” and assumes the responsibility of querying the LAN for group members.

Based on the group membership information learned from the IGMP, a router can determine which (if any) multicast traffic needs to be forwarded to each of its “leaf” subnetworks. Multicast routers use this information, along with a multicast routing protocol, to support IP multicasting across the Internet.

IGMP provides the final step in an IP multicast packet delivery service since it is only concerned with the forwarding of multicast traffic from the local router to group members on directly attached subnetworks.

Continuous Computing routing switches support IP Multicast Filtering by:

- Passively snooping on the IGMP Query and IGMP Report packets transferred between IP Multicast Routers and IP Multicast host groups to learn IP Multicast group members, and
- Actively sending IGMP Query messages to solicit IP Multicast group members.

The purpose of IP multicast filtering is to optimize a switched network’s performance, so multicast packets will only be forwarded to those ports containing multicast group hosts members and routers instead of flooding to all ports in the subnet (VLAN).

Continuous Computing routing switches with IP multicast filtering/ switching capability not only passively monitor IGMP Query and Report messages, DVMRP Probe messages, PIM, and MOSPF Hello messages; they also actively send IGMP Query messages to learn locations of multicast routers and member hosts in multicast groups within each VLAN.

Note, however, IGMP neither alters nor routes any IP multicast packets. Since IGMP is not concerned with the delivery of IP multicast packets across subnetworks, an external IP multicast router is needed if IP multicast packets have to be routed across different subnetworks.

2 Installation

This chapter describes how to install the 24+2 Ethernet Switch. Topics include:

- Electrostatic discharge (ESD)
- Installing the 24+2 Ethernet Switch
- Powering on the 24+2 Ethernet Switch
- Connecting the input/output devices
- Connectors, pinouts and specifications

Electrostatic Discharge (ESD)

Caution! The Ethernet Switch contains electronic components that are extremely sensitive to static electricity. Ordinary amounts of static may destroy components.

What to do:

- Use an antistatic mat
- Use an antistatic wrist or foot strap

Unpacking

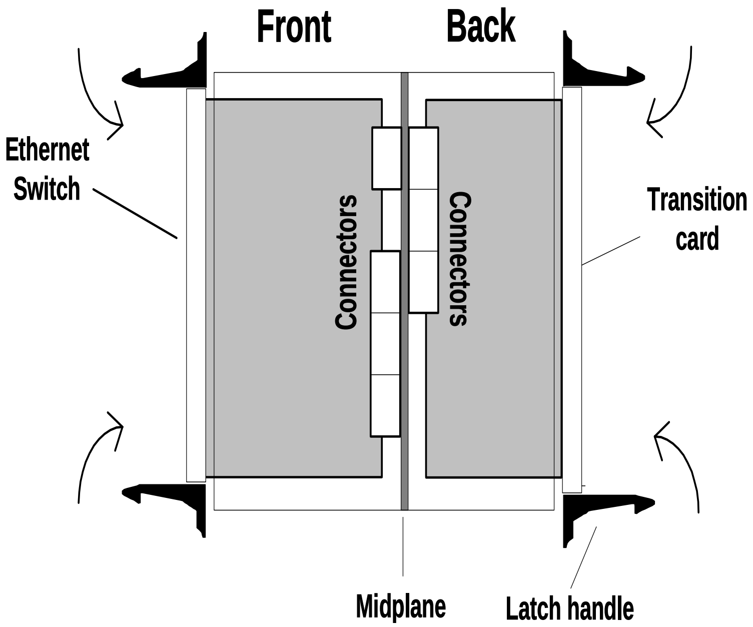
Caution! Always maintain an ESD-safe environment when handling the Ethernet Switch. It contains many components that can be destroyed by ESD.

- Inspect the shipping container for any in-transit damage and report it to shipping agent if necessary.
- Carefully unpack the system from its shipping container.

Installing the Ethernet Switch and Transition Card

1. Install Ethernet Switch into front of chassis.
2. Install transition card into rear of chassis.

See figure below for chassis positions of the Ethernet Switch and the transition card.



Ethernet Switch and transition card sideview

Installing the Switch and Card in the Chassis

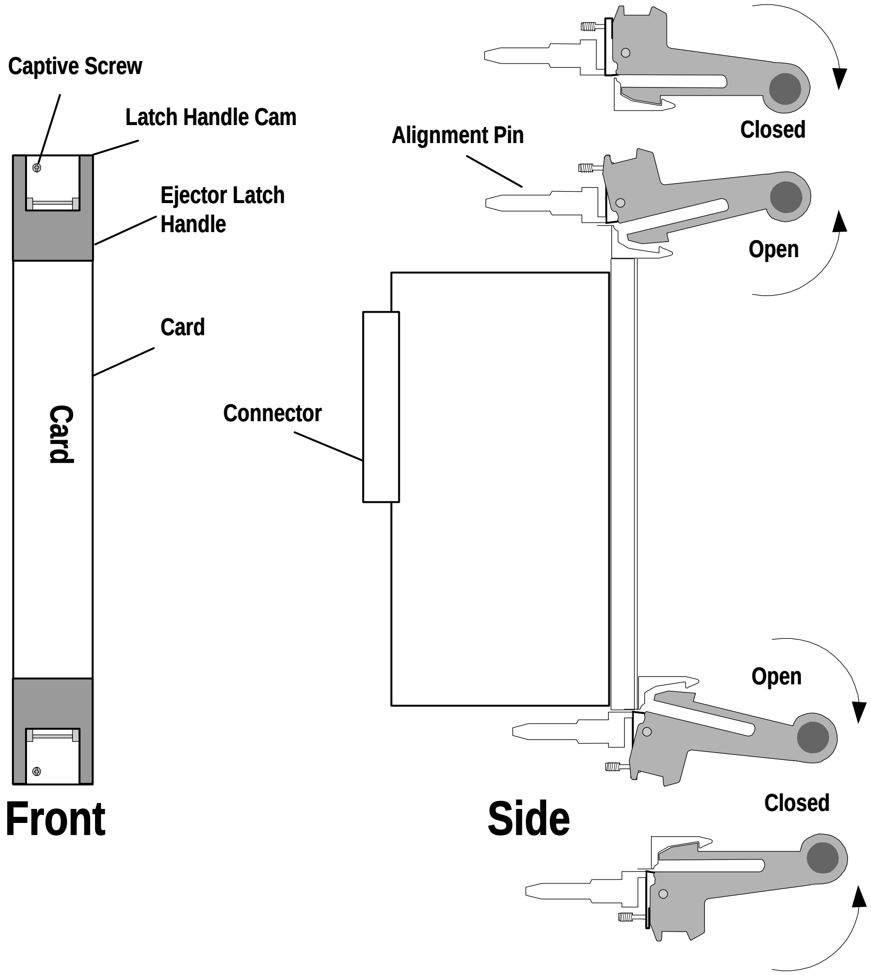
Caution! You cannot install an I/O card in the slot designated for a CPU card, or vice-versa.

1. Slide the card into its slot in the system chassis. As the card's ejector latches engage the chassis, apply forward pressure while pushing the ejector latch handles toward each other. This procedure applies to both the Ethernet Switch and transition card. See below for an illustration of Ethernet Switch and transition card installation and removal.

Note The transition card uses Type A/B connectors on J3 and J5 to ensure proper alignment, even in H.110 midplanes. J4 is not required for proper operation.

2. When properly installed, the connectors of each card will be fully engaged with the chassis's backplane. The Ethernet Switch's front panel will sit flush with the front panels of the other cards.

3. Install and tighten the captive screws supplied with the Ethernet Switch under each ejector latch handle to secure each card to the system chassis.



Card installation and removal

Powering on the CPCI 24+2 Ethernet Switch

When you power-on the CPCI 24+2 Ethernet Switch, it performs a Power-on Self Test (POST). During the POST, the Ethernet Switch performs a series of diagnostic procedures to make sure the basic system is functioning with integrity. The Switch then decompresses the run-time image and loads the image from the flash ROM into DRAM area. The system jump starts from this entry point.

If you press a key during the POST process, a menu prompts you with the following options:

Option	Description
Download Runtime Software from Serial Port	This option downloads the runtime system image to the Switch through the Switch's serial port. Before you select this option, make sure: • A host system is running a terminal emulation program that supports the Kermit file transfer protocol. • The host system's hard drive has the required binary file that will be downloaded to the Switch.
Configure the System	This option lets you modify any configurable parameter in the Switch's flash ROM before the Switch system boots.
Run Manufacturing Diagnostics	This option downloads the manufacturer's diagnostics. This option has the same download requirements as the runtime software apply here. When the file transfer is completed, the target system jumps to the entry point of the diagnostic program and starts executing the diagnostic code. The Main Menu of the diagnostic program appears, where you can initiate tests or obtain system information. Note that user intervention is not required when a test runs, unless an error occurs. If an error occurs during testing, you are given the choice of continuing the diagnostics or skipping the error.

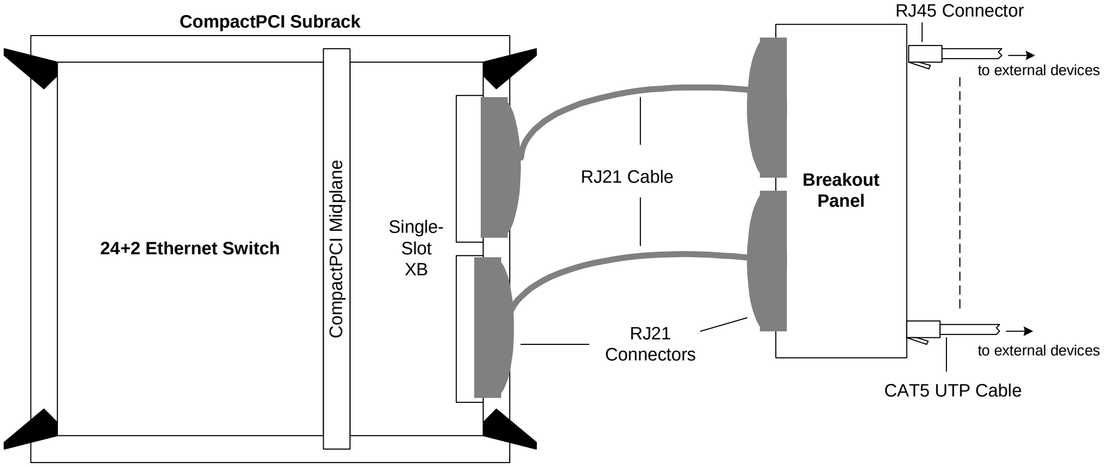
POST process options

Connecting the Input/Output Devices

Connecting the Network Devices

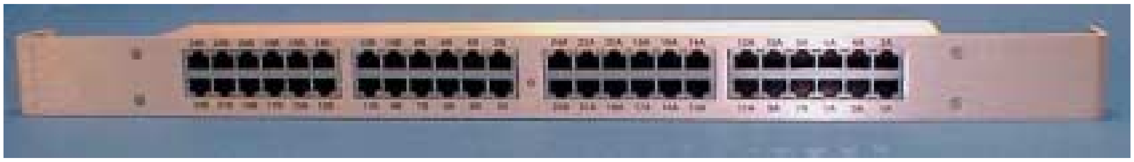
1. Plug in cables from the transition card to network devices.

- Use CAT5 UTP cable for 1000BaseT ports
- If connecting to host device, use straight-through cable
- If connecting to another switch or hub, use cross-over cable
- Use RJ-21 high-density cable to high-density breakout panel, then CAT5 UTP cables from the breakout panel to external devices (illustration below)



Connecting the breakout panel

Breakout panel (front). Use RJ-21 high-density cables to connect to transition card



Breakout panel (rear). Use RJ-45 cables to external devices

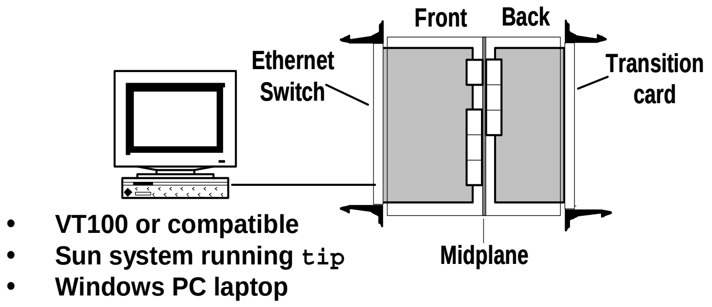
Breakout panel front

RJ-21 high-density cable connects to transition card

Connecting the Console Port

2. Connect to console.

- Use DB9 serial cable to terminal. This can be one of the following:
  - VT100 or compatible
  - Sun system running **tip**
  - Windows PC laptop with Hyperterm or another terminal program.
- Ensure the following proper settings:
  - Baud rate: **9600 bps**
  - Data bits: **8**
  - Parity: **none**
  - Stop bits: **1**
  - Flow control: **none**
- The Switch is configured as a DCE, so for most applications, a straight-through (not null-modem) cable should be correct.



Connecting the console

3. Login to console

At the screen prompt::

```
Continuous Computing 24+2 Ethernet Switch
System Name: switch_a

Console Login: admin
Password: 123456
```

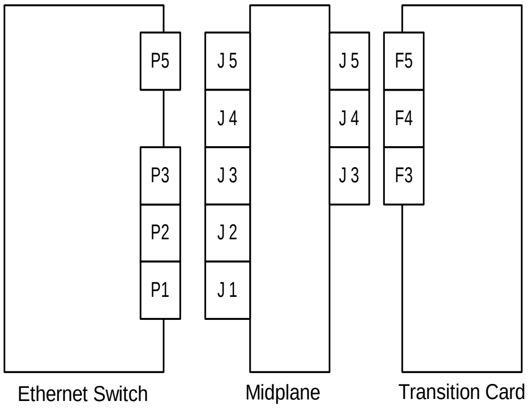
Examples of configuration:

- Change password
- Change system name
- Set IP address
- View port status

Refer to Chapter 3 for complete configuration information.Connectors, Pinouts and Specifications (Preliminary / Expected)

Connectors, Pinouts, and Specifications (Preliminary/Expected)

Connector Usage

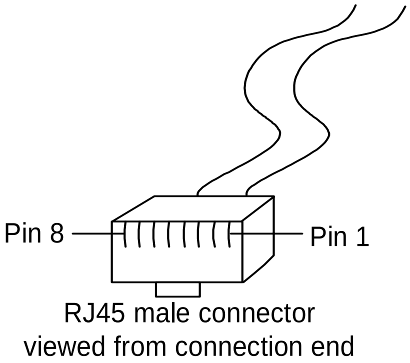


Connector Usage	
J1	PCI
J2	PCI
J3	10/100 Ethernet signals
J4	1000 Ethernet signals
J5	10/100 Ethernet signals

Connector usage

Pinouts

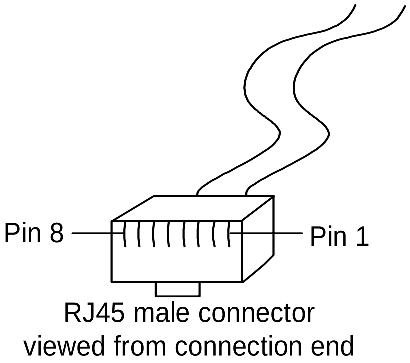
Splitter RJ45



PIN	Signal
1	RX+ _1
2	RX- _1
3	TX+ _1
4	RX+ _2
5	RX- _2
6	TX- _1
7	TX+ _2
8	TX- _2

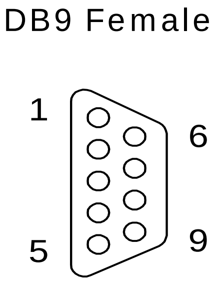
RJ45 male connector pinout

Standard RJ45



PIN	Signal
1	RX+
2	RX-
3	TX+
4	No Connect
5	No Connect
6	TX-
7	No Connect
8	No Connect

Standard RJ45 pinout



PIN	Signal	Signal DIR
1	DCD	OUTPUT
2	RXD	OUTPUT
3	TXD	INPUT
4	DTR	INPUT
5	GND	---
6	LBD	INPUT
7	RTS	INPUT
8	CTS	OUTPUT
9	No Connect	

*** The Switch DB9 pinout is DCE; when connecting to a PC or other DTE device, no null modem should be required.**

DB9 female pinout

Pin Assignments

Legend:

- SER_*NNN*: RS-232 console port signals
- MM*x*_TDP: 10/100Mb TX+ signal
- MM*x*_TDN: 10/100Mb TX- signal
- MM*x*_RDP: 10/100Mb RX+ signal
- MM*x*_RDN: 10/100Mb RX- signal
- GA*x*: geographical address signals
- NC: no connect

All other signals as defined by CompactPCI specifications

J5/P5 Connector

Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name
22A	TX19+	22B	TX19-	22C	FGND	22D		22E	
21A	RX19+	21B	RX19-	21C	FGND	21D		21E	
20A	TX18+	20B	TX18-	20C	FGND	20D		20E	
19A	RX18+	19B	RX18-	19C	FGND	19D		19E	
18A	TX17+	18B	TX17-	18C	FGND	18D		18E	
17A	RX17+	17B	RX17-	17C	FGND	17D		17E	
16A	TX16+	16B	TX16-	16C	FGND	16D		16E	
15A	RX16+	15B	RX16-	15C	FGND	15D		15E	
14A	TX15+	14B	TX15-	14C	FGND	14D		14E	
13A	RX15+	13B	RX15-	13C	FGND	13D		13E	
12A	TX14+	12B	TX14-	12C	FGND	12D		12E	
11A	RX14+	11B	RX14-	11C	FGND	11D		11E	
10A	TX13+	10B	TX13-	10C	FGND	10D		10E	
9A	RX13+	9B	RX13-	9C	FGND	9D		9E	
8A	TX12+	8B	TX12-	8C	FGND	8D		8E	
7A	RX12+	7B	RX12-	7C	FGND	7D		7E	
6A	TX11+	6B	TX11-	6C	FGND	6D		6E	
5A	RX11+	5B	RX11-	5C	FGND	5D		5E	
4A	TX10+	4B	TX10-	4C	FGND	4D		4E	
3A	RX10+	3B	RX10-	3C	FGND	3D		3E	
2A	TX9+	2B	TX9-	2C	FGND	2D		2E	
1A	RX9+	1B	RX9-	1C	FGND	1D		1E	

J4/P4 Connector

Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name
25A	TX20+	25B	TX20-	25C	FGND	25D		25E	
24A	RX20+	24B	RX20-	24C	FGND	24D		24E	
23A	TX21+	23B	TX21-	23C	FGND	23D		23E	
22A	RX21+	22B	RX21-	22C	FGND	22D		22E	
21A	TX22+	21B	TX22-	21C	FGND	21D		21E	
20A	RX22+	20B	RX22-	20C	FGND	20D		20E	
19A	TX23+	19B	TX23-	19C	FGND	19D		19E	
18A	RX23+	18B	RX23-	18C	FGND	18D		18E	
17A	TX24+	17B	TX24-	17C	FGND	17D		17E	
16A	RX24+	16B	RX24-	16C	FGND	16D		16E	
15A		15B		15C		15D		15E	
12-14	KEY AREA								
11A	MX1+_1	11B	MX1-_1	11C	FGND	11D	MX1+_3	11E	MX1-_3
10A	MX1+_0	10B	MX1-_0	10C	FGND	10D	MX1+_2	10E	MX1-_2
9A	MX2+_1	9B	MX2-_1	9C	FGND	9D	MX2+_3	9E	MX2-_3
8A	MX2+_0	8B	MX2-_0	8C	FGND	8D	MX2+_2	8E	MX2-_2
7A	LE_CLKO	7B	LE_DO	7C	DGND	7D	LE_SYNCO	7E	RESETLINK#
6A	LINK_LED	6B	ACT_LED	6C	DGND	6D	GIG_LED1	6E	GIG_LED2
5A	LED_S0#	5B	DGND	5C	LED_S1#	5D	DGND	5E	LED_S2#
4A	DGND	4B	+3.3V	4C	DGND	4D	+3.3V	4E	DGND
3A	TDI	3B	TDO	3C	TCK	3D	TMS	3E	TRST
2A	DGND	2B	TXD	2C	DGND	2D	RXD	2E	DGND
1A	CTS	1B	DTR	1C	NC	1D	RTS	1E	DSR

J3/P3 Connector

Pin #	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name
19A	SGA4	19B	SGA3	19C	SGA2	19D	SGA1	19E	SGA0
18A		18B		18C		18D		18E	
17A		17B		17C		17D		17E	
16A	TX8+	16B	TX8-	16C	FGND	16D		16E	
15A	RX8+	15B	RX8-	15C	FGND	15D		15E	
14A	TX7+	14B	TX7-	14C	FGND	14D		14E	
13A	RX7+	13B	RX7-	13C	FGND	13D		13E	
12A	TX6+	12B	TX6-	12C	FGND	12D		12E	
11A	RX6+	11B	RX6-	11C	FGND	11D		11E	
10A	TX5+	10B	TX5-	10C	FGND	10D		10E	
9A	RX5+	9B	RX5-	9C	FGND	9D		9E	
8A	TX4+	8B	TX4-	8C	FGND	8D		8E	
7A	RX4+	7B	RX4-	7C	FGND	7D		7E	
6A	TX3+	6B	TX3-	6C	FGND	6D		6E	
5A	RX3+	5B	RX3-	5C	FGND	5D		5E	
4A	TX2+	4B	TX2-	4C	FGND	4D		4E	
3A	RX2+	3B	RX2-	3C	FGND	3D		3E	
2A	TX1+	2B	TX1-	2C	FGND	2D		2E	
1A	RX1+	1B	RX1-	1C	FGND	1D		1E	

J2/P2 Connector

Pin #	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin#	Signal Name
22A	GA4	22B	GA3	22C	GA2	22D	GA1	22E	GA0
21A		21B	GND	21C		21D		21E	
20A		20B	GND	20C		20D	GND	20E	
19A	GND	19B	GND	19C		19D		19E	
18A		18B		18C		18D	GND	18E	
17A		17B	GND	17C		17D		17E	
16A		16B		16C		16D	GND	16E	
15A		15B	GND	15C		15D		15E	
14A		14B		14C		14D		14E	
13A		13B		13C		13D		13E	
12A		12B		12C		12D		12E	
11A		11B		11C		11D		11E	
10A		10B		10C		10D		10E	
9A		9B		9C		9D		9E	
8A		8B		8C		8D		8E	
7A		7B		7C		7D		7E	
6A		6B		6C		6D		6E	
5A		5B		5C		5D		5E	
4A		4B		4C		4D		4E	
3A		3B		3C		3D		3E	
2A		2B		2C		2D		2E	
1A		1B		1C		1D		1E	

J1 Connector

Pin #	Signal Name	Pin#	Signal Name	Pin#	Signal Name	Pin #	Signal Name	Pin #	Signal Name
25A	EARLY_5V	25B		25C		25D	EARLY_3.3V	25E	EARLY_5V
24A		24B	EARLY_5V	24C	LONG_I/O	24D		24E	
23A	EARLY_3.3V	23B		23C		23D	LONG_5V	23E	
22A		22B	GND	22C	LONG_3.3V	22D		22E	
21A	EARLY_3.3V	21B		21C		21D		21E	
20A		20B	GND	20C	EARLY_I/O	20D		20E	
19A	EARLY_3.3V	19B		19C		19D	GND	19E	
18A		18B	GND	18C	EARLY_3.3V	18D		18E	
17A	EARLY_3.3V	17B	IPMB_SCL	17C	IPMB_SDA	17D	GND	17E	
16A		16B	GND	16C	EARLY_I/O	16D		16E	
15A	EARLY_3.3V	15B		15C		15D	BD_SEL	15E	
12-14	KEY AREA								
11A		11B	NC	11C		11D	GND	11E	
10A		10B	GND	10C	EARLY_3.3V	10D		10E	
9A		9B		9C		9D	GND	9E	
8A		8B	GND	8C	EARLY_I/O	8D		8E	
7A		7B		7C		7D	GND	7E	
6A		6B	GND	6C	LONG_3.3V	6D		6E	
5A		5B		5C	PCI_RST#	5D	GND	5E	
4A	IPMB_PWR	4B	HEALTHY	4C	LONG_I/O	4D		4E	
3A		3B		3C		3D	LONG_5V	3E	
2A		2B	EARLY_5V	2C		2D		2E	
1A	EARLY_5V	1B	-12V	1C		1D	+12V	1E	EARLY_5V

3 CLI Commands

This chapter describes how to manage and configure the 24+2 Ethernet Switch using Command Line Interface (CLI) commands.

Local Console Management

You can manage the 24+2 Ethernet Switch locally by connecting a VT100 terminal, or a personal computer or workstation with terminal emulation software, to the 24+2 Ethernet Switch serial port. The terminal or workstation connects to the 24+2 Ethernet serial port using a straight-through cable that has the appropriate connectors on each end.

This management method is ideal when:

- The network is unreliable.
- The Network Manager does not have direct network connection.
- A Network Manager does not support SNMP.

The 24+2 Ethernet Switch's serial port's default setting is set to 9600 baud using a character format of 8 data bits, no parity, and 1 stop bit. Therefore, configure the terminal or workstation to use these settings before you log on to the 24+2 Ethernet Switch. You can change this default setting, if desired, after you log on.

Remote Console Management

You can manage the 24+2 Ethernet Switch remotely by having a remote host establish a Telnet connection to the 24+2 Ethernet Switch via an Ethernet or modem link.

Using this management method:

- The host must run a SLIP protocol if a modem is used.
- The 24+2 Ethernet Switch must have an Internet Protocol (IP) address.

The Remote Console Management interface is identical in appearance and functionality to the Local Console Management interface described in the previous section.

Logging on to the 24+2 Ethernet Switch

To log on to the 24+2 Ethernet Switch:

1. At the screen prompt:

```
Continuous Computing 24+2 Ethernet Switch Console

Login: admin

Password: 123456
```

Enter the console interface default console name (**admin**) and password (**123456**) or user-defined password if you changed the default password.

Note Only one console and five telnet users can log on to the 24+2 Ethernet Switch concurrently. However, multiple users should not modify the configuration at the same time.

The 24+2 CLI Commands (Firmware)

On the 24+2, a command-line console interface lets you set up the switch's parameters. The commands available at this interface are described in the following sections.

NOTE: Port number are from 1 to 26. Port 25 and port 26 corresponds to the GIGA bit ports.

Show Command

The **`show`** command is used to view the current settings of the switch. All the non-italics from the commands below represent tokens of the command. The options and their associated parameters are described in the paragraphs below. All parameters are required unless otherwise specified.

- **`show port status all`**

  `all` Shows all the port's status on the switch. This is optional and can be omitted. If it is omitted, all the port statuses are shown.

Example:

```
show port status all
```

- **`show port status <`*`port#`*`>`**

  `port#` Shows the status of a particular port.

Example:

```
show port status 1
```

- **`show rev`**

  `rev` Shows the hardware's and software's revision.

Example:

```
show rev
```

- **`show mac <`*`port#`*`>`**

  `port#` Shows all the MAC addresses on that particular port.

Example:

```
show mac 1
```

- **`show ip <`*`vlanid`*`>`**

  `vlanid` The ID of the VLAN to be searched, specified as a decimal value.

  This parameter is optional. If it is omitted, the default VLAN (VLAN ID 1) is used. Possible values are from 1 to 128.

Example:

```
show ip
show ip 3
```

- **`show err_stat port <`*`port#`*`>`**

  The **`show err_stat`** shows the error statistics based on these entries:

  1. Undersize frame count.
  2. CRC frame error count.
  3. Oversize frame count.
  4. Jabber frame count.

  `port#` The port number to collect the error statistics.

Example:

```
show err-stat port 1
```

- **`show gen_stat port <`*`port#`*`>`**

  The **`gen-stat`** shows the general statistics based on these entries:

  (**Note:** A good byte is from a good frame and a bad byte is from a bad frame.)

  1. Total bytes received.

2. Total frames received.
3. Broadcast frames received.
4. Multicast frames received.
5. Good bytes received.
6. Good frames received.

port# The port number to collect the general statistics.

Example:

```
show gen-stat port 1
```

- **show local mac**

  Shows the local MAC address of the switch itself. It has no parameters.

Example:

```
24+2 CLI> show local mac
  local MAC address
  0002bb000015
```

Upload Command

The **upload** command offers two ways of transferring files to the switch either by using Kermit or TFTP. The options and their associated parameters are described in the paragraphs below. All parameters are required unless otherwise specified.

- **upload kermit**

  kermit Starts a download process using the Kermit protocol.

Example:

```
download kermit
```

- **upload tftp <*ip_address filename*>**

  tftp Starts a download process using the TFTP protocol.

  ip_address The IP address, specified in standard decimal notation, of the TFTP server.

  filename The name of the file to be downloaded including its path.

Example:

```
upload tftp 172.17.2.3 firmware.ram
upload tftp 172.17.3.3 /home/welcome/firmware.ram
```

Set Command

The **set** command is used to set the value of various configurable options. The first parameter supplied on the command line identifies the option to be set and the remaining parameters vary depending on the specified option. The options and their associated parameters are described in the paragraphs below. All parameters are required unless otherwise specified.

- **set mac learning *<number/none/unlimited>* port *<port#>***

  port# The port to set. A decimal number from 1 to 26.

  number Decimal number from 0 to 15. A zero decimal number also means no MAC learning.

  none Sets MAC learning to none

  unlimited Sets MAC learning to unlimited.

Example:

```
set mac learning 1 port 3
set mac learning none port 4
set mac learning unlimited port 5
```

- **set mac address *<mac_address>* port *<port#>***

  mac_address MAC address to be assigned to the specified port. The address is entered as six two-digit hexadecimal values. (see example below).

  port# The port number to be configured, entered as a decimal value in the range 1 – 26. The port parameter is optional. If it is omitted, the MAC address is assigned to port #1.

Example:

```
set mac address 00013e0b45d0 port 3
```

- **set port speed *<speedtype>* port *<port#>***

  port# The port number to be configured, entered as a decimal value in the range of 1 to 25.

  speedtype The values can be 10, 100, or 1000. Only port 25 and 26 can be set to 1000. The rest takes a value of either 10 or 100.

Example:

```
set port speed 10 port 1
set port speed 100 port 3
set port speed 1000 port 25
```

- **set port duplex <*duplextype*> port <*port#*>**

  port# The port number to be configured, entered as a decimal value in the range of 1 to 26.

duplextype The values can be **half** for half duplex or **full** for full duplex.

Example:

```
set port duplex half port 3
```

- **set flood <*number/none/unlimited*>**

number Decimal value from 1 to 2422. Larger than 2422 automatically sets the limit to unlimited.

none Sets the flood limit to none.

unlimited Sets the flood limit to unlimited.

Example:

```
set flood 10
set flood none
set flood unlimited
```

- **set ip <*address netmask vlanid*>**

address The IP address for the VLAN identified by ***vlanid***. The IP address should be in standard decimal dot notation.

netmask Subnetwork mask for this IP address, specified in standard decimal Dot notation.

vlanid The ID of the VLAN to be configured, specified as a decimal value. This parameter is optional. If it is omitted, the default VLAN (VLAN ID 1) is configured. The vlanid range from 1 – 128.

Example:

```
set ip 172.17.2.1 255.255.0.0
set ip 172.17.2.2 255.255.0.0 3
```

- **set gateway <*netaddress gateaddress*>**

netaddress The IP address of the target network, in standard decimal dot format.

gateaddress The IP address of the gateway to the target network, in standard decimal dot format.

Example:

```
set gateway 172.17.2.1 172.17.0.251
```

- **set port <*enable/disable*> port <*port#*>**

port# The port number to be enabled or disabled.

enable Enables the port.

disable Disables the port.

Example:

```
set port enable port 4
set port disable port 3
```

- **set trap ip <*ip_address*> trap <*trap#*>**

  trap# A decimal number from 1 to 5. Right now, the switch can support up to 5 traps.

  ip_address The IP address of the target machine where the traps are collected. It is specified in standard decimal dot notation.

Example:

```
set trap ip 172.17.2.3 trap 1
set trap ip 172.17.2.3 trap 4
```

- **set local mac <*mac_addr*>**

  mac_addr The MAC address of the switch. The address is entered as a six two-digit hexadecimal values (see example below).

Example:

```
set local mac 0002bb00006d
set local mac 0002bb00006e
```

- **set trap cold_start <*enable/disable*>**

  enable Enables the cold start trap.

  disable Disables the cold start trap.

Example:

```
set trap cold_start enable
set trap cold_start disable
```

- **set trap warm_start *enable/disable***

  enable Enables the warm start trap.

  disable Disables the warm start trap.

Example:

```
set trap warm_start enable
set trap warm_start disable
```

- **set trap link_down <*enable/disable*>**

  enable Enables the link down trap.

  disable Disables the link down trap.

Example:

```
set trap link_down enable
set trap link_down disable
```

- **set trap link_up <*enable/disable*>**

  enable Enables the link up trap.

  disable Disables the link up trap.

Example:

```
set trap link_up enable
set trap link_up disable
```

- **set trap authentication_failure <*enable/disable*>**

  enable Enables the authentication failure trap.

  disable Disables the authentication failure trap.

Example:

```
set trap authentication_failure enable
set trap authentication_failure disable
```

- **set trap rising_alarm <*enable/disable*>**

  enable Enables the raising alarm trap.

  disable Disables the raising alarm trap.

Example:

```
set trap rising_alarm enable
set trap rising_alarm disable
```

- **set trap falling_alarm <*enable/disable*>**

  enable Enables the falling alarm trap.

  disable Disables the falling alarm trap.

Example:

```
set trap falling_alarm enable
set trap falling_alarm disable
```

- **set trap topology_change <*enable/disable*>**

enable Enables the topology change trap.

disable Disables the topology change trap.

Example:

```
set trap topology_change enable
set trap topology_change disable
```

- **set trap undersize <*enable/disable*>**

  enable Enables the undersize frame trap

  disable Disables the undersize frame trap.

Example:

```
set trap undersize enable
set trap undersize disable
```

- **set trap crc <*enable/disable*>**

  enable Enables the `crc` frame error trap.

  disable Disables the `crc` frame error trap.

Example:

```
set trap crc enable
set trap crc disable
```

- **set trap oversize <*enable/disable*>**

  enable Enables the oversize frame trap

  disable Disables the oversize frame trap

Example:

```
set trap oversize enable
set trap oversize disable
```

- **set trap jabber <*enable/disable*>**

  enable Enables the jabber frame (a continuous transmission of corrupted or random data) trap.

  disable Disables the jabber frame trap.

Example:

```
set trap jabber enable
set trap jabber disable
```

- **`set baud baud_rate`**

Sets the serial's baud rate. The correct baud rates are:

`9600, 19200, 38400, 57600, 115200,` and **`auto.`**

Example:

```
set baud 9600
set baud auto
```

Remove Command

The remove command removes an entry to one of the switch's table(s) .

- **`rm mac <address> <port#>`**

address	MAC address to be removed from a port.
port#	The port number where the MAC address to be removed resides.

Example:

```
rm mac 00013e0b45d0 3
```

Extra Commands

These extra commands are included so that you can have extra control to the switch's settings.

- **`set telnet <enable/disable>`**

enable	Enables telnet.
disable	Disables telnet.

Example:

```
set telnet enable
set telnet disable
```

- **`set snmp <enable/disable>`**

enable	Enables SNMP
disable	Disables SNMP

Example:

```
set snmp enable
set snmp disable
```

- **`clear`** Clears the entire screen.
- **`save setting`** Saves the current system settings

- **`reboot`** Reboots the switch.
- **`quit`** Returns to the log in prompt.
- **`Set username`** ***`<admin/guest username>`***

  `admin/guest` Sets either the admin's or the guest's username

  `username` The new user's name.

Example:

```
set username
```

- **`set password admin/guest`** ***`<password>`***

  `admin/guest` Sets either the admin's or the guest's password.

  `password` The new password.

Example:

```
set password admin 123456
set password guest 123456
```

- **`set system_name`** ***`<name>`***

  `name` The new name of the switch.

Example:

```
set system_name slug
```

- **`ping`** ***`<ip_address>`***

  `ip_address` IP address of the machine to ping. The IP address should be in standard decimal dot notation.

Example:

```
ping 172.17.2.1
```

- **`reset statistics all/port`** ***`<port#>`***

  (**Note:** You can either reset a particular port by using the keyword **`port`** with a port number or you can set all port counters by just using **`all`**)

  `port#` Indicates the port number to be reset.

Example:

```
reset statistics all
reset statistics port 3
```

The 24+2 CLI Commands (boot)

The 24+2 CLI boot has only a few commands. They include **`help`**, **`boot 1`**,

boot 2, and **rstcon**. Refer to the commands below for their usage.

boot [1 | 2]

Typing **boot 1** lets the user boot the switch and completely load the firmware. Typing **boot** is also the same as typing **boot 1**. Typing **boot 2** lets you go to the diagnostic screen of the switch. From there, you can see the following items:

```
Manufacturing Diagnostics Main Menu.
   1   e  (!) Executes The Selected Diagnostic Tests.
   2   z  (*) Switch Controller #0.
   3   o  (*) Switch Controller #1.
   4   x  (*) Switch Controller Tests (Xpipe And Multi-
              cast).
   5   c  (*) CPU / Peripherals Menu.
   6   g  (*) General Options.
   7   m  (*) Select Memory Testing Algorithms/Options.
   8   s  (!) Show Tests Presently Selected.
   9   d  (!) Show Default Manufacturing Tests.
  10   i  (!) Information About This System.
  11   t  (!) Execute Manufacturing Tests Using The
                Manufacturing Setup.
  12   a  (+) Enable All Manufacturing Tests.
  13   p  (*) Select The Ports To Test During SMI, Link
              And Loopback Tests.
  14   h  (!) Shows A Brief Help Screen.
  15   n  (*) Select Internal DS Memories To Test.
  16   l  (*) Show Test Log.
```

rstcon

The **rstcon** resets the console parameters to a previous system configuration. It then saves it to the flash.

Example:

```
24+2 Boot Cli> rstcon

Programming File to FLASH Memory...
```

upload

The upload command starts a Kermit download. You need will need a hyperterminal or a minicom program which is connected to the serial port in front of the switch to be able to download the switch's firmware.

Example:

```
24+2 Boot Cli> upload
```

help

The **help** command gives you the following output:

Example:

```
24+2 Boot Cli> help
boot [1|2]   : Switch Boot
rstcon       : Reset Console Parameters
upload       : Upload File
```

The 24+2 CLI Output (firmware)

The 24+2 CLI has the following return statuses. Please refer to the commands below to see which status belongs to a command. Any command issued on the CLI with just a command token will return a MISSING_ PARAM_ERR status. On the other hand, a word that does not belong to the command list will return command not found. The output would be: hello: command not found. When the command is successful, it returns SUCCESS.

```
SUCCESS
ERROR
MGMT_SET_ERR
MGMT_GET_ERR
UNKNOWN_CMD_ERR
INVALID_VLANID_ERR
INVALID_PORT_ERR
MISSING_PARAM_ERR
INVALID_SETTING_ERR
MAC_DOES_NOT_EXIST_ERR
INVALID_INDEX_ERR
TOO_LONG_USERNAME_ERR
TOO_LONG_PASSWORD_ERR
FLASH_UPDATE_ERR
TOO_LONG_SYSTEM_NAME_ERR
TIMEOUT_ERR
UNREACHABLE_HOST_ERR
NO_RESOURCES_ERR
INVALID_IP_ERR
```

```
INVALID_MASK_ERR
TOO_MANY_ARGUMENTS_ERR
UNKNOWN_TOKEN_ERR
INVALID_TRAP_NUMBER_ERR
INVALID_MAC_ADDRESS_ERR
```

Show Command

```
24+2 Cli> show
MISSING_PARAM_ERR
```

- **show {port status [all | *<port#>*]}**

Possible return statuses:

INVALID_PORT_ERR:

Example:

```
24+2 CLI> show port status 30
INVALID_PORT_ERR: 30
```

UNKNOWN_TOKEN_ERR:

Example:

```
24+2 CLI> show port some 30
UNKNOWN_TOKEN_ERR: some
```

TOO_MANY_ARGUMENTS_ERR:

Example:

```
24+2 CLI> show port status 2 hello
TOO_MANY_ARGUMENTS_ERR
```

SUCCESS:

Example:

```
24+2 CLI> show port status all
            Port 1:  Forwarding
            Port 2:  Disabled (Link Down)
            Port 3:  Disabled (Link Down)
```

```
          Port 4:  Disabled (Link Down)
          Port 5:  Disabled (Link Down)
          Port 6:  Disabled (Link Down)
          Port 7:  Disabled (Link Down)
          Port 8:  Disabled (Link Down)
          Port 9:  Disabled (Link Down)
          Port 10:  Disabled (Link Down)
          Port 11:  Disabled (Link Down)
          Port 12:  Disabled (Link Down)
          Port 13:  Disabled (Link Down)
          Port 14:  Disabled (Link Down)
          Port 15:  Disabled (Link Down)
          Port 16:  Disabled (Link Down)
          Port 17:  Disabled (Link Down)
          Port 18:  Disabled (Link Down)
          Port 19:  Disabled (Link Down)
          Hit return to continue
          Port 20:  Disabled (Link Down)
          Port 21:  Disabled (Link Down)
          Port 22:  Disabled (Link Down)
          Port 23:  Disabled (Link Down)
          Port 24:  Disabled (Link Down)
          Port 25:  Disabled (Link Down)
          Port 26:  Disabled (Link Down)
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show port status 1
Port 1:  Forwarding
```

- **show {rev}**

Possible return statuses:

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> show rev all
TOO_MANY_ARGUMENTS_ERR
```

SUCCESS:

Example:

```
24+2 CLI> show rev
Hardware Revision      :  300008 Rev. 3
Hardware Configuration :  c2m2g24.rom
Software Revision      :  v2.20f-Alcatel.001
Firmware Revision      :  v1.26a
```

- **show {mac [<*port#*>]}**

Possible return statuses:

```
INVALID_PORT_ERR:
24+2 CLI> show mac hello
INVALID_PORT_ERR: hello
```

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> show mac 28
INVALID_PORT_ERR: 28
```

```
TOO_MANY_ARGUMENTS_ERR
```

Example:

```
24+2 CLI> show mac 1 hello
TOO_MANY_ARGUMENTS_ERR
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show mac 2
```

`port 2 - 000000000001`

- **`show {ip [<`*`vlanid`*`>]}`**

Possible return statuses:

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> show ip 3 hello
TOO_MANY_ARGUMENTS_ERR
```

```
MGMT_GET_ERR:
```

Example:

```
24+2 CLI> show ip 2
MGMT_GET_ERR
```

```
INVALID_VLANID_ERR:
```

Example:

```
24+2 CLI> show ip 0
INVALID_VLANID_ERR: 0
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show ip
VLAN(1) - 172.17.2.1
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show ip 1
VLAN(1) - 172.17.2.1
```

- **show {[err_stat | gen_stat] port [<*port#*>] }**

Possible return statuses:

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> show err_stat port 0
INVALID_PORT_ERR: 0
```

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> show gen_stat port 0
INVALID_PORT_ERR: 0
```

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> show err_stat
MISSING_PARAM_ERR
```

```
MISSING_PARAM_ERR
```

Example:

```
24+2 CLI> show gen_stat
MISSING_PARAM_ERR
```

```
UNKNOWN_TOKEN_ERR:
```

Example:

```
24+2 CLI> show gen_stat hello 1
UNKNOWN_TOKEN_ERR: hello
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show err_stat port 1
            Error Statistics on port: #1
            Undersize frame count: 0
            CRC frame error count: 0
            Over size frame count: 0
            Jabber frame count:    0
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show gen_stat port 1
Port Statistics on port: #1
Total Bytes Received:    0
Total Frames Received:   0
Total Broadcast Packets Received: 0
Total Multicast Frames Received:  0
Total Good Bytes Received:      0
Total Good Frames Received:    0
```

- **show { local mac }**

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> show local
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> show local mac hello
TOO_MANY_ARGUMENTS_ERR
```

```
SUCCESS:
```

Example:

```
24+2 CLI> show local mac
    local MAC address
    0002bb000015
```

Upload Command

```
24+2 CLI> upload
MISSING_PARAM_ERR
```

- **upload { kermit }**

SUCCESS:

Example:

```
24+2 CLI> upload kermit
Start Kermit Send on terminal
Press Ctrl-C a few times to abort
Receiving...
Transfer Completed

Checking File Integrity

Programming File to FLASH
FLASH Memory Reprogrammed
You must reboot to execute the new image
```

TOO_MANY_ARGUMENTS_ERR:

Example:

```
24+2 CLI> upload kermit hello
TOO_MANY_ARGUMENTS_ERR
```

- **upload { tftp [<*ip_address*>] [<*filename*>] }**

MISSING_PARAM_ERR:

Example:

```
24+2 CLI> upload tftp
MISSING_PARAM_ERR


TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> upload tftp 172.17.0.250 vtx2080.rom helo
TOO_MANY_ARGUMENTS_ERR


INVALID_IP_ERR:
```

Example:

```
UNKNOWN_TOKEN_ERR:
```

Example:

```
24+2 CLI> upload hello 172.17.0.250 vtx2080.rom
UNKNOWN_TOKEN_ERR


TFTP Error:
```

Example:

```
24+2 CLI> upload tftp 172.17.2.1 vtx280.rom
Receiving...
LOAD_ERR_NO_SUCH_FILE


SUCCESS:
```

Example:

```
24+2 CLI> upload tftp 172.17.0.250 vtx2080.rom
Receiving...  -
Checking File Integrity

Programming File to FLASH
FLASH Memory Reprogrammed
You must reboot to execute the new image
```

Other return statuses:

```
LOAD_SUCCESS
LOAD_ERR_NO_SUCH_FILE
LOAD_ERR_FILE_TOO_BIG
LOAD_ERR_TFTP_ERR
LOAD_ERR_LOADING_IN_PROGRESS
LOAD_ERR_OPEN_FAILS
LOAD_ERR_NO_MEMORY
LOAD_ERR_UNREACHABLE_HOST
LOAD_ERR_NO_TFTP_SERVER
LOAD_ERR_TIMEOUT
```

Set Command

```
24+2 CLI> set
MISSING_PARAM_ERR
```

```
24+2 CLI> set sys_name
UNKNOWN_TOKEN_ERR: sys_name
```

- **set { mac learning [<*number* | *none* | *unlimited*>] port [<*port#*>] }**

Possible return statuses:

MISSING_PARAM_ERR:

Example:

```
24+2 CLI> set mac learning
MISSING_PARAM_ERR
```

TOO_MANY_ARGUMENTS_ERR:

Example:

```
24+2 CLI> set mac learning unlimited port 1 hello
TOO_MANY_ARGUMENTS_ERR
```

UNKNOWN_TOKEN_ERR:

Example:

```
24+2 CLI> set mac hello unlimited port 1
UNKNOWN_TOKEN_ERR: hello
```

INVALID_PORT_ERR:

Example:

```
24+2 CLI> set mac learning unlimited port 30
INVALID_PORT_ERR: 30
```

- **set { mac address [*<mac_address>*] port [*<port#>*] }**

Possible return statuses:

MISSING_PARAM_ERR:

Example:

```
24+2 CLI> set mac address
MISSING_PARAM_ERR
```

TOO_MANY_ARGUMENTS_ERR:

Example:

```
24+2 CLI> set mac address 1 port 3 hello
TOO_MANY_ARGUMENTS_ERR
```

INVALID_MAC_ADDRESS_ERR:

Example:

```
24+2 CLI> set mac address hello port 2
INVALID_MAC_ADDRESS_ERR: hello
```

```
UNKNOWN_TOKEN_ERR:
```

Example:

```
24+2 CLI> set mac hello 1 port 3
UNKNOWN_TOKEN_ERR: hello
```

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> set mac address 1 port 0
INVALID_PORT_ERR: 0
```

```
MGMT_SET_ERR:
```

Example:

```
MGMT_GET_ERR:
```

Example:

- **set { port speed [*<speedtype>*] port [*<port#>*] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set port speed
MISSING_PARAM_ERR
```

`TOO_MANY_ARGUMENTS_ERR:`

Example:

```
24+2 CLI> set port speed 100 port 3 hello
TOO_MANY_ARGUMENTS_ERR
```

`UNKNOWN_TOKEN_ERR:`

Example:

```
24+2 CLI> set port speed 100 hello 3
UNKNOWN_TOKEN_ERR: hello
```

`INVALID_PORT_ERR:`

Example:

```
24+2 CLI> set port speed 100 port 30
INVALID_PORT_ERR: 30
```

`INVALID_SETTING_ERR:`

Example:

```
24+2 CLI> set port speed 10 port 25
INVALID_SETTING_ERR: Port 25 & 26 are GIGABIT ports
```

`MGMT_GET_ERR:`

Example:

`MGMT_SET_ERR:`

Example:

- **set { port duplex [*<duplextype>*] port [*<port#>*] }**

Possible return statuses:

`MISSING_PARAM_ERR:`

Example:

```
24+2 CLI> set port duplex
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR
```

Example:

```
24+2 CLI> set port duplex half port 3 hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_SETTING_ERR
```

Example:

```
24+2 CLI> set port duplex half port 26
INVALID_SETTING_ERR: Port 25 & 26 are GIGABIT ports
```

```
INVALID_PORT_ERR
```

Example:

```
24+2 CLI> set port duplex half port 30
INVALID_PORT_ERR: 30
```

```
UNKNOWN_TOKEN_ERR
```

Example:

```
24+2 CLI> set port hello half port 3
UNKNOWN_TOKEN_ERR: hello
```

```
MGMT_GET_ERR
```

Example:

```
MGMT_SET_ERR
```

Example:

- **set { flood [*<number* | *none* | *unlimited>*] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set flood
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR
```

Example:

```
24+2 CLI> set flood 1 hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_SETTING_ERR
```

Example:

```
24+2 CLI> set flood hello
INVALID_SETTING_ERR: hello
```

```
MGMT_SET_ERR:
```

Example:

- **set { ip [*<address>*] [*<netmask>*] [*<vlanid>*] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set ip
MISSING_PARAM_ERR
```

```
INVALID_IP_ERR:
```

Example:

```
24+2 CLI> set ip 256.17.2.1 255.255.0.0 1
INVALID_IP_ERR: 256.17.2.1
```

```
MGMT_SET_ERR:
```

Example:

```
INVALID_MASK_ERR
```

Example:

```
24+2 CLI> set ip 172.17.2.1 256.255.0.0 1
INVALID_MASK_ERR: 256.255.0.0
```

```
INVALID_VLANID_ERR
```

Example:

```
24+2 CLI> set ip 172.17.2.1 255.255.0.0 0
INVALID_VLANID_ERR: 0
```

```
TOO_MANY_ARGUMENTS_ERR
```

Example:

```
24+2 CLI> set ip 172.17.2.1 255.255.0.0 1 hello
TOO_MANY_ARGUMENTS_ERR
```

- **set { gateway [*<netaddress>*] [*<gateaddress>*] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set gateway
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set gateway 172.17.2.1 172.17.0.251 hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_IP_ERR:
```

Example:

```
24+2 CLI> set gateway 256.17.2.1 172.17.0.251
INVALID_IP_ERR: 256.17.2.1
```

```
MGMT_SET_ERR:
```

Example:

- **set { port [*<enable | disable>*] port [*<port#>*] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set port
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set port enable port 1 hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> set port enable port 0
INVALID_PORT_ERR: 0
```

```
UNKNOWN_TOKEN_ERR:
```

Example:

```
24+2 CLI> set port hello port 0
UNKNOWN_TOKEN_ERR: hello
```

```
MGMT_SET_ERR
```

Example:

- **set { trap ip [<*ip_address*>] trap [<*trap#*>] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set trap ip
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set trap ip 172.17.2.2 trap 1 hello
TOO_MANY_ARGUMENTS_ERR
```

```
UNKNOWN_TOKEN_ERR:
```

Example:

```
24+2 CLI> set trap hello 172.17.2.2 trap 1
UNKNOWN_TOKEN_ERR: hello
```

```
INVALID_IP_ERR:
```

Example:

```
24+2 CLI> set trap ip 256.17.2.2 trap 1
```

```
INVALID_IP_ERR: 256.17.2.2
```

```
INVALID_TRAP_NUMBER_ERR:
```

Example:

```
24+2 CLI> set trap ip 172.17.2.2 trap hi
INVALID_TRAP_NUMBER_ERR: hi
```

```
MGMT_SET_ERR:
```

Example:

- **set { local mac [*<mac_addr>*] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set local mac
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set local mac 0002bb000015 hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_MAC_ADDRESS_ERR:
```

Example:

```
24+2 CLI> set local mac hello
INVALID_MAC_ADDRESS_ERR: hello
```

```
SUCCESS:
```

Example:

```
24+2 CLI> set local mac 0002bb000015
Saving the local MAC address to the flash
```

```
SUCCESS
```

```
FLASH_UPDATE_ERR:
```

Example:

- **set { trap [cold_start | warm_start | link_down | link_up | authentication_failure | rising_alarm | falling_alarm | topology_change | undersize | crc | oversize | jabber [<*enable* | *disable*>] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set trap
MISSING_PARAM_ERR
```

```
UNKNOWN_TOKEN_ERR:
```

Example:

```
24+2 CLI> set trap hello enable
UNKNOWN_TOKEN_ERR: hello
```

```
INVALID_SETTING_ERR:
```

Example:

```
24+2 CLI> set trap warm_start hello
INVALID_SETTING_ERR: hello
```

```
SUCCESS:
```

Example:

```
24+2 CLI> set trap warm_start enable
SUCCESS
```

- **set { [snmp | telnet] [<*enable* | *disable*>] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set telnet


MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set telnet enable hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_SETTING_ERR:
```

Example:

```
24+2 CLI> set telnet hello
INVALID_SETTING_ERR: hello
```

```
SUCCESS:
```

Example:

```
24+2 CLI> set telnet enable
SUCCESS
```

- **set { password [guest | admin] [<*password*>] }**

Possible return statuses:

MISSING_PARAM_ERR:

Example:

```
24+2 CLI> set password
MISSING_PARAM_ERR
```

TOO_MANY_ARGUMENTS_ERR

Example:

```
24+2 CLI> set password guest 123456 hello
TOO_MANY_ARGUMENTS_ERR
```

FLASH_UPDATE_ERR:

Example:

TOO_LONG_PASSWORD_ERR:

Example:

```
24+2 CLI> set password guest 12345678901234567890
TOO_LONG_PASSWORD_ERR: Password must be less than 17
         characters
```

UNKNOWN_TOKEN_ERR:

Example:

```
24+2 CLI> set password hello me
UNKNOWN_TOKEN_ERR: hello
```

SUCCESS:

Example:

```
24+2 CLI> set password guest me
Updating the GUEST PASSWORD to flash
SUCCESS
```

- **set { user_name [guest | admin] [*<user_name>***

```
          ] }
```

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set user_name
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR
```

Example:

```
24+2 CLI> set user_name guest guest hello
TOO_MANY_ARGUMENTS_ERR
```

```
TOO_LONG_USERNAME_ERR:
```

Example:

```
24+2 CLI> set user_name guest 12345678901234567890
TOO_LONG_USERNAME_ERR: User name must be less than 11
          characters
```

```
FLASH_UPDATE_ERR:
```

Example:

```
UNKNOWN_TOKEN_ERR
```

Example:

```
24+2 CLI> set user_name hello me
UNKNOWN_TOKEN_ERR: hello
```

```
SUCCESS:
```

Example:

```
24+2 CLI> set user_name guest me
```

```
Updating the GUEST USERNAME to flash
SUCCESS
```

- **set { system_name [<*name*>] }**

Possible return statuses:

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set system_name
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set system_name me hello
TOO_MANY_ARGUMENTS_ERR
```

```
SUCCESS:
```

Example:

```
24+2 CLI> set system_name captain_good
Updating SYSTEM NAME to flash
SUCCESS
```

- **set { baud [baud_rate] }**

```
Possible return statuses:

MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> set baud
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> set baud 9600 hello
TOO_MANY_ARGUMENTS_ERR
```

```
BAUD_RATE_ERR:
```

Example:

```
      24+2 CLI> set baud 2499
```

```
SUCCESS:
```

Example:

```
24+2 CLI> set baud 9600
SUCCESS
```

- **commands that are not found**

```
24+2 CLI> hello
hello:  command not found
```

Remove Command

```
24+2 CLI> rm
MISSING_PARAM_ERR
```

- **rm { mac [<*address*>] [<*port*>] }**

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> rm mac
```

```
MISSING_PARAM_ERR
```

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> rm mac 0 1 hello
TOO_MANY_ARGUMENTS_ERR
```

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> rm mac 0 0
INVALID_PORT_ERR: 0
```

```
INVALID_MAC_ADDRESS_ERR:
```

Example:

```
MGMT_SET_ERR:
```

Example:

```
MAC_DOES_NOT_EXIST_ERR:
```

Example:

```
24+2 CLI> rm mac 0 1
MAC_DOES_NOT_EXIST_ERR:
```

Ping Command

```
24+2 CLI> ping
MISSING_PARAM_ERR
```

- **ping { [*<ip_address>*] }**

Possible return statuses:

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> ping 172.17.2.1 hello
TOO_MANY_ARGUMENTS_ERR
```

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> ping
MISSING_PARAM_ERR
```

```
INVALID_IP_ERR:
```

Example:

```
24+2 CLI> ping me
INVALID_IP_ERR: me
```

```
SUCCESS:
```

Example:

```
24+2 CLI> ping 172.17.2.50
SUCCESS
```

```
UNREACHABLE_HOST_ERR:
```

Example:

```
NO_RESOURCES_ERR:
```

Example:

```
TIMEOUT_ERR:
```

Example:

```
24+2 CLI> ping 172.17.0.250
TIMEOUT_ERR
```

Reset Command

```
24+2 CLI> reset
MISSING_PARAM_ERR
```

- **reset { statistics [all] | { [port] *<port#>* } }**

Possible return statuses:

```
TOO_MANY_ARGUMENTS_ERR:
```

Example:

```
24+2 CLI> reset statistics all me
TOO_MANY_ARGUMENTS_ERR
```

```
MISSING_PARAM_ERR:
```

Example:

```
24+2 CLI> reset statistics
MISSING_PARAM_ERR
```

```
SUCCESS:
```

Example:

```
24+2 CLI> reset statistics all
SUCCESS
```

```
INVALID_PORT_ERR:
```

Example:

```
24+2 CLI> reset statistics port 27
INVALID_PORT_ERR: 27
```

Extra Commands

- **`clear`**

Possible return statuses:

```
NONE:
```

- **`save setting`**

Possible return statuses:

```
NONE:
```

- **`reboot`**

Possible return statuses:

```
NONE:
```

- **`quit`**

Possible return statuses:

```
NONE:
```

The 24+2 CLI Output (Boot)

The CLI Output from the boot prompt are shown below. If the command is not supported, the output results to `Invalid Command`.

```
24+2 Boot CLI> hello
```

```
Invalid Command
```

Help

Possible return statuses:

```
24+2 Boot CLI> help
        boot [1|2]          : Switch Boot
        rstcon              : Reset Console Parameters
        upload              : Upload File
```

Upload

Possible return statuses:

```
24+2 Boot CLI> upload
File Upload Completedow...
Checking File Integrity... Done
Programming File to FLASH Memory...
Done
```

boot [1 | 2]

Possible return statuses:

```
None:
```

rstcon

Possible return statuses:

```
24+2 Boot Cli> rstcon

Programming File to FLASH Memory...
```

The 24+2 CLI Command Index (firmware)

Show Command

- **show { port status [all | *<port#>*] }**
- **show { rev }**
- **show { mac [*<port#>*] }**
- **show { ip [*<vlanid>*] }**
- **show { [err_stat | gen_stat] port [*<port#>*] }**
- **show { local mac }**

Upload Command

- **upload { kermit }**
- **upload { tftp [*<ip_address>*] [*<filename>*] }**

Set Command

- **set { mac learning [*<number | none | unlimited>*] port [*<port#>*] }**
- **set { mac address [*<mac_address>*] port [*<port#>*] }**
- **set { port speed [*<speedtype>*] port [*<port#>*] }**
- **set { port duplex [*<duplextype>*] port [*<port#>*] }**
- **set { flood [*<number | none | unlimited>*] }**

- set { ip [*<address>*] [*<netmask>*] [*<vlanid>*] }
- set { gateway [*<netaddress>*] [*<gateaddress>*] }
- set { port [*<enable* **|** *disable>*] port [*<port#>*] }
- set { trap ip [*<ip_address>*] trap [*<trap#>*] }
- set { local mac [*<mac_addr>*] }
- set { trap [cold_start | warm_start | link_down | link_up | authentication_failure | rising_alarm | falling_alarm | topology_change | undersize | crc | oversize | jabber] [*<enable* **|** *disable>*] }
- set { [snmp | telnet] [*<enable* **|** *disable>*] }
- set { password [guest | admin] [*<password>*] }
- set { user_name [guest | admin] [*<user_name>*] }
- set { system_name [*<name>*] }
- set { local mac [*<mac_address>*] }
- set { baud [baud_rate] }h

Remove Command

- rm { mac [*<address>*] [*<port>*] }

Ping Command

- ping { [*<ip_address>*] }

Reset Command

- reset { [all] | { [port] *<port #>* } }

Extra Commands

- **`clear`**
- **`save setting`**
- **`reboot`**
- **`quit`**

The 24+2 CLI Command Index (boot)

```
boot  { [1 | 2] }
rstcon
upload
help
```

4 SNMP and RMON Management

This chapter describes the 24+2 Ethernet Switch's Simple Network Management Protocol (SNMP) and Remote Monitoring (RMON) capabilities. Topics include:

- Overview
- SNMP Agent and MIB-2
- RMON MIB and Bridge MIB
- Vertex MIB

Overview

RMON is an abbreviation for the Remote Monitoring MIB (Management Information Base). RMON is a system defined by the Internet Engineering Task Force (IETF) document RFC 1757, which defines how networks can be monitored remotely.

RMONs typically consist of two components: an RMON probe and a management workstation:

- The RMON probe is an intelligent device or software agent that continually collects statistics about a LAN segment or VLAN. The RMON probe transfers the collected data to a management workstation on request or when a pre-defined threshold is reached.
- The management workstation collects the statistics that the RMON probe gathers The workstation can reside on the same network as the probe, or it can have an in-band or out-of-band connection to the probe.

The 24+2 Ethernet Switch provides RMON capabilities that allow network administrators to set parameters and view statistical counters defined in MIB-II, Bridge MIB, and RMON MIB. RMON activities are performed at a Network Management Station running an SNMP network management application with graphical user interface.

SNMP Agent and MIB-2 (RFC1213)

The SNMP Agent running on the Switch manager CPU is responsible for:

- Retrieving MIB counters from various layers of software modules according to the SNMP GET/GET NEXT frame messages.
- Setting MIB variables according to the SNMP SET frame message.
- Generating an SNMP TRAP frame message to the Network Management Station if the threshold of a certain MIB counter is reached or if other trap conditions (such as the following) are met:
  - Warm start
  - Cold start
  - Link up
  - Link down
  - Authentication failure
  - Rising alarm
  - Falling alarm
  - Topology change

MIB-2 defines a set of manageable objects in various layers of the TCP/IP protocol suites. MIB-2 covers all manageable objects from layer 1 to layer 4 and, as a result, is the major SNMP MIB supported by all vendors in the networking industry. The 24+2 Ethernet Switch supports a complete implementation of SNMP Agent and MIB-2.

RMON MIB (RFC 1757) and Bridge MIB (RFC 1493)

The 24+2 Ethernet Switch provides hardware-based RMON counters in the Switch chipset. The Switch manager CPU polls these counters periodically to collect the statistics in a format that complies with the RMON MIB definition.

RMON Groups Supported

The 24+2 Ethernet Switch supports the following RMON MIB groups defined in RFC1757:

- RMON Statistics Group — maintains utilization and error statistics for the Switch port being monitored.
- RMON History Group — gathers and stores periodic statistical samples from the previous Statistics Group.
- RMON Alarm Group — allows a network administrator to define alarm thresholds for any MIB variable. An alarm can be associated with Low Threshold, High Threshold, or both. A trigger can trigger an alarm

when the value of a specific MIB variable exceeds a threshold, falls below a threshold, or exceeds or falls below a threshold.

- RMON Event Group — allows a network administrator to define actions based on alarms. SNMP Traps are generated when RMON Alarms are triggered. The action taken in the Network Management Station depends on the specific network management application.

Bridge Groups Supported

The 24+2 Ethernet Switch supports the following four groups of Bridge MIB (RFC1493):

- The dot1dBase Group — a mandatory group that contains the objects applicable to all types of bridges.
- The dot1dStp Group — contains the objects that denote the bridge's state with respect to the Spanning Tree Protocol. If a node does not implement the Spanning Tree Protocol, this group will not be implemented. This group is applicable to any transparent only, source route, or SRT bridge that implements the Spanning Tree Protocol.
- The dot1dTp Group — contains objects that describe the entity's transparent bridging status. This group is applicable to transparent operation only and SRT bridges.
- The dot1dStatic Group — contains objects that describe the entity's destination-address filtering status. This group is applicable to any type of bridge which performs destination-address filtering.

Vertex Private MIB

The following illustrates the Vertex private MIB.

```
VERTEX-MIB DEFINITIONS ::= BEGIN

IMPORTS
        IpAddress
                FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212
        DisplayString, PhysAddress
                FROM RFC1213-MIB
        MODULE-IDENTITY, NOTIFICATION-TYPE
                FROM SNMPv2-SMI;


   ccpu MODULE-IDENTITY
       LAST-UPDATED "0106300000Z"
       ORGANIZATION "Continuous Computing Corpora-
tion"
       CONTACT-INFO
              "        Robert Cagle

                Postal: Continuous Computing Corpo-
ration
                        9380 Carroll Park Drive
                        San Diego, CA  92121
                        US

                   Tel: +1 858 882 8834
                   Fax: +1 858 777 3388

                E-mail: rcagle@ccpu.com"
       DESCRIPTION
              "The MIB module to describe the 24+2
Switch"
       ::= { iso(1) org(3) dod(6) internet(1) pri-
vate(4) enterprises(1) 7994 }

lanModule       OBJECT IDENTIFIER ::= { ccpu 1 }
wanModule       OBJECT IDENTIFIER ::= { ccpu 2 }
swBeat          OBJECT IDENTIFIER ::= { ccpu 3 }
traps    OBJECT IDENTIFIER ::= { ccpu 4 }

vniConsole     OBJECT IDENTIFIER ::= { lanModule 1 }
vniSystemAdmin OBJECT IDENTIFIER ::= { lanModule 2 }
vniSystem      OBJECT IDENTIFIER ::= { lanModule 3 }
vniPort        OBJECT IDENTIFIER ::= { lanModule 4 }
vniBridge      OBJECT IDENTIFIER ::= { lanModule 5 }
vniVLAN        OBJECT IDENTIFIER ::= { lanModule 6 }
vniVLANIP      OBJECT IDENTIFIER ::= { lanModule 7 }
vniProxyARP    OBJECT IDENTIFIER ::= { lanModule 8 }
vniIPRoute     OBJECT IDENTIFIER ::= { lanModule 9 }
vniRIP        OBJECT IDENTIFIER ::= { lanModule 10 }
vniSNMP       OBJECT IDENTIFIER ::= { lanModule 11 }
vniTFTP       OBJECT IDENTIFIER ::= { lanModule 12 }
vniDHCP       OBJECT IDENTIFIER ::= { lanModule 13 }
vniHTTP       OBJECT IDENTIFIER ::= { lanModule 14 }
vniIPAccess   OBJECT IDENTIFIER ::= { lanModule 15 }
vniMirror     OBJECT IDENTIFIER ::= { lanModule 16 }
vniTrunk      OBJECT IDENTIFIER ::= { lanModule 17 }
vniSDB        OBJECT IDENTIFIER ::= { lanModule 18 }
vniSDBMacCountOBJECT IDENTIFIER ::= { lanModule 19 }
vniIGMP       OBJECT IDENTIFIER ::= { lanModule 20 }
vniStack      OBJECT IDENTIFIER ::= { lanModule 21 }

swMaster        OBJECT IDENTIFIER ::= { swBeat   1 }
swSlave         OBJECT IDENTIFIER ::= { swBeat   2 }
```

```
undersizeFrameOBJECT IDENTIFIER ::= { traps 1 }
oversizeFrameOBJECT IDENTIFIER ::= { traps 2 }
crcFrameOBJECT IDENTIFIER ::= { traps 3 }
jabberFrameOBJECT IDENTIFIER ::= { traps 4 }


vniConsoleBaudRate  OBJECT-TYPE
    SYNTAX      INTEGER {
                    b9600(1),
                    b19200(2),
                    b38400(3),
                    b57600(4),
                    b115200(5),
                    bAuto(6)
              }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Supported baudrate"
    ::= { vniConsole 1 }

vniConsoleFlowControl   OBJECT-TYPE
    SYNTAX      INTEGER {
                    disabled(1),
                    rts-cts(2),
                    xon-xoff(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Flow Control Settings"
    ::= { vniConsole 2 }

vniConsoleModemControl  OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Modem Control Settings"
    ::= { vniConsole 3 }

vniConsoleModemSetupFlag    OBJECT-TYPE
    SYNTAX      INTEGER {
                    default(1),
                    custom(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Flag to tell whether to use default or custom
        modem setup string (defined below)"
    ::= { vniConsole 4 }

vniConsoleDefaultModemSetupString  OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..43))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Default Modem Setup String"
    ::= { vniConsole 5 }

vniConsoleCustomModemSetupString  OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..43))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Custom Modem Setup String"
    ::= { vniConsole 6 }

vniConsoleSLIPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "SLIP State"
    ::= { vniConsole 7 }

vniConsoleSLIPAddress   OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "SLIP (IP) Address"
    ::= { vniConsole 8 }

vniConsoleSLIPNetmask   OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "SLIP (IP) Network Mask"
    ::= { vniConsole 9 }

VniSystemAdminEntry ::=
    SEQUENCE {
        vniSystemAdminUserID
```

```
                INTEGER,
            vniSystemAdminPassword
                DisplayString,
            vniSystemAdminName
                DisplayString
        }

vniSystemAdminTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniSystemAdminEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
      "System Administration Configuration Settings
Table"
    ::= { vniSystemAdmin 1 }

vniSystemAdminEntry OBJECT-TYPE
    SYNTAX      VniSystemAdminEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
      "System Administration Configuration Settings
Entry"
    INDEX       { vniSystemAdminUserID }
    ::= { vniSystemAdminTable 1 }

vniSystemAdminUserID    OBJECT-TYPE
    SYNTAX      INTEGER {
                    admin(1),
                    guest(2)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "User ID"
    ::= { vniSystemAdminEntry 1 }

vniSystemAdminPassword   OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..17))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "User password"
    ::= { vniSystemAdminEntry 2 }

vniSystemAdminName   OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..17))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "User name"
    ::= { vniSystemAdminEntry 3 }

vniSystemStatCollect OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Statistics collection flag"
    ::= { vniSystem 1 }

vniSystemRebootOnError OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Reboot on Error flag"
    ::= { vniSystem 2 }

vniSystemTelnetLogin OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Remote Telnet login flag"
    ::= { vniSystem 3 }

vniSystemSTPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Spanning Tree Protocol functionality"
    ::= { vniSystem 4 }

vniSystemGVRPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "GARP VLAN Registration Protocol functional-
ity"
    ::= { vniSystem 5 }

vniSystemIGMPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    disabled(1),
                    passive(2),
                    active(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Internet Group Management Protocol function-
ality"
    ::= { vniSystem 6 }

vniSystemConfigOperation OBJECT-TYPE
    SYNTAX      INTEGER {
                    none(1),
                    loadDefault(2),
                    saveCurrent(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "System Configuration operations.  This is a
write-only
        variable.  On read, it will always return
none(1).  On loadDefault(2)
        write, it will reboot the system."
    ::= { vniSystem 7 }

vniSystemAutoSaveState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Automatic Configuration Save functionality"
    ::= { vniSystem 8 }

vniSystemAutoSaveDelay OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Automatic Configuration Save delay"
    ::= { vniSystem 9 }

vniSystemHardwareRev OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..31))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Hardware Revision"
    ::= { vniSystem 10 }

vniSystemSoftwareRev OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..31))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Software Revision"
    ::= { vniSystem 11 }

vniSystemFirmwareRev OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..31))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Firmware Revision"
    ::= { vniSystem 12 }

vniSystemHardwareCfg OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..31))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Hardware configuration"
    ::= { vniSystem 13 }

vniSystemReboot OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "System Warm Reboot Functionality"
    ::= { vniSystem 14 }

vniSystemResetRMONCount OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Reset RMON counters on all the ports. Please
refer to vniPortEntry group to reset specific port."
    ::= { vniSystem 15 }

vniSystemSplash OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..10))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Splash Text"
    ::= { vniSystem 16 }

vniSystemDump OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Stack and memory dumping feature on error."
    ::= { vniSystem 17 }

vniSystemDumpCoreCount OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
         "Number of System Dump Cores."
    ::= { vniSystem 18 }

vniSystemOkToPull OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
         "Enable or Disable the Ok To Pull LED"
    ::= { vniSystem 19 }


VniPortEntry ::=
    SEQUENCE {
        vniPortNo
            INTEGER,
        vniPortFuncType
            INTEGER,
        vniPortSpeedType
            INTEGER,
        vniPortSpeedConfig
            INTEGER,
        vniPortDupConfig
            INTEGER,
        vniPortFlowConfig
            INTEGER,
        vniPortSpeedStatus
            INTEGER,
        vniPortDupStatus
            INTEGER,
        vniPortFlowStatus
            INTEGER,
        vniPortLinkStatus
            INTEGER,
        vniPortPriority
            INTEGER,
        vniPortLearningLimit
            INTEGER,
        vniPortResetRMONCount
            INTEGER
    }

vniPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniPortEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Port table"
    ::= { vniPort 1 }

vniPortEntry OBJECT-TYPE
    SYNTAX      VniPortEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Port entry"
    INDEX       { vniPortNo }
    ::= { vniPortTable 1 }

vniPortNo OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Number"
    ::= { vniPortEntry 1 }
```

```
vniPortFuncType OBJECT-TYPE
    SYNTAX      INTEGER {
                    normal(1),
                    trunkGroup(2),
                    mirror(3),
                    stackPort(4)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Function Type"
    ::= { vniPortEntry 2 }

vniPortSpeedType OBJECT-TYPE
    SYNTAX      INTEGER {
                    type10M(1),
                    type100M(2),
                    typeMII(3),
                    type1G(4)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Speed Type"
    ::= { vniPortEntry 3 }

vniPortSpeedConfig OBJECT-TYPE
    SYNTAX      INTEGER {
                    auto(1),
                    speed10M(2),
                    speed100M(3),
                    speed1G(4)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Port Speed Configuration"
    ::= { vniPortEntry 4 }

vniPortDupConfig OBJECT-TYPE
    SYNTAX      INTEGER {
                    auto(1),
                    half(2),
                    full(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Port Duplexity Configuration"
    ::= { vniPortEntry 5 }
```

```
vniPortFlowConfig OBJECT-TYPE
    SYNTAX      INTEGER {
                    auto(1),
                    enabled(2),
                    disabled(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Port Flow Control Configuration"
    ::= { vniPortEntry 6 }

vniPortSpeedStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    none(1),
                    speed10M(2),
                    speed100M(3),
                    speed1G(4)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Speed Status"
    ::= { vniPortEntry 7 }

vniPortDupStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    none(1),
                    half(2),
                    full(3)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Duplexity Status"
    ::= { vniPortEntry 8 }

vniPortFlowStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    none(1),
                    enabled(2),
                    disabled(3)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Flow Control Status"
    ::= { vniPortEntry 9 }

vniPortLinkStatus OBJECT-TYPE
    SYNTAX      INTEGER {
```

```
                    up(1),
                    down(2)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Port Link Status"
    ::= { vniPortEntry 10 }

vniPortPriority OBJECT-TYPE
    SYNTAX      INTEGER(0..7)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
      "Default Priority for untagged frames received
by this port"
    ::= { vniPortEntry 11 }

vniPortLearningLimit OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Dynamic MAC Address learning limit
            0 for no learning
            -1 for unlimited learning"
    ::= { vniPortEntry 12 }

vniPortResetRMONCount OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "reset RMON counters on this port."
    ::= { vniPortEntry 13}

vniBridgeAgingTime OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Bridge Aging Time
             0 not allowed
            -1 for unlimited aging (no forgetting)"
    ::= { vniBridge 1 }

vniBridgeFloodLimit OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Flood Limit for all ports in packets per
second
             0 for no flooding
             -1 for unlimited flooding"
    ::= { vniBridge 2 }

VniBridgeFilterSrcEntry ::=
    SEQUENCE {
        vniBridgeFilterSrcMacAddress
            PhysAddress,
        vniBridgeFilterSrcStatus
            INTEGER
    }

vniBridgeFilterSrcTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniBridgeFilterSrcEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Source MAC address filtering table"
    ::= { vniBridge 3 }

vniBridgeFilterSrcEntry OBJECT-TYPE
    SYNTAX      VniBridgeFilterSrcEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "MAC Address filter table entry"
    INDEX       { vniBridgeFilterSrcMacAddress }
    ::= { vniBridgeFilterSrcTable 1 }

vniBridgeFilterSrcMacAddress OBJECT-TYPE
    SYNTAX      PhysAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Source MAC address for filtering table"
    ::= { vniBridgeFilterSrcEntry 1 }

vniBridgeFilterSrcStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    valid(1),
                    invalid(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Row status of the entry"
    ::= { vniBridgeFilterSrcEntry 2 }

VniBridgeFilterDestEntry ::=
    SEQUENCE {
        vniBridgeFilterDestMacAddress
            PhysAddress,
        vniBridgeFilterDestStatus
            INTEGER
    }

vniBridgeFilterDestTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF VniBridgeFilterDestEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Source MAC address filtering table"
    ::= { vniBridge 4 }

vniBridgeFilterDestEntry OBJECT-TYPE
    SYNTAX      VniBridgeFilterDestEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "MAC Address filter table entry"
    INDEX       { vniBridgeFilterDestMacAddress }
    ::= { vniBridgeFilterDestTable 1 }

vniBridgeFilterDestMacAddress OBJECT-TYPE
    SYNTAX      PhysAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Source MAC address for filtering table"
    ::= { vniBridgeFilterDestEntry 1 }

vniBridgeFilterDestStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    valid(1),
                    invalid(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Row status of the entry"
    ::= { vniBridgeFilterDestEntry 2 }

VniVlanEntry ::=
    SEQUENCE {
        vniVLANID
            INTEGER,
        vniVLANName
```

```
            DisplayString(SIZE(0..31)),
        vniVLANType
            INTEGER,
        vniVLANStaticPortMap
            OCTET STRING,
        vniVLANStaticForbiddenPortMap
            OCTET STRING,
        vniVLANStaticUntaggedPortMap
            OCTET STRING,
        vniVLANCurrentPortMap
            OCTET STRING,
        vniVLANCurrentUntaggedPortMap
            OCTET STRING,
        vniVLANCurrentRegisteredPortMap
            OCTET STRING,
        vniVLANStatus
            INTEGER
    }

vniVLANTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniVlanEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "VLAN Settings Table"
    ::= { vniVLAN 1 }

vniVLANEntry OBJECT-TYPE
    SYNTAX      VniVlanEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "VLAN Settings Entry"
    INDEX       { vniVLANID }
    ::= { vniVLANTable 1 }

vniVLANID OBJECT-TYPE
    SYNTAX      INTEGER(1..4094)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "VLAN ID (valid values 1-4094)"
    ::= { vniVLANEntry 1 }

vniVLANName OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..31))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "VLAN Name.  This can only be set during
creation of VLAN.  It
```

```
        cannot be modified in a later time."
    ::= { vniVLANEntry 2 }

vniVLANType OBJECT-TYPE
    SYNTAX      INTEGER {
                    local(1),
                    remote(2)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The method on how the VLAN ID is learned."
    ::= { vniVLANEntry 3 }

vniVLANStaticPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Ports that always belong to the VLAN."
    ::= { vniVLANEntry 4 }

vniVLANStaticForbiddenPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Ports that are forbidden from belonging to
the VLAN."
    ::= { vniVLANEntry 5 }

vniVLANStaticUntaggedPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Untagged ports of the VLAN."
    ::= { vniVLANEntry 6 }

vniVLANCurrentPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Ports currently belonging to the VLAN."
    ::= { vniVLANEntry 7 }

vniVLANCurrentUntaggedPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Untagged ports currently belonging to the
VLAN."
    ::= { vniVLANEntry 8 }

vniVLANCurrentRegisteredPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
       "Registered ports currently belonging to the
VLAN."
    ::= { vniVLANEntry 9 }

vniVLANStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    valid(1),
                    invalid(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Row status of the entry"
    ::= { vniVLANEntry 10 }

VniVlanIpEntry ::=
    SEQUENCE {
        vniVLANIpIndex
            INTEGER,
        vniVLANIpAddress
            IpAddress,
        vniVLANIpNetmask
            IpAddress,
        vniVLANIpFrameType
            INTEGER,
        vniVLANIpBootpState
            INTEGER
    }

vniVLANIPTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniVlanIpEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "VLAN IP Settings Table"
    ::= { vniVLANIP 1 }

vniVLANIPEntry OBJECT-TYPE
    SYNTAX      VniVlanIpEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "VLAN IP Settings Entry"
    INDEX       { vniVLANIpIndex }
    ::= { vniVLANIPTable 1 }

vniVLANIpIndex OBJECT-TYPE
    SYNTAX      INTEGER(1..4094)
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "VLAN ID (valid values 1-4094)"
    ::= { vniVLANIPEntry 1 }

vniVLANIpAddress OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "VLAN IP Address"
    ::= { vniVLANIPEntry 2 }

vniVLANIpNetmask OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "VLAN IP Network Mask"
    ::= { vniVLANIPEntry 3 }

vniVLANIpFrameType OBJECT-TYPE
    SYNTAX      INTEGER {
                    ethernetII(1),
                    ethernetSNAP(2),
                    ethernet8022(3),
                    ethernet8023(4)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Frame type of the selected VLAN"
    ::= { vniVLANIPEntry 4 }

vniVLANIpBootpState OBJECT-TYPE
    SYNTAX      INTEGER {
                    bootp(1),
                    disabled(2),
                    dhcp(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
```

```
        "BOOTP state"
    ::= { vniVLANIPEntry 5 }

VniProxyARPEntry ::=
    SEQUENCE {
        vniProxyARPIpAddr
            IpAddress,
        vniProxyARPState
            INTEGER
    }

vniProxyARPTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniProxyARPEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Proxy ARP Table"
    ::= { vniProxyARP 1 }

vniProxyARPEntry OBJECT-TYPE
    SYNTAX      VniProxyARPEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Proxy ARP Entry"
    INDEX       { vniProxyARPIpAddr }
    ::= { vniProxyARPTable 1 }

vniProxyARPIpAddr OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Proxy ARP IP Address"
    ::= { vniProxyARPEntry 1 }

vniProxyARPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Proxy ARP State"
    ::= { vniProxyARPEntry 2 }

VniIPRouteEntry ::=
    SEQUENCE {
        vniIPRouteAddress
            IpAddress,
        vniIPRouteNetmask
            IpAddress,
        vniIPRouteGateway
            IpAddress,
        vniIPRouteMetric
            INTEGER,
        vniIPRouteStatus
            INTEGER
    }

vniIPRouteTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniIPRouteEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Table"
    ::= { vniIPRoute 1 }

vniIPRouteEntry OBJECT-TYPE
    SYNTAX      VniIPRouteEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Entry"
   INDEX     { vniIPRouteAddress, vniIPRouteNetmask
}
    ::= { vniIPRouteTable 1 }

vniIPRouteAddress OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Entry Address"
    ::= { vniIPRouteEntry 1 }

vniIPRouteNetmask OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Entry Netmask"
    ::= { vniIPRouteEntry 2 }

vniIPRouteGateway OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Entry Gateway"
    ::= { vniIPRouteEntry 3 }
```

```
vniIPRouteMetric OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Entry Metric"
    ::= { vniIPRouteEntry 4 }

vniIPRouteStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    valid(1),
                    createRequest(2),
                    underCreation(3),
                    invalid(4)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Routing Entry Row Status"
    ::= { vniIPRouteEntry 5 }

VniRIPEntry ::=
    SEQUENCE {
        vniRIPIndex
            INTEGER,
        vniRIPSetting
            INTEGER,
        vniRIPTxDestAddress
            INTEGER,
        vniRIPTxRoutes
            INTEGER,
        vniRIPTxDefaultRoute
            INTEGER,
        vniRIPRxSetting
            INTEGER,
        vniRIPRxDefaultRoute
            INTEGER,
        vniRIPSplitHorizon
            INTEGER,
        vniRIPPoisonedReverse
            INTEGER,
        vniRIPTriggeredResponse
            INTEGER
    }

vniRIPTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniRIPEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
```

```
        "Routing Information Protocol Table"
    ::= { vniRIP 1 }

vniRIPEntry OBJECT-TYPE
    SYNTAX      VniRIPEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Routing Information Protocol Entry"
    INDEX       { vniRIPIndex }
    ::= { vniRIPTable 1 }

vniRIPIndex OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Index value for RIP Table"
    ::= { vniRIPEntry 1 }

vniRIPSetting OBJECT-TYPE
    SYNTAX      INTEGER {
                    version1(1),
                    version2(2),
                    disabled(3)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "RIP settings.  vniRIPSetting version1 is 
incompatible with
        vniRIPTxDestAddress multicast."
    ::= { vniRIPEntry 2 }

vniRIPTxDestAddress OBJECT-TYPE
    SYNTAX      INTEGER {
                    broadcast(1),
                    multicast(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
      "RIP advertisement destination address.  vniR-
IPTxDestAddress
       multicast is incompatible with vniRIPSetting 
version1."
    ::= { vniRIPEntry 3 }

vniRIPTxRoutes OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "RIP route advertisement.  vniRIPTxRoutes
disabled is
        incompatible with vniRIPTxDefaultRoute en-
abled."
    ::= { vniRIPEntry 4 }

vniRIPTxDefaultRoute OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "RIP default route advertisement.  vniRIPTx-
DefaultRoute enabled
      is incompatible with vniRIPTxRoutes disabled."
    ::= { vniRIPEntry 5 }

vniRIPRxSetting OBJECT-TYPE
    SYNTAX      INTEGER {
                    version1(1),
                    version2(2),
                    version1or2(3),
                    disabled(4)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "RIP update acceptance.  vniRIPRxSetting dis-
abled is incompatible
        with vniRIPRxDefaultRoute enabled."
    ::= { vniRIPEntry 6 }

vniRIPRxDefaultRoute OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "RIP default route update acceptance.  vniR-
IPRxDefaultRoute
       enabled is incompatible with vniRIPRxSetting
disabled."
```

```
    ::= { vniRIPEntry 7 }

vniRIPSplitHorizon OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "RIP Split Horizon settings.  vniRIPSplitHo-
rizon disabled is
        incompatible with vniRIPPoisonedReverse en-
abled."
    ::= { vniRIPEntry 8 }

vniRIPPoisonedReverse OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "RIP Poisoned Reverse settings.  vniRIPPoi-
sonedReverse enabled
       is incompatible with vniRIPSplitHorizon dis-
abled."
    ::= { vniRIPEntry 9 }

vniRIPTriggeredResponse OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "RIP Triggered Response Settings"
    ::= { vniRIPEntry 10 }

vniSNMPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "SNMP state"
    ::= { vniSNMP 1 }
```

```
vniSNMPCommunitySet OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Set community string currently active for
this device"
    ::= { vniSNMP 2 }

vniSNMPCommunityGet OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Get community string currently active for
this device"
    ::= { vniSNMP 3 }

vniSNMPTrapCommunity1 OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap community string 1 for this device"
    ::= { vniSNMP 4 }

vniSNMPTrapCommunity2 OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap community string 2 for this device"
    ::= { vniSNMP 5 }

vniSNMPTrapCommunity3 OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap community string 3 for this device"
    ::= { vniSNMP 6 }

vniSNMPTrapCommunity4 OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap community string 4 for this device"
    ::= { vniSNMP 7 }
```

```
vniSNMPTrapCommunity5 OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..35))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap community string 5 for this device"
    ::= { vniSNMP 8 }

vniSNMPTrapIpAddress1 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap IP Address 1 for this device"
    ::= { vniSNMP 9 }

vniSNMPTrapIpAddress2 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap IP Address 2 for this device"
    ::= { vniSNMP 10 }

vniSNMPTrapIpAddress3 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap IP Address 3 for this device"
    ::= { vniSNMP 11 }

vniSNMPTrapIpAddress4 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap IP Address 4 for this device"
    ::= { vniSNMP 12 }

vniSNMPTrapIpAddress5 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trap IP Address 5 for this device"
    ::= { vniSNMP 13 }

vniSNMPTrapColdStart OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Cold Start Trap functionality"
    ::= { vniSNMP 14 }

vniSNMPTrapWarmStart OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Warm Start Trap functionality"
    ::= { vniSNMP 15 }

vniSNMPTrapLinkDown OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Link Down Trap functionality"
    ::= { vniSNMP 16 }

vniSNMPTrapLinkUp OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Link Up Trap functionality"
    ::= { vniSNMP 17 }

vniSNMPTrapAuthenticationFailure OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Authentication Failure Trap functionality"
    ::= { vniSNMP 18 }
```

```
vniSNMPTrapRisingAlarm OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Rising Alarm Trap functionality"
    ::= { vniSNMP 19 }

vniSNMPTrapFallingAlarm OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Falling Alarm Trap functionality"
    ::= { vniSNMP 20 }

vniSNMPTrapTopologyChange OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Topology Change Trap functionality"
    ::= { vniSNMP 21 }

vniSNMPTrapUndersizeFrame OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable or disable undersize frame trap"
    ::= { vniSNMP 22 }

vniSNMPTrapCRCFrame OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable or disable CRC frame trap"
    ::= { vniSNMP 23 }

vniSNMPTrapOversizeFrame OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable or disable Oversize frame trap"
    ::= { vniSNMP 24 }

vniSNMPTrapJabberFrame OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable or disable Jabber frame trap"
    ::= { vniSNMP 25 }

vniTFTPFilename OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..51))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Filename for TFTP download"
    ::= { vniTFTP 1 }

vniTFTPIpAddress OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IPAddress of TFTP server"
    ::= { vniTFTP 2 }

vniTFTPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    start(1),
                    inProgress(2),
                    done(3),
                    error(4)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "State of the current TFTP session"
    ::= { vniTFTP 3 }

VniDHCPGatewayEntry ::=
    SEQUENCE {
        vniDHCPGatewayVLANID
            INTEGER,
        vniDHCPGatewayState
            INTEGER,
        vniDHCPGatewayMaxHops
            INTEGER,
        vniDHCPGatewayDelay
            INTEGER,
        vniDHCPGatewayIPAddress1
            IpAddress,
        vniDHCPGatewayIPAddress2
            IpAddress,
        vniDHCPGatewayIPAddress3
            IpAddress,
        vniDHCPGatewayIPAddress4
            IpAddress,
        vniDHCPGatewayRelayIfArray
            OCTET STRING
    }

vniDHCPGatewayTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniDHCPGatewayEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway Table"
    ::= { vniDHCP 1 }

vniDHCPGatewayEntry OBJECT-TYPE
    SYNTAX      VniDHCPGatewayEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway Entry"
    INDEX       { vniDHCPGatewayVLANID }
    ::= { vniDHCPGatewayTable 1 }

vniDHCPGatewayVLANID OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "VLAN ID"
    ::= { vniDHCPGatewayEntry 1 }
```

```
vniDHCPGatewayState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway functionality state"
    ::= { vniDHCPGatewayEntry 2 }

vniDHCPGatewayMaxHops OBJECT-TYPE
    SYNTAX      INTEGER(1..16)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "DHCP Gateway maximum number of network hops"
    ::= { vniDHCPGatewayEntry 3 }

vniDHCPGatewayDelay OBJECT-TYPE
    SYNTAX      INTEGER(0..65535)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway delay"
    ::= { vniDHCPGatewayEntry 4 }

vniDHCPGatewayIPAddress1 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway Preferred IP Address 1"
    ::= { vniDHCPGatewayEntry 5 }

vniDHCPGatewayIPAddress2 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway Preferred IP Address 2"
    ::= { vniDHCPGatewayEntry 6 }

vniDHCPGatewayIPAddress3 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway Preferred IP Address 3"
    ::= { vniDHCPGatewayEntry 7 }
```

```
vniDHCPGatewayIPAddress4 OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "DHCP Gateway Preferred IP Address 4"
    ::= { vniDHCPGatewayEntry 8 }

vniDHCPGatewayRelayIfArray OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..64))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Array of VLAN IDs (2 bytes each) with a
maximum of
        32 VLAN IDs."
    ::= { vniDHCPGatewayEntry 9 }

vniHTTPState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "HTTP state"
    ::= { vniHTTP 1 }

vniHTTPPort OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "HTTP port number"
    ::= { vniHTTP 2 }

vniIPAccessCheckState OBJECT-TYPE
    SYNTAX      INTEGER {
                    enabled(1),
                    disabled(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Access Check state"
    ::= { vniIPAccess 1 }

VniIPAccessEntry ::=
    SEQUENCE {
```

```
                vniIPAccessIndex
                    INTEGER,
                vniIPAccessAddress
                    IpAddress,
                vniIPAccessValidBitCount
                    INTEGER,
                vniIPAccessFlags
                    INTEGER
            }

        vniIPAccessTable OBJECT-TYPE
            SYNTAX      SEQUENCE OF VniIPAccessEntry
            ACCESS      not-accessible
            STATUS      mandatory
            DESCRIPTION
                "IP Access Check table"
            ::= { vniIPAccess 2 }

        vniIPAccessEntry OBJECT-TYPE
            SYNTAX      VniIPAccessEntry
            ACCESS      not-accessible
            STATUS      mandatory
            DESCRIPTION
                "IP Access Entry"
            INDEX       { vniIPAccessIndex }
            ::= { vniIPAccessTable 1 }

        vniIPAccessIndex OBJECT-TYPE
            SYNTAX      INTEGER
            ACCESS      read-only
            STATUS      mandatory
            DESCRIPTION
                "IP Access Check Table index"
            ::= { vniIPAccessEntry 1 }

        vniIPAccessAddress OBJECT-TYPE
            SYNTAX      IpAddress
            ACCESS      read-write
            STATUS      mandatory
            DESCRIPTION
                "IP Access Check Table IP Address"
            ::= { vniIPAccessEntry 2 }

        vniIPAccessValidBitCount OBJECT-TYPE
            SYNTAX      INTEGER
            ACCESS      read-write
            STATUS      mandatory
            DESCRIPTION
                "IP Access Check Table Valid Bit Count"
            ::= { vniIPAccessEntry 3 }
```

```
vniIPAccessFlags OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP Access Check Table Flags
         bit #    feature
             0    HTTP
             1    telnet
             2    SNMP"
    ::= { vniIPAccessEntry 4 }

VniMirrorEntry ::=
    SEQUENCE {
        vniMirrorIndex
            INTEGER,
        vniMirrorPort
            INTEGER,
        vniMirrorMode
            INTEGER,
        vniMirrorAvailablePortMap
            OCTET STRING
    }

vniMirrorTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniMirrorEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Port Mirroring table"
    ::= { vniMirror 1 }

vniMirrorEntry OBJECT-TYPE
    SYNTAX      VniMirrorEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Mirror Entry"
    INDEX       { vniMirrorIndex }
    ::= { vniMirrorTable 1 }

vniMirrorIndex OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Mirror index"
    ::= { vniMirrorEntry 1 }

vniMirrorPort OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Mirror port"
    ::= { vniMirrorEntry 2 }

vniMirrorMode OBJECT-TYPE
    SYNTAX      INTEGER {
                    receive(1),
                    transmit(2)
                }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Mirror mode"
    ::= { vniMirrorEntry 3 }

vniMirrorAvailablePortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Trunk port map"
    ::= { vniMirrorEntry 4 }

VniTrunkEntry ::=
    SEQUENCE {
        vniTrunkIndex
            INTEGER,
        vniTrunkPortMap
            OCTET STRING,
        vniTrunkSelection1PortMap
            OCTET STRING,
        vniTrunkSelection2PortMap
            OCTET STRING
    }

vniTrunkTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniTrunkEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Port Trunking table"
    ::= { vniTrunk 1 }

vniTrunkEntry OBJECT-TYPE
    SYNTAX      VniTrunkEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Trunk Entry"
    INDEX       { vniTrunkIndex }
    ::= { vniTrunkTable 1 }

vniTrunkIndex OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Trunk index"
    ::= { vniTrunkEntry 1 }

vniTrunkPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Trunk port map"
    ::= { vniTrunkEntry 2 }

vniTrunkSelection1PortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Trunk selection ort map 1"
    ::= { vniTrunkEntry 3 }

vniTrunkSelection2PortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Trunk selection port map 2"
    ::= { vniTrunkEntry 4 }

VniSDBEntry ::=
    SEQUENCE {
        vniSDBMacAddress
            PhysAddress,
        vniSDBVlanId
            INTEGER,
        vniSDBPortNo
            INTEGER
    }

vniSDBTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniSDBEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Switching Database table"
    ::= { vniSDB 1 }

vniSDBEntry OBJECT-TYPE
    SYNTAX      VniSDBEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "SDB Entry"
    INDEX       { vniSDBMacAddress, vniSDBVlanId }
    ::= { vniSDBTable 1 }

vniSDBMacAddress OBJECT-TYPE
    SYNTAX      PhysAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Switching Database MAC address"
    ::= { vniSDBEntry 1 }

vniSDBVlanId OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Switching Database VLAN ID"
    ::= { vniSDBEntry 2 }

vniSDBPortNo OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Switching Database port number"
    ::= { vniSDBEntry 3 }

VniSDBMacCountEntry ::=
    SEQUENCE {
        vniSDBMacCountPortNo
            INTEGER,
        vniSDBMacCountVlanId
            INTEGER,
        vniSDBMacCountNo
            INTEGER
    }

vniSDBMacCountTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniSDBMacCountEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "Switching Database Mac Address Count"
    ::= { vniSDBMacCount 1 }

vniSDBMacCountEntry OBJECT-TYPE
    SYNTAX      VniSDBMacCountEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "SDB Mac Address CountEntry"
    INDEX       { vniSDBMacCountPortNo, vniSDBMac-
CountVlanId }
    ::= { vniSDBMacCountTable 1 }

vniSDBMacCountPortNo OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Switching Database Mac count port number
index"
    ::= { vniSDBMacCountEntry 1 }

vniSDBMacCountVlanId OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
       "Switching Database Mac Count VLAN ID index"
    ::= { vniSDBMacCountEntry 2 }

vniSDBMacCountNo OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Switching Database Mac Address Count"
    ::= { vniSDBMacCountEntry 3 }

VniIGMPEntry ::=
    SEQUENCE {
        vniIGMPVlanId
            INTEGER,
        vniIGMPIpGroup
            IpAddress,
        vniIGMPPortMap
            OCTET STRING
    }

vniIGMPTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniIGMPEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IGMP database table"
    ::= { vniIGMP 1 }

vniIGMPEntry OBJECT-TYPE
    SYNTAX      VniIGMPEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IGMP entry"
    INDEX       { vniIGMPVlanId, vniIGMPIpGroup }
    ::= { vniIGMPTable 1 }

vniIGMPVlanId OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Vlan ID as first index"
    ::= { vniIGMPEntry 1 }

vniIGMPIpGroup OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "IP multicast group address"
    ::= { vniIGMPEntry 2 }

vniIGMPPortMap OBJECT-TYPE
    SYNTAX      OCTET STRING
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "IP multicast ports"
    ::= { vniIGMPEntry 3 }

VniIGMPHostEntry ::=
    SEQUENCE {
        vniIGMPHostVlanId
            INTEGER,
        vniIGMPHostIpGroup
            IpAddress,
        vniIGMPHostAddress
            PhysAddress
    }

vniIGMPHostTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniIGMPHostEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IGMP database host table"
    ::= { vniIGMP 2 }

vniIGMPHostEntry OBJECT-TYPE
    SYNTAX      VniIGMPHostEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IGMP Host entry"
    INDEX       { vniIGMPHostVlanId, vniIGMPHostIp-
Group, vniIGMPHostAddress }
    ::= { vniIGMPHostTable 1 }

vniIGMPHostVlanId OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Vlan ID as first index"
    ::= { vniIGMPHostEntry 1 }

vniIGMPHostIpGroup OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "IP multicast group address"
    ::= { vniIGMPHostEntry 2 }

vniIGMPHostAddress OBJECT-TYPE
    SYNTAX      PhysAddress
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "IP multicast host"
    ::= { vniIGMPHostEntry 3 }

vniStackCurrentSize OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "Stack Size"
    ::= { vniStack 1 }

vniStackMasterSwitchId OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
```

```
            "Switch ID"
        ::= { vniStack 2 }

    vniStackState OBJECT-TYPE
        SYNTAX      INTEGER {
                        enabled(1),
                        disabled(2)
                    }
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Stack State"
        ::= { vniStack 3 }

    vniStackId OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Stack Identifier used as Stack IP Address"
        ::= { vniStack 4 }

    vniStackNetmask OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Stack Netmask"
        ::= { vniStack 5 }

    vniStackMacType OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Stack MAC Type"
        ::= { vniStack 6 }

    vniStackSize OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Stack Size"
        ::= { vniStack 7 }

    vniStackSwitchId OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
```

```
            "Switch ID"
        ::= { vniStack 8 }

    vniStackVlanId OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Configurable Stack Identifier"
        ::= { vniStack 9 }

    vniStackLocalPortMap OBJECT-TYPE
        SYNTAX      OCTET STRING
        ACCESS      read-write
        STATUS      mandatory
        DESCRIPTION
            "Local Stack Port Map."
        ::= { vniStack 10 }

    VniStackPortEntry ::=
        SEQUENCE {
            vniStackPortNo
                INTEGER,
            vniStackPortSwitchId
                INTEGER,
            vniStackPortSwitchPortNo
                INTEGER
        }

    vniStackPortTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF VniStackPortEntry
        ACCESS      not-accessible
        STATUS      mandatory
        DESCRIPTION
            "Stack Port Mapping Table"
        ::= { vniStack 11 }

    vniStackPortEntry OBJECT-TYPE
        SYNTAX      VniStackPortEntry
        ACCESS      not-accessible
        STATUS      mandatory
        DESCRIPTION
            "vniStackPortEntry"
        INDEX       { vniStackPortNo }
        ::= { vniStackPortTable 1 }

    vniStackPortNo OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
            "Stacking Port Number"
        ::= { vniStackPortEntry 1 }

    vniStackPortSwitchId OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
            "Slave Switch ID"
        ::= { vniStackPortEntry 2 }

    vniStackPortSwitchPortNo OBJECT-TYPE
        SYNTAX      INTEGER
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
            "Slave’s local port number"
        ::= { vniStackPortEntry 3 }
    --
    -- 24+2 SwitchBeat MIB OBJECTS
    --
    swMasterTraps  OBJECT IDENTIFIER ::= { swMaster  0 }

    swSlaveNoResp   NOTIFICATION-TYPE
              OBJECTS { swIPPollAddr,  swIPPollType,
    swTmPollEnab,
                      swTmLastPoll,  swTmLastPollRsp,
    swTotPolls,
                             swTotPollResps, swTotPoll-
    RespTOs,  swTotPollOosResps,
                              swTotPollDstUnreach }
                    STATUS  current
                    DESCRIPTION
                    "Slave failed to respond to polls."
        ::= { swMasterTraps 1 }

    swIPPollList   OBJECT IDENTIFIER ::= { swMaster  1 }

    SwIPPollEntry ::=
        SEQUENCE {
            swIPPollAddr
                IpAddress,
            swIpSrcPort
                INTEGER,
            swIpDstPort
                INTEGER,
            swIPPollEnab
                INTEGER,
            swIPPollInterv
                INTEGER,
            swIPPollType
```

```
            INTEGER,
        swIPPollMin
            INTEGER,
        swIPMastrTrpDelay
            INTEGER,
        swIPPollRoute
            INTEGER,
        swIPMastrTrpAddr1
            IpAddress,
        swIPMastrTrpAddr2
            IpAddress,
        swTmPollEnab
            INTEGER,
        swTmLastPoll
            INTEGER,
        swTmLastPollRsp
            INTEGER,
        swTotPolls
            INTEGER,
        swTotPollResps
            INTEGER,
        swTotPollRespTOs
            INTEGER,
        swTotPollOosResps
            INTEGER,
        swTotPollDstUnreach
            INTEGER
    }

swIPPollTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF SwIPPollEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list tabular data"
    ::= { swIPPollList 1 }

swIPPollEntry   OBJECT-TYPE
    SYNTAX      SwIPPollEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list table entry"
    INDEX       { swIPPollAddr }
    ::= { swIPPollTable 1 }

swIPPollAddr    OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
```

```
        "IP address of slave to poll"
    ::= { swIPPollEntry 1 }

swIpSrcPort     OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "The source’s IP Port number for sends and
receives "
    ::= { swIPPollEntry 2 }

swIpDstPort     OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "The destination’s IP Port number for sends
and receives "
    ::= { swIPPollEntry 3 }

swIPPollEnab    OBJECT-TYPE
    SYNTAX      INTEGER
               {
                 nogo (0),
                 go   (1)
               }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable/disable polling for this IP address"
    ::= { swIPPollEntry 4 }

swIPPollInterv  OBJECT-TYPE
    SYNTAX      INTEGER(50..5000)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Polling duty cycle in msec"
    ::= { swIPPollEntry 5 }

swIPPollType    OBJECT-TYPE
    SYNTAX      INTEGER
               {
                 ping (0),
                 ruup (1)
               }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Select polling protocol (Ping or RUUP)"
    ::= { swIPPollEntry 6 }

swIPPollMin     OBJECT-TYPE
    SYNTAX      INTEGER(0..100)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
      "Minimum number of missed polls before sending
a trap"
    ::= { swIPPollEntry 7 }

swIPMastrTrpDelay OBJECT-TYPE
    SYNTAX        INTEGER(50..5000)
    ACCESS        read-write
    STATUS        mandatory
    DESCRIPTION
        "Trap duty cycle in msec"
    ::= { swIPPollEntry 8 }

swIPPollRoute   OBJECT-TYPE
    SYNTAX      INTEGER
               {
                 nogo (0),
                 go   (1)
               }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable/disable IP routing of polls"
    ::= { swIPPollEntry 9 }

swIPMastrTrpAddr1 OBJECT-TYPE
    SYNTAX        IpAddress
    ACCESS        read-write
    STATUS        mandatory
    DESCRIPTION
        "First IP address of where to send Traps"
    ::= { swIPPollEntry 10 }

swIPMastrTrpAddr2 OBJECT-TYPE
    SYNTAX        IpAddress
    ACCESS        read-write
    STATUS        mandatory
    DESCRIPTION
        "Second IP address of where to send Traps"
    ::= { swIPPollEntry 11 }

swTmPollEnab    OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "SYSUPTIME when polling was enabled"
    ::= { swIPPollEntry 12 }

swTmLastPoll    OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "SYSUPTIME of latest poll"
    ::= { swIPPollEntry 13 }

swTmLastPollRsp OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "SYSUPTIME of latest response to a poll"
    ::= { swIPPollEntry 14 }

swTotPolls      OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
      "Total number of polls sent to this IP address"
    ::= { swIPPollEntry 15 }

swTotPollResps  OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
       "Total number of poll responses from this IP
address"
    ::= { swIPPollEntry 16 }

swTotPollRespTOs  OBJECT-TYPE
    SYNTAX        INTEGER
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION
       "Total number of poll responses from this IP
address that are out of sequence"
    ::= { swIPPollEntry 17 }

swTotPollOosResps OBJECT-TYPE
    SYNTAX        INTEGER
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION
```

```
       "Total number of poll responses from this IP
address that are out of sequence"
    ::= { swIPPollEntry 18 }

swTotPollDstUnreach OBJECT-TYPE
    SYNTAX          INTEGER
    ACCESS          read-only
    STATUS          mandatory
    DESCRIPTION
       "Total number of polls for which Destination
Unreachable was received"
    ::= { swIPPollEntry 19 }

swSlaveTraps   OBJECT IDENTIFIER ::= { swSlave  0 }

swMasterNoPoll  NOTIFICATION-TYPE
               OBJECTS { swIPMonAddr, swTmMonEnab,
swTmLastMon, swTotMons,
                         swTotMonPollTOs, swTotMon-
PollOos }
                STATUS  current
                DESCRIPTION
             "Master failed to poll within specified
interval."
    ::= { swSlaveTraps 1 }

swIPMonList    OBJECT IDENTIFIER ::= { swSlave  1 }

SwIPMonEntry ::=
    SEQUENCE {
        swIPMonAddr
            IpAddress,
        swIpMonSrcPort
            INTEGER,
        swIpMonDstPort
            INTEGER,
        swIPMonEnab
            INTEGER,
        swIPMaxWait
            INTEGER,
        swIpMonMinMissed
            INTEGER,
        swIPSlaveTrpDelay
            INTEGER,
        swIPSlaveTrpAddr1
            IpAddress,
        swIPSlaveTrpAddr2
            IpAddress,
        swTmMonEnab
            INTEGER,
        swTmLastMon
```

```
            INTEGER,
        swTotMons
            INTEGER,
        swTotMonPollTOs
            INTEGER,
        swTotMonPollOos
            INTEGER
    }

swIPMonTable    OBJECT-TYPE
    SYNTAX      SEQUENCE OF SwIPMonEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP monitor list tabular data"
    ::= { swIPMonList 1 }

swIPMonEntry    OBJECT-TYPE
    SYNTAX      SwIPMonEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP monitor list table entry"
    INDEX       { swIPMonAddr }
    ::= { swIPMonTable 1 }

swIPMonAddr     OBJECT-TYPE
    SYNTAX      IpAddress
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP address of master to monitor"
    ::= { swIPMonEntry 1 }

swIpMonSrcPort  OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "The source’s IP Port number for sends and
receives "
    ::= { swIPMonEntry 2 }

swIpMonDstPort  OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "The destination’s IP Port number for sends
and receives "
    ::= { swIPMonEntry 3 }
```

```
swIPMonEnab     OBJECT-TYPE
    SYNTAX      INTEGER
               {
                 nogo (0),
                 go   (1)
               }
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Enable/disable monitoring for this IP ad-
dress"
    ::= { swIPMonEntry 4 }

swIPMaxWait     OBJECT-TYPE
    SYNTAX      INTEGER(50..10000)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Maximum interval to wait for a poll from
this IP address in msec"
    ::= { swIPMonEntry 5 }

swIpMonMinMissed OBJECT-TYPE
    SYNTAX       INTEGER
    ACCESS       read-write
    STATUS       mandatory
    DESCRIPTION
        "Minimum number of missed polls before de-
claring a fault"
    ::= { swIPMonEntry 6 }

swIPSlaveTrpDelay OBJECT-TYPE
    SYNTAX        INTEGER(50..5000)
    ACCESS        read-write
    STATUS        mandatory
    DESCRIPTION
        "Trap duty cycle in msec"
    ::= { swIPMonEntry 7 }

swIPSlaveTrpAddr1 OBJECT-TYPE
    SYNTAX        IpAddress
    ACCESS        read-write
    STATUS        mandatory
    DESCRIPTION
        "First IP address of where to send Traps"
    ::= { swIPMonEntry 8 }

swIPSlaveTrpAddr2 OBJECT-TYPE
    SYNTAX        IpAddress
    ACCESS        read-write
    STATUS        mandatory
    DESCRIPTION
        "Second IP address of where to send Traps"
    ::= { swIPMonEntry 9 }

swTmMonEnab     OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "SYSUPTIME when monitoring was enabled"
    ::= { swIPMonEntry 10 }

swTmLastMon     OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "SYSUPTIME of latest poll received"
    ::= { swIPMonEntry 11 }

swTotMons       OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
       "Total number of polls received from this IP
address"
    ::= { swIPMonEntry 12 }

swTotMonPollTOs OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
       "Total number of timeouts waiting for polls"
    ::= { swIPMonEntry 13 }

swTotMonPollOos OBJECT-TYPE
    SYNTAX      INTEGER
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
       "Total number of polls received with out-of-
sequence N(S) or N(R)"
    ::= { swIPMonEntry 14 }
```

```
undersizeTrap   OBJECT IDENTIFIER ::= { undersize-
Frame  0 }

undersizeFrameTrap   NOTIFICATION-TYPE
                OBJECTS { snmpUndersizeFrameCount,
snmpUndersizeFramePortNum }
                STATUS  current
                DESCRIPTION
               "undersize frame error has increased
since last polled."
    ::= { undersizeTrap 1 }

undersizeFrameList    OBJECT IDENTIFIER ::= { under-
sizeFrame  1 }


UndersizeFrameEntry ::=
    SEQUENCE {
        snmpUndersizeFrameCount
            INTEGER,
        snmpUndersizeFramePortNum
            INTEGER
    }

undersizeFrameTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF UndersizeFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list tabular data"
    ::= { undersizeFrameList 1 }

undersizeFrameEntry   OBJECT-TYPE
    SYNTAX      UndersizeFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list table entry"
    INDEX       { snmpUndersizeFramePortNum }
    ::= { undersizeFrameTable 1 }

snmpUndersizeFrameCount    OBJECT-TYPE
    SYNTAX      INTEGER(0..50000)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "IP address of slave to poll"
    ::= { undersizeFrameEntry 1 }

snmpUndersizeFramePortNum    OBJECT-TYPE
    SYNTAX      INTEGER(1..26)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Enable/disable polling for this IP address"
    ::= { undersizeFrameEntry 2 }
```

```
oversizeTrap  OBJECT IDENTIFIER ::= { oversizeFrame
0 }

oversizeFrameTrap   NOTIFICATION-TYPE
                OBJECTS { snmpOversizeFrameCount,
snmpOversizeFramePortNum }
                STATUS  current
                DESCRIPTION
               "oversize frame error has increased
since last polled."
    ::= { oversizeTrap 1 }

oversizeFrameList    OBJECT IDENTIFIER ::= { over-
sizeFrame  1 }


OversizeFrameEntry ::=
    SEQUENCE {
        snmpOversizeFrameCount
            INTEGER,
        snmpOversizeFramePortNum
            INTEGER
    }

oversizeFrameTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF OversizeFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list tabular data"
    ::= { oversizeFrameList 1 }

oversizeFrameEntry   OBJECT-TYPE
    SYNTAX      OversizeFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list table entry"
    INDEX       { snmpOversizeFramePortNum }
    ::= { oversizeFrameTable 1 }

snmpOversizeFrameCount    OBJECT-TYPE
    SYNTAX      INTEGER(0..50000)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Enable/disable polling for this IP address"
    ::= { oversizeFrameEntry 1 }

snmpOversizeFramePortNum  OBJECT-TYPE
    SYNTAX      INTEGER(1..26)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Polling duty cycle in msec"
    ::= { oversizeFrameEntry 2 }
```

```
crcTrap   OBJECT IDENTIFIER ::= { crcFrame  0 }

crcFrameTrap   NOTIFICATION-TYPE
              OBJECTS { snmpCrcFrameCount, snmpCrc-
FramePortNum }
                STATUS  current
                DESCRIPTION
               "crc frame error has increased since
last polled."
    ::= { crcTrap 1 }

crcFrameList   OBJECT IDENTIFIER ::= { crcFrame  1 }


CrcFrameEntry ::=
    SEQUENCE {
        snmpCrcFrameCount
            INTEGER,
        snmpCrcFramePortNum
            INTEGER
    }
```

```
crcFrameTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF CrcFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list tabular data"
    ::= { crcFrameList 1 }

crcFrameEntry   OBJECT-TYPE
    SYNTAX      CrcFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list table entry"
    INDEX       { snmpCrcFramePortNum }
    ::= { crcFrameTable 1 }

snmpCrcFrameCount    OBJECT-TYPE
    SYNTAX      INTEGER(0..50000)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Enable/disable polling for this IP address"
    ::= { crcFrameEntry 1 }

snmpCrcFramePortNum  OBJECT-TYPE
    SYNTAX      INTEGER(1..26)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Polling duty cycle in msec"
    ::= { crcFrameEntry 2 }
```

```
jabberTrap  OBJECT IDENTIFIER ::= { jabberFrame  0 }

jabberFrameTrap   NOTIFICATION-TYPE
                OBJECTS { snmpJabberFrameCount,
snmpJabberFramePortNum }
                STATUS  current
                DESCRIPTION
                 "jabber frame error has increased
since last polled."
    ::= { jabberTrap 1 }
```

```
jabberFrameList    OBJECT IDENTIFIER ::= { jabber-
Frame  1 }


JabberFrameEntry ::=
    SEQUENCE {
        snmpJabberFrameCount
            INTEGER,
        snmpJabberFramePortNum
            INTEGER
    }

jabberFrameTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF JabberFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list tabular data"
    ::= { jabberFrameList 1 }

jabberFrameEntry   OBJECT-TYPE
    SYNTAX      JabberFrameEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "IP poll list table entry"
    INDEX       { snmpJabberFramePortNum }
    ::= { jabberFrameTable 1 }

snmpJabberFrameCount    OBJECT-TYPE
    SYNTAX      INTEGER(0..50000)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
       "Enable/disable polling for this IP address"
    ::= { jabberFrameEntry 1 }

snmpJabberFramePortNum  OBJECT-TYPE
    SYNTAX      INTEGER(1..26)
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "Polling duty cycle in msec"
    ::= { jabberFrameEntry 2 }



END
```

5 Troubleshooting and Technical Support

This chapter includes:

- Troubleshooting the 24+2 Ethernet Switch
- Contacting Technical Support

Troubleshooting

In the event that the 24+2 Ethernet Switch should fail in any way, ensure first that the external cables are properly connected.

Removing and Reinstalling the 24+2 Ethernet Switch

If the external cables are properly connected and you continue to experience problems with the Switch, try removing it and the transition card from the system according to the procedures below. Next, reinstall the Switch according to the instructions in Chapter 2.

Removing the 24+2 Ethernet Switch

To remove the Ethernet Switch:

1. Remove power.
2. Disconnect I/O connections on front, if any.
3. Remove the Ethernet Switch.

Removing the Transition Card

To remove the transition card:

1. Remove power.
2. Disconnect I/O connections on rear, if any.
3. Remove the transition card.

Unable to Establish a Link

If you are unable to establish a link, make sure you are using the correct cables for your type of connection.

Host-to-Switch	Switch-to-Switch
Straight-through cable	Crossover cable

Connection cables

Contacting Technical Support

If you are unable to get the 24+2 Ethernet Switch to function properly, contact the Technical Support team at Continuous Computing by any of the methods listed below.

Note Please be sure to include the serial numbers for each affected part. In addition, we will need to know any relevant details about the environment in which the Switch is operating, including network topology, cabling, power, and type of hosts being used.

Contacting Technical Support

To contact the Technical Support team at Continuous Computing, do one of the following:

- Email us at support@ccpu.com
- Visit our support website at http://support.ccpu.com

  This site features our automatic technical support system. Create a new user profile, then submit a new ticket at the "Welcome to SupportWizard" page. This process ensures that our team delivers a timely solution to any technical problem you have.
- Call us at (858) 882-8911, 9:00 a.m. - 5:00 p.m. (PST)

Note If you have a Gold or Platinum service contract, follow the contact instructions provided with your contract.

Numerics

1000Base-FX ports 1-5
10Base-T/100Base-TX ports 1-4

A

Assigning an IP address 5-2
Auto-sensing mode 1-5

B

Back pressure flow control 1-5
Back view 2-1
Basic functions
  Layer-2 switching 1-7
Boot CLI commands 3-11
Boot CLI output 3-41
Bridge groups supported 4-3
Bridge MIB 4-2
Broadcast containment 1-10

C

Class-of-Service (CoS) support 1-12
CLI commands 3-1
Command Line Interface 3-1

D

DHCP 1-14

E

Enhanced security 1-10
Extra commands 3-10, 3-41, 3-45

F

Fast Ethernet
  ports 1-4
Filtering 1-8
Firmware CLI commands 3-2, 3-13
Flow control
  back pressure 1-5
Forwarding 1-9
Front view 2-1
Full-duplex mode 1-5

G

GVRP 1-14

H

Half-duplex mode 1-5
Help command 3-42
High-level system specifications 1-2

I

ICMP Router Discovery 1-15
IGMP snooping 1-15
Inter-VLAN communication 1-12
Intra-VLAN communication 1-12
IP address assignment 5-2
IP multicast filtering 1-15

L

Learning 1-8
Local console management 3-1
Logging on 3-2, 5-2

M

Management
  local console 3-1
  remote console 3-1, 5-1
  SNMP 3-2
Membership into VLANs 1-11
MIB, Vertex private 4-3
MIB-2 4-2
Multicast switching 1-9
Multicast-based multimedia applications 1-10

P

Password 3-2
Password default 3-2
Physical ports 1-4
Ping command 3-38, 3-44
Ports
  1000Base-FX 1-5
  10Base-T/100Base-TX 1-4
  physical 1-4

Q

Quality-of-Service (QoS) support 1-12

R

Remote console management 3-1, 5-1
Remote learning 1-11
Remove command 3-10, 3-37, 3-44

Reset command 3-40, 3-44
RFC 1213 4-2
RFC 1493 4-2
RFC 1757 4-2
RMON
  defined 4-1
  groups supported 4-2
RMON MIB 4-2

S
Security 1-10
Set command 3-4, 3-22, 3-43
Show command 3-2, 3-14
SNMP
  Agent 4-2
  management 3-2
Switching
  multicast 1-9
  unicast 1-8

U
Unicast switching 1-8
  filtering 1-8
  forwarding 1-9
  learning 1-8
  VLAN classification 1-8
Upload command 3-4, 3-20, 3-42, 3-43

V
Vertex private MIB 4-3
Views 2-1
Virtual LAN (see VLAN) 1-10
VLAN
  broadcast containment 1-10
  configuration 1-12
  definition 1-10
  enhanced security 1-10
  membership 1-11
  multicast-based multimedia applications 1-10
  remote learning 1-11
VLAN classification 1-8
VLAN membership
  definitions 1-11
  learning 1-11

X
XpressFlow
  high-level specifications 1-2
XpressFlow Switch
  IP Address 5-2
  logging on to 3-2, 5-2
  physical ports 1-4