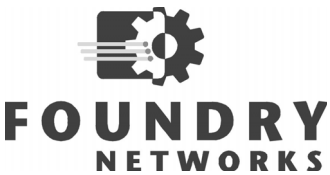
Foundry Diagnostic Guide



2100 Gold Street
P.O. Box 649100
San Jose, CA 95164-9100
Tel 408.586.1700
Fax 408.586.1900

November 2002

Contents

Chapter 1
Getting Started

Introduction

The *Foundry Diagnostic Guide* describes the diagnostic commands available on Foundry devices. The software procedures show how to perform tasks using the Command Line Interface (CLI).

Audience

This manual is designed for system administrators and support personnel with a working knowledge of Layer 2 and Layer 3 switching and routing.

If you are using a Foundry Layer 3 Switch, you should be familiar with the following protocols if applicable to your network – IP, RIP, OSPF, IS-IS, BGP4, MBGP, IGMP, PIM, DVMRP, IPX, AppleTalk, FSRP, VRRP, and VRRPE.

Nomenclature

This guide uses the following typographical conventions to show information:

Italic	highlights the title of another publication and occasionally emphasizes a word or phrase.
Bold	highlights a CLI command.
Bold Italic	highlights a term that is being defined.

NOTE: A note emphasizes an important fact or calls your attention to a dependency.

WARNING: A warning calls your attention to a possible hazard that can cause injury or death.

CAUTION: A caution calls your attention to a possible hazard that can damage equipment.

Related Publications

The following Foundry Networks documents supplement the information in this guide.

- *Foundry Switch and Router Installation and Basic Configuration Guide* – provides hardware and software installation information, and configuration information for system-level features.
- *Foundry Security Guide* – provides procedures for securing management access to Foundry devices and for

protecting against Denial of Service (DoS) attacks.

- *Foundry Enterprise Configuration and Management Guide* – provides configuration information for enterprise routing protocols including IP, RIP, IP multicast, OSPF, BGP4, VRRP and VRRPE.
- *Foundry NetIron Service Provider Configuration and Management Guide* – provides configuration information for IS-IS and MPLS.
- *Foundry Switch and Router Command Line Interface Reference* – provides a list and syntax information for all the Layer 2 Switch and Layer 3 Switch CLI commands.
- *Foundry ServerIron Installation and Configuration Guide* – provides installation instructions as well as detailed feature descriptions, procedures, and application examples for Server Load Balancing (SLB), Global SLB (GSLB), Transparent Cache Switching (TCS), and URL Switching.
- *Foundry ServerIron Firewall Load Balancing Guide* – provides detailed feature descriptions, procedures, and application examples for Firewall Load Balancing (FWLB).
- *Foundry ServerIron Command Line Interface Reference* – provides detailed syntax information for all ServerIron CLI commands.

To order additional copies of these manuals, do one of the following:

- Call 1.877.TURBOCALL (887.2622) in the United States or 1.408.586.1881 outside the United States.
- Send email to info@foundrynet.com.

What's New In This Edition?

This edition describes the following software release:

- 07.6.01 (Layer 2 Switch and Layer 3 Switch)
- 07.2.25 (ServerIron)

This release applies to the following products:

- NetIron Internet Backbone router
- BigIron with M2 (Management II) or higher modules
- BigIron with Velocity Management Module version I (VM1)
- FastIron II, FastIron II Plus, and FastIron III with M2 or higher modules
- FastIron 4802
- ServerIron XL, ServerIron 400, and ServerIron 400

For a list of the enhancements, see the "Getting Started" chapter in the *Foundry Switch and Router Installation and Basic Configuration Guide*.

NOTE: If you want documentation specifically for a 07.1.x release, see the January, 2001 edition of the manuals and the release notes for the release you are using. For the 07.2.06 release or a 07.3.x release, see the June, 2001 edition and the release notes.

How to Get Help

Foundry Networks technical support will ensure that the fast and easy access that you have come to expect from your Foundry Networks products will be maintained.

Web Access

- http://www.foundrynetworks.com

Email Access

Technical requests can also be sent to the following email address:

- support@foundrynet.com

Telephone Access

- 1.877.TURBOCALL (887.2622) United States
- 1.408.586.1881 Outside the United States

Warranty Coverage

Contact Foundry Networks using any of the methods listed above for information about the standard and extended warranties.

Chapter 2
Using the Foundry Diagnostic Commands

The Foundry diagnostic commands are tools that you can use to gather information about Foundry devices. The diagnostic commands start with **de**, **debug**, **mm**, **phy**, and **ptrace**.

de Displays information about CPU buffer allocations.

debug Reports debugging information that you can use to resolve configuration problems.

mm Displays the contents of a specified address on every module. (Available on Chassis devices only)

phy Displays information about PHY (hardware) registers for a specified port.

ptrace Displays information on the console when a specified kind of packet is transmitted or received.

In addition, the **show ip bgp debug** command reports information about resource allocation and errors in a BGP configuration.

These commands are available in Privileged EXEC mode on the Command Line Interface (CLI) only. You cannot use them in IronView or the device's Web management interface. For complete syntax information for the diagnostic commands, see the next chapter, "Foundry Diagnostic Command Reference" on page 3-1.

Many of the diagnostic commands are meant to be used in conjunction with calls to Foundry technical support. If you report a problem, the support engineer may ask you to execute one or more of the diagnostic commands described in this guide. Some of the diagnostic commands report information about internal hardware settings and registers that is relevant primarily to Foundry engineering staff. Consequently, this information is not described in detail here.

The following table lists some of the tasks you can perform using the diagnostic commands:

Task	Relevant Commands
Tracing packets	ptrace *
Displaying AppleTalk information	debug appletalk ptrace appletalk *
Displaying BGP information	debug ip bgp * show ip bgp debug
Displaying OSPF packet information	debug ip ospf packet
Displaying VRRP packet information	debug ip vrrp packet

Task	Relevant Commands
Displaying BPDU packet information	debug spanning
Recovering a frozen console	debug serial *
Displaying CPU buffer information	de
Reading hardware registers	debug serial state phy
Displaying RSVP packet information	ptrace mpls rsvp
Displaying IS-IS packet information	debug isis

Using an ACL to Filter Debug Output

You can use an ACL to filter output from **debug** commands. For example, you can set up an ACL that permits packets from an IP address, then apply that ACL to a **debug** command. When you start the **debug** command, only messages related to that IP address are displayed in the output for that command.

The following example limits output from the **debug ip tcp packet** command to only messages related to incoming packets from 10.10.10.10.

First, set up an ACL to permit packets from host 10.10.10.10. For example:

```
BigIron(config)# access-list 100 permit ip host 10.10.10.10 any
```

Then apply this ACL to the **debug ip tcp** command. You can specify no more than one ACL per protocol.

```
BigIron# debug ip tcp acl 100
```

Syntax: debug ip <protocol> acl <acl-id>

Then enter the **debug ip tcp packet** command to start generating debug output.

```
BigIron# debug ip tcp packet
```

Syntax: [no] debug ip tcp packet

Only messages related to packets inbound from 10.10.10.10 are displayed in the output for the **debug ip tcp packet** command. To display messages related to outbound packets sent to 10.10.10.10, add another entry to the ACL, specifying 10.10.10.10 as the destination host. For example:

```
BigIron(config)# access-list 100 permit ip any host 10.10.10.10
```

The **show debug** command displays ACLs applied to debug commands. For example:

```
BigIron# show debug
Debug message destination:  Console
TCP:
          TCP:  packet debugging is on
          TCP:  Display is bound to ACL 100
```

Syntax: show debug

Chapter 3
Foundry Diagnostic Command Reference

This chapter lists and provides syntax and examples for the CLI **de**, **debug**, **mm**, **phy**, and **ptrace** commands.

About the Diagnostic Commands

You can enter the diagnostic commands at the Privileged EXEC CLI level. The following tables list the diagnostic commands and contains page references to descriptions of each command.

Diagnostic Commands for Layer 2 Switches and Layer 3 Switches

The following diagnostic commands are supported on Layer 2 Switches and Layer 3 Switches.

Diagnostic Commands for the ServerIron

The ServerIron supports the following diagnostic commands.

Diagnostic Commands – Syntax Descriptions

The following commands are available at the Privileged EXEC level of the CLI for Foundry switches and routers, except where noted.

de

Displays information about CPU buffer allocations.

EXAMPLE:

```
BigIron# de
GADDR        = 043a1588 TOT_IN       =       260 TOT_OUT      =       259
CPU_R        =         85        GET_B        =       175
SNOOP_M      =        175        SNOOP        =        28
FREE_B       =         56        FREE_B_M     =         0
Dram buf     =         63        No-bufs      =         0
```

The following table describes the output from the **de** command:

Table 3.1: Output from the de command

This Field...	Displays...
GADDR	Address of g_sw_sys
TOT_IN	Total number of CPU buffer allocations.
TOT_OUT	Total number of CPU buffer deallocations.
CPU_R	CPU read queue buffers.
GET_B	CPU buffers allocated by BM_GET_BUFFER.
SNOOP	Number of snoop operations.
SNOOP_M	Number of management snoop operations.
FREE_B	Number of buffers freed using BM_FREE_BUFFER or BM_FREE_BUFFER_MGMT.
FREE_B_M	Additional counter indicating number of buffers freed using just BM_FREE_BUFFER_MGMT.
Dram buf	Amount of available packet processing memory. This number should always be close to 64.
No-bufs	Number of times the CPU was unsuccessful in obtaining packet processing memory. This number should be 0 under normal operation.

Syntax: de

Possible values: N/A

Default value: N/A

debug all

Activates all debugging functions on the device. The **no** form of the command deactivates all debugging functions.

NOTE: Activating all debugging functions can generate a lot of output and greatly slow the operation of the device.

EXAMPLE:

```
BigIron# debug all
```

Syntax: [no] debug all

Possible values: N/A

Default value: N/A

debug appletalk

Displays the number of timer events dropped and insufficient zone allocations in an Appletalk configuration.

EXAMPLE:

```
BigIron# debug appletalk
Timer event Dropped: 0
Insufficient zone allocation: 0
```

Syntax: [no] debug appletalk

Possible values: N/A

Default value: N/A

debug atm multipoint

Displays ATM point-to-multipoint information.

EXAMPLE:

```
BigIron# debug atm multipoint
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug atm multipoint** command.

```
 TM_MULTIPOINT: INARP ATM length = 20
 ATM_MULTIPOINT: Tx INATMARP Request packet:
  source ip 1.1.1.2
  target ip 0.0.0.0
  inatmarp_pkt->src_atm_number_tl.length 0
  inatmarp_pkt->src_atm_subaddress_tl.length 0
  inatmarp_pkt->target_atm_number_tl.length 0
  inatmarp_pkt->target_atm_subaddress_tl.length 0
 BigIron Router#ATM_MULTIPOINT: INARP ATM length = 20
 ATM_MULTIPOINT: Rx INATMARP packet:
  source ip 1.1.1.1
  target ip 1.1.1.2
  inatmarp_pkt->src_atm_number_tl.length 0
  inatmarp_pkt->src_atm_subaddress_tl.length 0
  inatmarp_pkt->target_atm_number_tl.length 0
  inatmarp_pkt->target_atm_subaddress_tl.length 0
```

Syntax: [no] debug atm multipoint

Possible values: N/A

Default value: N/A

debug destination

Specifies a destination for debugging output. You can send debugging output to the console, Syslog buffer, a Telnet session, or an SSH session.

EXAMPLE:

```
BigIron# debug destination ssh 1
```


Syntax: debug destination console | logging | telnet <num> | ssh <num>

Possible values: Specify one of the following destinations:

console Directs debugging output to the system console.

logging Directs debugging output to the Syslog buffer and also to the Syslog server, if configured.

telnet <num> Directs debugging output to the specified Telnet session.

ssh <num> Directs debugging output to the specified SSH session.

Default value: By default, debugging output is sent to the Console.

debug gvrp packets

Displays GVRP information.

EXAMPLE:

```
BigIron# debug gvrp packets
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug gvrp packets** command.

```
GVRP: Port 2/1 RCV
GVRP: 0x2095ced4:  01 80 c2 00 00 21 00 e0 52 ab 87 40 00 28 42 42
GVRP: 0x2095cee4:  03 00 01 01 04 02 03 e9 04 01 03 eb 04 01 03 ec
GVRP: 0x2095cef4:  04 01 03 ef 04 01 03 f1 04 01 05 dd 04 01 09 cb
GVRP: 0x2095cf04:  04 01 0f a1 00 00
GVRP: Port 2/1 TX
GVRP: 0x207651b8:  01 80 c2 00 00 21 00 04 80 2c 0e 20 00 3a 42 42
GVRP: 0x207651c8:  03 00 01 01 02 00 04 05 03 e9 04 05 03 eb 04 05
GVRP: 0x207651d8:  03 ec 04 05 03 ef 04 05 03 f1 04 05 05 dd 04 05
GVRP: 0x207651e8:  09 cb 04 05 0f a1 04 02 00 02 04 01 00 07 04 01
GVRP: 0x207651f8:  00 09 04 01 00 0b 00 00
GVRP: Port 2/1 TX
GVRP: 0x207651b8:  01 80 c2 00 00 21 00 04 80 2c 0e 20 00 18 42 42
GVRP: 0x207651c8:  03 00 01 01 04 02 00 02 04 01 00 07 04 01 00 09
GVRP: 0x207651d8:  04 01 00 0b 00 00
```

Syntax: [no] debug gvrp packets

Possible values: N/A

Default value: N/A

debug ip arp

Displays information about ARP messages sent and received by the device.

EXAMPLE:

```
BigIron# debug ip arp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug arp** command.

```
                    [A]           [B]            [C]          [D]       [E]
 IP ARP: rcvd 192.168.4.56 000034ab67bd  , 192.168.4.32 00cdfeba23ab   9
 IP ARP: sent 192.168.4.32 000034ab67bd  , 192.168.4.4  00cdfeba23ab   9
```


Table 3.2 describes the contents of **debug ip arp** messages. The letters in brackets do not appear in the actual output.

Table 3.2: Output from the debug ip arp command

This Field...	Displays...
`rcvd or sent`	Indicates whether the packet was sent or received.
`[A] 192.168.4.56`	Source IP address.
`[B] 000034ab67bd`	Source MAC address.
`[C] 192.168.4.32`	Destination IP address.
`[D] 00cdfeba23ab`	Destination MAC address.
`[E] 9`	Port number.

Syntax: [no] debug ip arp

Possible values: N/A

Default value: N/A

debug ip bgp <address> updates

Displays BGP update information for a specific neighbor.

EXAMPLE:

```
BigIron# debug ip bgp 1.1.1.192 updates
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp <address> updates** command.

```
BGP: 1.1.1.192 rcvd UPDATE about 1.1.1.0/24 -- withdrawn
BGP: 1.1.1.192 rcvd UPDATE 5.5.5.0/24
BGP: 1.1.1.192 rcvd UPDATE about 5.5.5.0/24 -- withdrawn
```

Syntax: [no] debug ip bgp <ip-addr> updates

Possible values: Valid IP address

Default value: N/A

debug ip bgp dampening

Displays BGP dampening information

EXAMPLE:

```
BigIron# debug ip bgp dampening
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp dampening** command.

```
BGP: (1.1.1.1) dampening - route down 3.3.3.0/24
     Old Dampening: state was <*>, reuse_list_index=38, penalty=929, time=48,
flaps=1
     New state <h>, penalty=1893, reuse_list_index=43, offset=44
BGP: (1.1.1.1) Dampening - Route 3.3.3.0/24 up
     State was <h>, penalty=1893, time=390, flaps=2
     New state <*> penalty=1396, reuse_list_index=82, curr_offset=83
BGP: (1.1.1.100) Free Dampening 3.3.3.0/24

Total number of IP routes: 1
Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
      Destination       NetMask           Gateway           Port   Cost   Type
1     1.1.1.0           255.255.255.0     0.0.0.0           1      1      D
```

Syntax: [no] debug ip bgp dampening

Possible values: N/A

Default value: N/A

debug ip bgp events

Displays messages when BGP-related events occur. BGP-related events include starting or stopping a peer and opening or closing a BGP TCP connection.

EXAMPLE:

```
BigIron# debug ip bgp events
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp events** command.

```
BGP: 3.3.3.1 start peer
BGP: 3.3.3.1 stop peer
BGP: 3.3.3.1 BGP-TCP Connection opened
BGP: 3.3.3.1 TCP_OPEN done
BGP: 3.3.3.1 keep alive timer expired
```

Syntax: [no] debug ip bgp events

Possible values: N/A

Default value: N/A

debug ip bgp in

Displays BGP inbound information.

EXAMPLE:

```
BigIron# debug ip bgp in
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp in** command.

```
BGP: rcvd message KEEPALIVE_MESSAGE from peer 1.1.1.100, length (incl. header) 19
BGP: rcvd message UPDATE from peer 1.1.1.100, length (incl. header) 27
BGP: rcvd message OPEN_MESSAGE from peer 1.1.1.100, length (incl. header) 29
```

Syntax: [no] debug ip bgp in

Possible values: N/A

Default value: N/A

debug ip bgp keepalives

Displays BGP keepalive information

EXAMPLE:

```
BigIron# debug ip bgp keepalives
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp keepalives** command.

```
BGP: send keepalives to peer 3.3.3.100
```

Syntax: [no] debug ip bgp keepalives

Possible values: N/A

Default value: N/A

debug ip bgp out

Displays BGP outbound information.

EXAMPLE:

```
BigIron# debug ip bgp out
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp out** command.

```
BGP: send UPDATE message to peer 1.1.1.100, length (incl. header) 19
BGP: send KEEPALIVE_MESSAGE message to peer 1.1.1.100, length (incl. header) 19
BGP: send OPEN_MESSAGE message to peer 1.1.1.100, length (incl. header) 19
```

Syntax: [no] debug ip bgp out

Possible values: N/A

Default value: N/A

debug ip bgp updates

Displays BGP update information for all neighbors or those specified in an IP prefix list.

EXAMPLE:

```
BigIron# debug ip bgp updates
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip bgp updates** command.

```
BGP: 3.3.3.100 rcvd UPDATE 4.4.4.0/24
BGP: 3.3.3.100 rcvd UPDATE about 4.4.4.0/24 -- withdrawn
```

Syntax: [no] debug ip bgp updates [<prefix-list>]

Possible values: The <prefix-list> parameter specifies an IP prefix list. Only the routes permitted by the prefix list are displayed.

Default value: N/A

debug ip dvmrp detail

Displays detailed messages about DVMRP events, including sending reports, updating the forwarding table, and inserting table entries.

EXAMPLE:

```
BigIron# debug ip dvmrp detail
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip dvmrp detail** command.

```
DVMRP: send report DVMRP report to 224.0.0.4
DVMRP: send report DVMRP report to 2.2.2.1
DVMRP: updating fwd table due to a child is deleted
DVMRP: updating fwd table due to a entry is deleted
DVMRP: updating fwd table due to adding entry
DVMRP: insert entry source 1.1.1.0  group 239.255.162.2
```

Syntax: [no] debug ip dvmrp detail

Possible values: N/A

Default value: N/A

debug ip dvmrp in

Displays messages related to inbound DVMRP information.

EXAMPLE:

```
BigIron# debug ip dvmrp in
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip dvmrp in** command.

```
DVMRP: accept report. src ip 2.2.2.1 dest ip 224.0.0.4 group 0.6.5.3 port 7
DVMRP: accept probe. src ip 2.2.2.1 dest ip 224.0.0.4 group 0.6.5.3 port 7
DVMRP: accept prune. src ip 2.2.2.1 dest ip 2.2.2.100 group 0.6.5.3 port 7
```

Syntax: [no] debug ip dvmrp in

Possible values: N/A

debug ip dvmrp out

Displays messages related to outbound DVMRP information.

EXAMPLE:

```
BigIron# debug ip dvmrp out
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip dvmrp out** command.

```
DVMRP: send report. src ip 2.2.2.1 dest ip 224.0.0.4
DVMRP: send probe. src 2.2.2.1 dest 2.2.2.100 port 7
```

Syntax: [no] debug ip dvmrp out

Possible values: N/A

debug ip dvmrp pruning

Displays DVMRP pruning information.

EXAMPLE:

```
BigIron# debug ip dvmrp pruning
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip dvmrp pruning** command.

```
DVMRP: delete entry 00000003 idx 273
DVMRP: delete all entries for source 1.1.1.0
DVMRP: update fwd table by adding group 239.255.162.1 router 3.3.3.100 interface 9
DVMRP: update fwd table by adding group 239.255.162.2 router 3.3.3.100 interface 9
DVMRP: update fwd table by deleting group 239.255.162.1 router 3.3.3.100 interface 9
DVMRP: dvmrp delete prune state: Int6 Index 255 Prune Index 3
```

Syntax: [no] debug ip dvmrp pruning

Possible values: N/A

Default value: N/A

debug ip icmp events

Displays messages when ICMP events, including sending and receiving ICMP echo requests, occur.

EXAMPLE:

```
BigIron# debug ip icmp events
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip icmp events** command.

```
ICMP: rcvd echo request packet of length 40 from 1.1.1.2
ICMP: send echo request packet of length 60 to 1.1.1.2
```

Syntax: [no] debug ip icmp events

Possible values: N/A

Default value: N/A

debug ip icmp packets

Displays information related to ICMP packets sent or received on the device.

EXAMPLE:

```
BigIron# debug ip icmp packets
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip icmp packets** command.

```
ICMP:dst (1.2.3.4), src (0.0.0.0) echo request type
```

Syntax: [no] debug ip icmp packets

Possible values: N/A

Default value: N/A

debug ip igmp

Displays IGMP related information.

EXAMPLE:

```
BigIron# debug ip igmp
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip igmp** command.

```
IGMP: send message to 1.1.1.1 port ethernet 1 type 17 size 28
IGMP: send query to all port. type 17 port ethernet 7 ver 2
IGMP: rcvd v2 membership report from 1.1.1.2 group address 239.255.162.1 port ethernet
1 size 8
IGMP: rcvd membership query from 2.2.2.100 group address 0.0.0.0 port ethernet 7 size 8
IGMP: rcvd pim from 2.2.2.100 group address 16.0.0.0 port ethernet 7 size 12
```

Layer 2 Switch Output

The output generated by the **debug ip igmp** command is different on Layer 2 Switches. The following message is displayed whenever the Layer 2 Switch sends out a query packet. One message per VLAN is shown.

```
QUERY packet sent
```

The following message is displayed each time the aging process starts on the Layer 2 Switch and finds a multicast group that has aged out on a port:

```
REMOVING group 239.255.162.5 port 4/15
```

The following messages can also appear when the **debug ip igmp** command is entered on a Layer 2 Switch:

```
REPORT from 192.168.2.120 port 4/15 to 239.255.162.5 on vlan 1
LEAVE from 192.168.2.120 port 4/15 to 239.255.162.5 on vlan 1
QUERY  from 192.168.2.1 port 4/6 to 224.0.0.1 on vlan 1
DVMRP packet from 192.168.2.11 on port 2/8 on vlan 1
PIM_V1 packet from %192.168.2.11 on port 2/8 on vlan 1
PIM_V2 packet from %192.168.2.11 on port 2/8 on vlan 1
```

Syntax: [no] debug ip igmp

Possible values: N/A

Default value: N/A

debug ip msdp alarms

Displays information about MSDP alarms.

EXAMPLE:

```
BigIron# debug ip msdp alarms
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip msdp alarms** command.

```
MSDP: S=xxxxxxx P=0 Initiate Transport Connection to MSDP peer
```

Syntax: [no] debug ip msdp alarms

Possible values: N/A

Default value: N/A

debug ip msdp events

Displays messages when significant MSDP events occur.

EXAMPLE:

```
BigIron# debug ip msdp events
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip msdp events** command.

```
MSDP: 172.16.2.4: Closing session
MSDP: 172.16.2.4: Peer back to IDLE state
MSDP: (172.16.2.4) START peer
MSDP: 172.16.2.4: Closing session
MSDP: 172.16.2.4: Peer back to IDLE state
MSDP: Originating SA
MSDP: (172.16.2.4) START peer
MSDP: 172.16.2.4: TCP Connection to Remote Peer is Open
MSDP: 172.16.2.4: MSDP-TCP Connection opened
MSDP: 172.16.2.4: TCP_OPEN DONE, State 4
MSDP: Remote Peer closed TCP connection
```

Syntax: [no] debug ip msdp events

Possible values: N/A

Default value: N/A

debug ip msdp message

Displays information when MSDP messages are sent or received on the device.

EXAMPLE:

```
BigIron# debug ip msdp message
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip msdp message** command.

```
MSDP: 172.16.2.4: send keepalive message
MSDP: 172.16.2.4: TLV 4 Send Message to peer. length=3
MSDP:  P=0 MSDP Header Rcvd: Len=3 Type=4
MSDP: 172.16.2.4: KEEP_ALIVE Received Type 00000004 State=4 Length=3
MSDP: 172.16.2.4: send keepalive message
MSDP: 172.16.2.4: TLV 4 Send Message to peer. length=3
MSDP:  P=0 MSDP Header Rcvd: Len=3 Type=4
MSDP: 172.16.2.4: KEEP_ALIVE Received Type 00000004 State=4 Length=3
```

Syntax: [no] debug ip msdp message

Possible values: N/A

Default value: N/A

debug ip nat icmp

Displays information about ICMP packets whose source or destination matches a specified IP address.

EXAMPLE:

```
BigIron# debug ip nat icmp 10.10.100.18
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip nat icmp** command.

```
NAT: icmp src 10.10.100.18 => trans 192.168.2.79 dst 204.71.202.127
NAT: 192.168.2.79 204.71.202.127 ID 35768 len 60 txfid 13 icmp (8/0/512/519)
NAT: 204.71.202.127 10.10.100.18 ID 11554 len 60 txfid 15 icmp (0/0/512/519)
```


Syntax: [no] debug ip nat icmp <ip-addr>

Possible values: A valid IP address. An IP address of 0.0.0.0 matches any ICMP packet.

Default value: N/A

debug ip nat udp

Displays information about UDP packets whose source or destination matches a specified IP address.

EXAMPLE:

```
BigIron# debug ip nat udp 10.10.100.18
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip nat udp** command.

```
NAT: udp src 10.10.100.18:1561 => trans 192.168.2.79:65286 dst 192.168.3.11:53
NAT: 192.168.2.79:65286 192.168.3.11:53 ID 35512 len 58 txfid 13
NAT: 192.168.3.11:53 10.10.100.18:1560 ID 8453 len 346 txfid 15
```

Syntax: [no] debug ip nat udp <ip-addr>

Possible values: A valid IP address. An IP address of 0.0.0.0 matches any UDP packet.

Default value: N/A

debug ip nat tcp

Displays information about TCP packets whose source or destination matches a specified IP address.

EXAMPLE:

```
BigIron# debug ip nat tcp 10.10.100.18
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip nat tcp** command.

```
NAT: tcp src 10.10.100.18:1473 => trans 192.168.2.78:8016 dst 192.168.2.158:53
NAT: 192.168.2.78:8016 192.168.2.158:53 flags S ID 57970 len 44 txfid 13
NAT: 192.168.2.158:53 10.10.100.18:1473 flags S A ID 22762 len 44 txfid 15
NAT: 192.168.2.78:8016 192.168.2.158:53 flags A ID 58226 len 40 txfid 13
NAT: 192.168.2.78:8016 192.168.2.158:53 flags A ID 58482 len 77 txfid 13
NAT: 192.168.2.158:53 10.10.100.18:1473 flags A ID 23018 len 42 txfid 15
NAT: 192.168.2.78:8016 192.168.2.158:53 flags A ID 58738 len 40 txfid 13
NAT: 192.168.2.158:53 10.10.100.18:1473 flags A ID 23274 len 131 txfid 15
NAT: 192.168.2.78:8016 192.168.2.158:53 flags FA ID 58994 len 40 txfid 13
NAT: 192.168.2.158:53 10.10.100.18:1473 flags A ID 23530 len 40 txfid 15
NAT: 192.168.2.158:53 10.10.100.18:1473 flags FA ID 23786 len 40 txfid 15
NAT: 192.168.2.78:8016 192.168.2.158:53 flags A ID 59250 len 40 txfid 13
```

Syntax: [no] debug ip nat tcp <ip-addr>

Possible values: A valid IP address. An IP address of 0.0.0.0 matches any TCP packet.

Default value: N/A

debug ip nat transdata

Displays information about network translation requests and responses.

EXAMPLE:

```
BigIron# debug ip nat transdata
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip nat transdata** command.

```
NAT: icmp src 10.10.100.18:2048 => trans 192.168.2.79 dst 204.71.202.127
NAT: udp src 10.10.100.18:1561 => trans 192.168.2.79:65286 dst 192.168.3.11:53
NAT: tcp src 10.10.100.18:1473 => trans 192.168.2.78:8016 dst 192.168.2.158:53
```

Syntax: [no] debug ip nat transdata

Possible values: N/A

Default value: N/A

debug ip ospf adj

Displays information related to OSPF adjacency events. Adjacency events include adding or removing an interface, receiving hello messages from an adjacency, and broadcasting hello messages to an adjacency.

EXAMPLE:

```
BigIron# debug ip ospf adj
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf adj** command.

```
OSPF: 1.1.1.100 is added to interface neighbor list
OSPF: 4.4.4.101 is removed from interface neighbor list
OSPF: rvcd hello from 207.95.6.146 area 1 from 207.9
OSPF: broadcast hello to area 1 of all neighbors of 207.95.6.52
```

Syntax: [no] debug ip ospf adj

Possible values: N/A

Default value: N/A

debug ip ospf events

Displays messages when significant OSPF events occur. These events include backup designated router (BDR) election, designated router (DR) election, and receiving and sending database description (DBD) packets.

EXAMPLE:

```
BigIron# debug ip ospf events
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf events** command.

```
OSPF: DR/BDR election for 1.1.1.1 on ve 2
OSPF: elect BDR(backup designated router): Router ID 1.1.1.10 IP interface 1.1.1.10
OSPF: elect DR(designated router): Router ID 1.1.1.1, IP interface 1.1.1.1
OSPF: rcvd DBD from 1.1.1.1 on ve 2 flag 0x0 len 32 mtu 1500
OSPF: send DBD to 1.1.1.1 on ve 2 flag 0x0 len 232
```

Syntax: [no] debug ip ospf events

Possible values: N/A

Default value: N/A

debug ip ospf flood

Displays OSPF link state advertisement (LSA) flooding information.

EXAMPLE:

```
BigIron# debug ip ospf flood
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf flood** command.

```
OSPF: flooding 1 advertisement out interface 207.95.6.52
OSPF: attempting to flood rcvd LSA area = 00000001 interface type = 1
OSPF: flood advertisement throughout the entire autonomous system
```

Syntax: [no] debug ip ospf flood

Possible values: N/A

Default value: N/A

debug ip ospf lsa-generation

Displays information related to OSPF link state advertisements (LSAs).

EXAMPLE:

```
BigIron# ip ospf lsa-generation
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf lsa-generation** command.

```
OSPF: rcvd LSA type = 5, router ID 207.95.6.0  seq_num = 80000058
OSPF: ospf ls acknowledgement packet received!
OSPF: processing advertisement
```

Syntax: [no] debug ip ospf lsa-generation

Possible values: N/A

Default value: N/A

debug ip ospf packet

Displays information about OSPF packets sent and received on the device

EXAMPLE:

```
BigIron# debug ip ospf packet
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf packet** command.

```
OSPF: rcvd. v:2 t:1 l:48 rid:207.95.6.146
      aid:207.95.6.146 chk:00007920 aut:0 auk:00000000 00000000
OSPF: send v:2 t:1 l:48 rid:1.1.1.1
      aid:1.1.1.1 chk:0000F630 aut:0 auk:00000000 00000000
```

Table 3.3 describes the contents of **debug ip ospf packet** messages.

Table 3.3: Output from the debug ip ospf packet command

This Field...	Displays...
rcvd. or send	Indicates whether the packet was sent or received.

Table 3.3: Output from the debug ip ospf packet command (Continued)

This Field...	Displays...
v:	OSPF version.
t:	OSPF packet type. Possible packet types are: 1 – Hello 2 – Data description 3 – Link state request 4 – Link state update 5 – Link state acknowledgment
l:	OSPF packet length in bytes.
rid:	OSPF router ID.
aid:	OSPF area ID.
chk:	OSPF checksum.
aut:	OSPF authentication type. Possible authentication types are: 0 – No authentication 1 – Simple password 2 – MD5
auk:	OSPF authentication key.

Syntax: [no] debug ip ospf packet

Possible values: N/A

Default value: N/A

debug ip ospf retransmission

Displays OSPF retransmission related events.

EXAMPLE:

```
BigIron# debug ip ospf retransmission
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf retransmission** command.

```
OSPF: examine each neighbor and add advertisement to the retransmission list if
necessary
OSPF: remove current database copy from all neighbors retransmission lists
```

Syntax: [no] debug ip ospf retransmission

Possible values: N/A

Default value: N/A

debug ip ospf spf

Displays information about shortest path first (SPF) or Dijkstra algorithm related OSPF events. This command lists new routing table entries when they are added, as well as the updated routing table.

EXAMPLE:

```
BigIron# debug ip ospf spf
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ospf spf** command.

```
OSPF: Running dijksttra for area 1
OSPF: Adding routing table entry for transit network 207.95.6.146
OSPF: adding stub networks for area 1

OSPF: New routing table:
OSPF: ---Entry #1
OSPF: destination 1.1.1.0, mask 255.255.255.0, type 0
OSPF: area 0.0.0.1 path cost 1, type 0
OSPF: next hop router 15.212.4.123, outgoing interface loopback 22
OSPF: advertising router 1.1.1.1
OSPF: ---Entry #2
OSPF: destination 4.4.4.0, mask 255.255.255.0, type 0
OSPF: area 0.0.0.1 path cost 1, type 0
OSPF: next hop router 16.148.4.123, outgoing interface loopback 22
OSPF: advertising router 1.1.1.1
```

(remaining routing table entries omitted)

Syntax: [no] debug ip ospf spf

Possible values: N/A

Default value: N/A

debug ip pim <address>

Displays information about PIM traffic related. Messages are displayed when hello, join, graft, and prune messages are sent or received.

EXAMPLE:

```
BigIron# debug ip pim 239.255.162.6
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip pim <address>** command.

```
PIM: send prune e7, source 1.1.1.2 group 239.255.162.6 nbr 2.2.2.1
PIM: rcvd prune e7, Source 1.1.1.2 group 239.255.162.6
PIM: send graft e7, source 1.1.1.2 group 239.255.162.6 nbr 2.2.2.1
PIM: rcvd graft e7, source 3.3.3.1 group 239.255.162.6
```

Syntax: [no] debug ip pim [<ip-addr>]

Possible values: Valid PIM group address.

Default value: N/A

debug ip pim events

Displays messages when PIM events, including deleting and adding group entries, occur.

EXAMPLE:

```
BigIron# debug ip pim events
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip pim events** command.

```
PIM: BEGIN Periodic join-prune msgs
PIM: END Periodic join-prune msgs
PIM: delete group 239.255.162.2
PIM: Begin sending Join/Prune msg to e7
PIM: delete group entry 239.255.162.2 port ethernet 1
```

Syntax: [no] debug ip pim events

Possible values: N/A

Default value: N/A

debug ip rip

Displays information about RIP routing transactions.

EXAMPLE:

```
BigIron# debug ip rip
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip rip** command.

```
RIP: sending updates(periodic) to 1.1.1.255 via ethernet 7 (1.1.1.100)
RIP: sending updates(triggered) to 1.1.1.255 via ethernet 7 (1.1.1.100)
RIP: rcvd updates from 1.1.1.100 on ethernet 7
```

Syntax: [no] debug ip rip

Possible values: N/A

Default value: N/A

debug ip rip database

Displays information about routes imported from other routing protocols, such as OSPF and BGP.

EXAMPLE:

```
BigIron# debug ip rip database
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip rip database** command.

```
RIP: process response packet
         header: type:RESPONSE PACKET, version:1

RIP: remove imported route
      Network Address   NetMask           Gateway           Port   Cost   Type
      7.7.7.0           255.255.255.0    *2.2.2.100         v3     2      O
      7.7.7.0           255.255.255.0     3.3.3.100         v4     2      O

RIP: add imported OSPF route

Total number of IP routes: 14
Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
      Destination       NetMask           Gateway           Port   Cost   Type
1     1.0.0.0           255.0.0.0         207.95.6.146      v8     0      B
2     1.1.1.0           255.255.255.0     0.0.0.0           v2     1      D
3     2.0.0.0           255.0.0.0         1.1.1.100         v2     2      R
4     2.2.2.0           255.255.255.0     0.0.0.0           v3     1      D
5     3.0.0.0           255.0.0.0         1.1.1.100         v2     2      R
6     3.3.3.0           255.255.255.0     0.0.0.0           v4     1      D
7     4.0.0.0           255.0.0.0         207.95.6.146      v8     0      B
8     4.4.4.0           255.255.255.0     0.0.0.0           9      1      D
9     6.0.0.0           255.0.0.0         1.1.1.100         v2     2      R
10    6.6.6.0           255.255.255.0    *2.2.2.100         v3     2      O
      6.6.6.0           255.255.255.0     3.3.3.100         v4     2      O
11    7.0.0.0           255.0.0.0         1.1.1.100         v2     2      R
12    7.7.7.0           255.255.255.0    *2.2.2.100         v3     2      O
      7.7.7.0           255.255.255.0     3.3.3.100         v4     2      O
13    192.192.192.0     255.255.255.0     207.95.6.146      v8     20     O
14    207.95.6.0        255.255.255.0     0.0.0.0           v8     1      D
```

Syntax: [no] debug ip rip database

Possible values: N/A

Default value: N/A

debug ip rip events

Displays information about RIP events, including aged-out routes and replies sent to other routers.

EXAMPLE:

```
BigIron# debug ip rip events
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip rip events** command.

```
RIP: route to 6.0.0.0 via next hop 1.1.1.100 aged out
RIP: send all routes reply to 1.1.1.100
RIP: received response from 1.1.1.100: 164 bytes
         route entry: family:2, target:6.0.0.0, metric:1
        route entry: family:2, target:207.95.6.0, metric:1

RIP: New routing table
Total number of IP routes: 6
Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
      Destination       NetMask           Gateway           Port   Cost   Type
1     1.0.0.0           255.0.0.0         207.95.6.146      v8     0      B
2     1.1.1.0           255.255.255.0     0.0.0.0           v2     1      D
3     2.0.0.0           255.0.0.0         207.95.6.146      v8     0      B
4     2.2.2.0           255.255.255.0     0.0.0.0           v3     1      D
5     3.0.0.0           255.0.0.0         1.1.1.100         v2     2      R
6     3.3.3.0           255.255.255.0     0.0.0.0           v4     1      D
```

Syntax: [no] debug ip rip events

Possible values: N/A

Default value: N/A

debug ip rip trigger

Displays information about RIP events triggered by adding or deleting a route.

EXAMPLE:

```
BigIron# debug ip rip trigger
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip rip trigger** command.

```
RIP: adding route to target:3.0.0.0 via gateway:1.1.1.9, metric: 2, port: 8, bits: 8
RIP: deleting route to target:3.0.0.0 via gateway:1.1.1.9
RIP: build route header: type:RESPONSE PACKET, version:1
RIP: build route entry: family:2, target:207.95.6.0, metric:1
RIP: periodic update sent on port 18
```

Syntax: [no] debug ip rip trigger

Possible values: N/A

Default value: N/A

debug ip ssh

Displays the status of SSH session negotiation.

EXAMPLE:

```
BigIron# debug ip ssh
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip ssh** command.

```
SSH: Server successfully sent to client its version number
SSH: Server received client's version number
SSH: client's version number SSH-1.5
SSH: Server version number matches client's version number
SSH: Server sent its host and server public keys to the client
SSH: Server received session key from the client
SSH: Server received client's name
SSH: Server authenticated the client with password
SSH: Client requested compression
SSH: Secure Shell is established!
```

Syntax: [no] debug ip ssh

Possible values: N/A

Default value: N/A

debug ip tcp <address>

Displays information about TCP packets from a specified IP address.

EXAMPLE:

```
BigIron# debug ip tcp 192.168.9.210
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip tcp <address>** command.

```
TCP: rcvd packet (len=20) 192.168.9.210:3669 -> 192.168.9.2:23

packet:syn:0,ack:1,rst:0,fin:1,hlen:5,chksum:00006fdf,seqn:2423494362,ackn:211
TCP: sent packet (len=40) 192.168.9.2:23 -> 192.168.9.210:3669
     packet: syn:0,ack:0,rst:1,fin:0,hlen:5,chksum:0000b93d,seqn:21521,ackn:0
TCP: sent packet 192.168.9.2:23 -> 192.168.9.210:3669
     packet: syn:0,ack:0,rst:1,fin:0,hlen:5,chksum:0000b93d,seqn:21521,ackn:0
```

Syntax: [no] debug ip tcp <address>

Possible values: IP address

Default value: N/A

debug ip tcp driver

Displays information about TCP driver related events, such as opening, closing, and aborting a TCP connection, or discarding TCP packets.

EXAMPLE:

```
BigIron# debug ip tcp driver
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip tcp driver** command.

```
TCP: aborting connection 1.1.1.1:23 -> 1.1.1.2:2559
TCP: closing connection 1.1.1.1:23 -> 1.1.1.2:2559
TCP: opening connection 207.95.6.52:3456 -> 207.95.6.146:23
```

Syntax: [no] debug ip tcp driver

Possible values: N/A

Default value: N/A

debug ip tcp memory

The **debug ip tcp memory** command causes messages to be displayed when memory is allocated or deallocated to the internal TCP buffers.

EXAMPLE:

```
BigIron# debug ip tcp memory
```

For example, when a user establishes a Telnet session with the device, an then terminates it, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip tcp memory** command.

```
TCP TCB ALLOCATED 210de822
TCP SEND BUFFER ALLOCATED 2111ec80
TCP SEND QUEUE BUFFER ALLOCATED 210d88dc
TCP SEND BUFFER ALLOCATED 2113695c
TCP SEND QUEUE BUFFER ALLOCATED 210d9714
TCP SEND BUFFER ALLOCATED 2111f838
TCP SEND QUEUE BUFFER ALLOCATED 210d894c
TCP SEND BUFFER ALLOCATED 21117174
TCP SEND QUEUE BUFFER ALLOCATED 210d8444
TCP SEND BUFFER ALLOCATED 210f4aac
TCP SEND QUEUE BUFFER ALLOCATED 210d6fb4
TCP SEND BUFFER ALLOCATED 210f5088
TCP SEND QUEUE BUFFER ALLOCATED 210d6fec
TCP SEND BUFFER FREED 2111ec80
TCP QUEUE BUFFER FREED 210d6fec
TCP RECEIVE QUEUE BUFFER ALLOCATED 210d6fec
TCP RECEIVE BUFFER ALLOCATED 21151530
TCP RECEIVE BUFFER FREED 21151530
TCP QUEUE BUFFER FREED 210d6fec
TCP RECEIVE QUEUE BUFFER ALLOCATED 210d6fec
TCP RECEIVE BUFFER ALLOCATED 21151530
TCP RECEIVE BUFFER FREED 21151530
TCP QUEUE BUFFER FREED 210d6fec
TCP TCB FREED 210de822
```

Syntax: [no] debug ip tcp memory

NOTE: Output from this command appears only on the console or syslog. The output is suppressed when sent to a Telnet or SSH session.

Possible values: N/A

Default value: N/A

debug ip tcp packet

Displays information about received and sent TCP packets.

EXAMPLE:

```
BigIron# debug ip tcp packet
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip tcp packet** command.

```
TCP: rcvd packet (len=20) 1.1.1.2:2526 -> 1.1.1.1:23
 packet:syn:0,ack:1,rst:0,fin:0,hlen:5,chksum:0000c34e,seqn:55807198,ackn:548539276
TCP: sent packet (len=20) 207.95.6.52:8104 -> 207.95.6.146:179
packet:syn:0,ack:1,rst:0,fin:0,hlen:5,chksum:00008b4a,seqn:36182260,ackn:2027586739
```

Syntax: [no] debug ip tcp packet

Possible values: N/A

Default value: N/A

debug ip tcp sack

Displays information about TCP Selective-ACK packets.

EXAMPLE:

```
BigIron# debug ip tcp sack
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip tcp sack** command.

```
TCP: process ACK, tcp state tcp_syn_recd
TCP: nothing to ACK, sequence number 21521, tcp is in sequence
TCP: process ACK, tcp state tcp_close_wait
```

Syntax: [no] debug ip tcp sack

Possible values: N/A

Default value: N/A

debug ip tcp transactions

Displays information about TCP transactions, including state changes and packet retransmissions.

EXAMPLE:

```
BigIron# debug ip tcp transactions
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip tcp transactions** command.

```
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change LISTEN -> SYN-RECEIVED
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change SYN-RECEIVED -> ESTABLISHED
TCP: retransmitted segment
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change ESTABLISHED -> FIN-WAIT-1
TCP: retransmitted segment
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change FIN-WAIT-1 -> FIN-WAIT-2
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change FIN-WAIT-2 -> TIME-WAIT
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change TIME-WAIT -> CLOSED
```

Syntax: [no] debug ip tcp transactions

Possible values: N/A

Default value: N/A

debug ip udp

Displays information about UDP packets.

EXAMPLE:

```
BigIron# debug ip udp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip udp** command.

```
UDP: sent src 1.1.168.192(port 161) -> dest 181.1.168.192(port 162), length:71
UDP: rcvd src 234.1.168.192(port 138) -> dest 255.1.168.192(port 138), length:209
```

Syntax: [no] debug ip udp

Possible values: N/A

Default value: N/A

debug ip vrrp events

Displays information about VRRP events, such as when a backup router transitions to a master, a router transitions to a backup router, a VRID is deleted, or a VRRP packet is dropped.

EXAMPLE:

```
BigIron# debug ip vrrp events
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip vrrp events** command.

```
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change LISTEN -> SYN-RECEIVED
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change SYN-RECEIVED -> ESTABLISHED
TCP: retransmitted segment
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change ESTABLISHED -> FIN-WAIT-1
TCP: retransmitted segment
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change FIN-WAIT-1 -> FIN-WAIT-2
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change FIN-WAIT-2 -> TIME-WAIT
TCP: 1.1.1.1:23 -> 1.1.1.2:2537: state change TIME-WAIT -> CLOSED
```

Syntax: [no] debug ip vrrp events

Possible values: N/A

Default value: N/A

debug ip vrrp packet

Displays information about VRRP packets and the IP addresses of backup routers.

EXAMPLE:

```
BigIron# debug ip vrrp packet
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug ip vrrp events** command.

```
VRRP: rcvd ver:2 type:1 vrid:1 pri:255 #ip:1 aut:0 adv:1 chk:56825
  Num of ip addr 1
      1.1.1.1   from sender 1.1.1.1
VRRP: send advertise! ver:2 type:1 vrid:1 pri:255 #ip:1 aut:0 adv:1 chk:56825
  Num of ip addr 1
  1.1.1.1
```


Table 3.4 describes the contents of **debug ip vrrp packet** messages.

Table 3.4: Output from the debug ip vrrp packet command

This Field...	Displays...
rcvd. or send	Indicates whether the packet was sent or received.
ver:	VRRP version; RFC 2338 defines version 2.
type:	VRRP packet type. Possible packet types are: 1 Advertisement
vrid:	Virtual Router Identifier.
pri:	Priority of the VRRP router.
#ip:	The number of IP addresses contained in this VRRP advertisement.
aut:	VRRP authentication type. Possible authentication types are: 0 No authentication 1 Simple text password 2 IP Authentication Header
adv:	
chk:	VRRP checksum.
Num of ip addr	

Syntax: [no] debug ip vrrp packet

Possible values: N/A

Default value: N/A

debug isis l1-csnp

Displays information about Level-1 Complete Sequence Number PDUs (CSNPs) sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l1-csnp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l1-csnp** command.

```
ISIS: Sending L1 CSNP on 2/24, length 1497
ISIS: Received L1 CSNP on 2/24, length 256 from 0004.8026.b337
```

Syntax: [no] debug isis l1-csnp

Possible values: N/A

Default value: N/A

debug isis l1-hello

Displays information about Level-1 Hello PDUs sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l1-hello
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l1-hello** command.


```
ISIS: Received L1 LAN IIH on 2/24, length 256 from 0004.8026.b337
ISIS: Sending L1 LAN IIH on 2/24, length 1497
ISIS: Received L1 LAN IIH on 2/24, length 256 from 0004.8026.b337
```

Syntax: [no] debug isis l1-hello

Possible values: N/A

Default value: N/A

debug isis l1-lsp

Displays information about Level-1 Link State PDUs (LSPs) sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l1-lsp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l1-lsp** command.

```
ISIS: Sending L1 LSP on 2/24, length 27
ISIS: Received L1 LSP on 2/24, length 256 from 0004.8026.b337
```

Syntax: [no] debug isis l1-lsp

Possible values: N/A

Default value: N/A

debug isis l1-psnp

Displays information about Level-1 Partial Sequence Number PDUs (PSNPs) sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l1-psnp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l1-psnp** command.

```
ISIS: Received L2 PSNP on 2/24, length 256
ISIS: Sending L1 PSNP on 2/24, length 35
```

Syntax: [no] debug isis l1-psnp

Possible values: N/A

Default value: N/A

debug isis l2-csnp

Displays information about Level-2 Complete Sequence Number PDUs (CSNPs) sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l2-csnp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l2-csnp** command.

```
ISIS: Sending L2 CSNP on 2/24, length 1497
ISIS: Received L2 CSNP on 2/24, length 256 from 0004.8026.b337
```

Syntax: [no] debug isis l2-csnp

Possible values: N/A

Default value: N/A

debug isis l2-hello

Displays information about Level-2 Hello PDUs sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l2-hello
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l2-hello** command.

```
ISIS: Received L2 LAN IIH on 2/24, length 256 from 0004.8026.b337
ISIS: Sending L2 LAN IIH on 2/24, length 1497
ISIS: Received L2 LAN IIH on 2/24, length 256 from 0004.8026.b337
```

Syntax: [no] debug isis l2-hello

Possible values: N/A

Default value: N/A

debug isis l2-lsp

Displays information about Level-2 Link State PDUs (LSPs) sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l2-lsp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l2-lsp** command.

```
ISIS: Sending L2 LSP on 2/24, length 27
ISIS: Received L2 LSP on 2/24, length 256 from 0004.8026.b337
```

Syntax: [no] debug isis l2-lsp

Possible values: N/A

Default value: N/A

debug isis l2-psnp

Displays information about Level-2 Partial Sequence Number PDUs (PSNPs) sent and received on the device.

EXAMPLE:

```
BigIron# debug isis l2-psnp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis l2-psnp** command.

```
ISIS: Received L2 PSNP on 2/24, length 256
ISIS: Sending L1 PSNP on 2/24, length 35
```

Syntax: [no] debug isis l2-psnp

Possible values: N/A

Default value: N/A

debug isis memory

Displays messages when memory is allocated or deallocated for IS-IS functions.

EXAMPLE:

```
BigIron# debug isis memory
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis memory** command.

```
ISIS: Memory Allocated for buffer descriptor at 21a54ad8
ISIS: Memory Allocated for packet-buffer at 211e1680
ISIS: Memory Released for buffer descriptor  at 21a54ad8
ISIS: Memory Allocation for circuit IP address failed
```

Syntax: [no] debug isis memory

Possible values: N/A

debug isis pp-hello

Displays information about Point-to-Point Hello PDUs sent and received on the device.

EXAMPLE:

```
BigIron# debug isis pp-hello
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis pp-hello** command.

```
ISIS: Sending PTP IIH on 9/1, length 1492
ISIS: Received PTP IIH on 9/1, length 256
```

Syntax: [no] debug isis pp-hello

Possible values: N/A

debug isis ppp

Displays information about OSI PPP packets sent and received on the device.

EXAMPLE:

```
BigIron# debug isis ppp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis ppp** command.

```
ISIS PPP: sending isis packet on pos port  512
ISIS: osicp datainput rx pkt length 1492 on unit 32
ISIS: Received PTP IIH on 9/1, length 256
ISIS: Sending PTP IIH on 9/1, length 1492
ISIS PPP: sending isis packet on pos port  512
```

Syntax: [no] debug isis ppp

Possible values: N/A

debug isis redistribution

Displays information about redistributed routes.

debug isis route-table

Displays messages when changes are made to the IS-IS route table.

EXAMPLE:

```
BigIron# debug isis route-table
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis route-table** command.

```
ISIS: Deleting route 13.10.0.0 255.255.0.0 level 2
ISIS: Deleting route 11.10.0.0 255.255.0.0 level 2
ISIS: Creating new route for 100.10.0.0 255.255.0.0 level 2 type 1
ISIS: Adding path Next hop = 192.147.201.100  Interface 2/4
ISIS: Creating new route for 12.10.0.0 255.255.0.0 level 2 type 1
```

Syntax: [no] debug isis route-table

Possible values: N/A

debug isis spf

Displays information about SPF calculations made for IS-IS.

EXAMPLE:

```
BigIron# debug isis spf
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis spf** command.

```
ISIS: Running Decision for level 1
ISIS: IsisgetMinTent
ISIS: Finished Decision for level 1
ISIS: Total Route Calculation Time for Level 1 is 0 miliseconds.
```

Syntax: [no] debug isis spf

Possible values: N/A

debug isis trace

Displays information about internal IS-IS functions occurring on the device.

EXAMPLE:

```
BigIron# debug isis trace
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug isis trace** command.

```
ISIS:proc_SNPE
ISIS:proc_SNPE
ISIS: build_csnp
ISIS: build_csnp
ISIS: sig_decision
```

Syntax: [no] debug isis trace

Possible values: N/A

debug spanning

Displays information about BPDU packets.

EXAMPLE:

```
BigIron# debug spanning
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages with the **no debug spanning** command.

```
ST: Port 2/1

 [A] [B][C][D]       [E]           [F]
0000 00 00 00 800000e052c37d40 00000000
```



```
          [G]       [H][I] [J]   [K]   [L]   [M]
800000e052c37d40 20 40 0000 0014 0002 000f
```

Table 3.5 describes the contents of **debug spanning** message. Note that the letters in brackets do not appear in the output.

Table 3.5: Output from the debug spanning command

This Field...	Displays...
ST:	Indicates that this is a spanning tree packet
Port 2/1	Interface receiving the packet
[A] 0000	Indicates that this is an IEEE BDPU packet.
[B] 00	Version number.
[C] 00	Command mode. This can be one of the following: 00 Config BPDU 80 Topology Change Notification BPDU
[D] 00	Acknowledgement of topology change. This can be one of the following: 00 No change 80 Change notification
[E] 800000e052c37d40	Root ID.
[F] 00000000	Root path cost.
[G] 800000e052c37d40	Bridge ID.
[H] 20	Port priority.
[I] 40	Port number.
[J] 0000	Message age in 1/256 seconds.
[K] 0014	Maximum age in 1/256 seconds.
[L] 0002	Hello time in 1/256 seconds.
[M] 000f	Forward delay in 1/256 seconds.

Syntax: [no] debug spanning

Possible values: N/A

Default value: N/A

mm

Displays the contents of a specified address on every module. This command is valid on the FastIron II and BigIron only.

EXAMPLE:

```
BigIron# mm 0190044c
(4)0190044c: 0000 0000 0000 0000 0000 0000 0000 0000
(4)0190045c: 0000 0000 0000 0000 0000 0000 0000 0000
(4)0190046c: 0000 0000 0000 0000 0000 0000 0000 0000
(4)0190047c: 0000 0000 0000 0000 0000 0000 0000 0000
```



```
1e90044c: 0000044c 00000450 00000454 00000458
1e90045c: 0000045c 00000460 00000464 00000468
1e90046c: 0000046c 00000470 00000474 00000478
1e90047c: 0000047c 00000480 00000484 00000488
```

Syntax: mm <address> [<length>]

Possible values: <length> can be up to 0x40 bytes.

Default value: If you do not specify the <length> parameter, 0x40 bytes are displayed.

phy

Displays information about PHY registers for a specified port. This command can be useful for resolving problems with NIC adapters that have linking problems.

EXAMPLE:

```
BigIron# phy 4/11
BCR reg 0, val = 1100
BSR reg 1, val = 7809
ID1 reg 2, val = 7810
ID2 reg 3, val = 0043
ANA reg 4, val = 01e1
ANLPA reg 5, val = 0000
ANE reg 6, val = 0000
MR reg  16, val = 0c00
IER reg  17, val = 0000
ISR reg 18, val = 4000
CR reg 19, val = 0000
CSR reg 20, val = 048b
```

```
/* Register 1: Basic Status Register (PHY_BSR_R) */
#define BSR_100BASE_T4        0x8000
#define BSR_100BASE_TX_FD     0x4000
#define BSR_100BASE_TX_HD     0x2000
#define BSR_10BASE_T_FD       0x1000
#define BSR_10BASE_T_HD       0x0800
#define BSR_AUTO_NEGO_DONE    0x0020
#define BSR_REMOTE_FAULT      0x0010
#define BSR_AUTO_NEGO_ABL     0x0008
#define BSR_LINK_UP           0x0004
```

```
/* Register 4: Auto-Negotiation Advertisement (PHY_ANA_R) */
#define ANA_NEXT_PAGE         0x8000
#define ANA_REMOTE_FAULT      0x2000
#define ANA_100BASE_T4        0x0200
#define ANA_100BASE_TX_FD     0x0100
#define ANA_100BASE_TX        0x0080
#define ANA_10BASE_T_FD       0x0040
#define ANA_10BASE_T          0x0020
#define ANA_SELECTOR_FIELD    0x001F
```

```
/* Register 5: Auto-Negotiation Link Partner Ability (PHY_ANLPA_R) */
#define ANL_NEXT_PAGE         0x8000
#define ANL_ACK               0x4000
#define ANL_REMOTE_FAULT      0x2000
#define ANL_100BASE_T4        0x0200
#define ANL_100BASE_TX_FD     0x0100
#define ANL_100BASE_TX        0x0080
#define ANL_10BASE_T_FD       0x0040
#define ANL_10BASE_T          0x0020
#define ANL_SELECTOR_FIELD    0x001F
```



```
/* Register 31: BASE-TX PHY Control (PHY_BPC_R) */
#define BPC_DISABLE_REC       0x2000
#define BPC_AUTO_NEG_CPL      0x1000
#define BPC_COMPENSAT_MASK    0x0C00
#define BPC_NO_COMPENSAT      0
#define BPC_HALF_COMPENSAT    0x0400
#define BPC_FULL_COMPENSAT    0x0800
#define BPC_AUTO_COMPENSAT    0x0C00
#define BPC_RLBEN             0x0200
#define BPC_DCREN             0x0100
#define BPC_NRZIEN            0x0080
#define BPC_4B5BEN            0x0040
#define BPC_TX_ISOLATE        0x0020
#define BPC_OPMODE_MASK       0x001C
#define BPC_OP_STILL_NEG      0x0000
#define BPC_OP_10B_HD         0x0004
#define BPC_OP_100B_HD        0x0008
#define BPC_OP_100B_T4        0x0010
#define BPC_OP_10B_FD         0x0014
#define BPC_OP_100B_FD        0x0018
#define BPC_OP_ISOLATE        0x001C
#define BPC_MLT3_DISAB        0x0002
#define BPC_SCRAMB_DISAB      0x0001
```

Syntax: phy <slot/port>

Possible values: <slot/port> must be a valid port on the switch or router.

Default value: N/A

ptrace aaa

Toggles tracing for AAA packets.

EXAMPLE:

```
BigIron# ptrace aaa
```

Syntax: ptrace aaa

Possible values: N/A

Default value: N/A

ptrace appletalk aarp

Toggles tracing for Appletalk Address Resolution Protocol (AARP) packets. When you enable this function, each time an AARP packet is encountered, a message appears on the console indicating whether the packet was transmitted or received, the port on which it was transmitted or received, and the data field of the packet.

EXAMPLE:

```
BigIron# ptrace appletalk aarp
```

Syntax: ptrace appletalk aarp

Possible values: N/A

Default value: N/A

ptrace appletalk aep

Toggles tracing for Appletalk Echo Protocol (AEP) packets. When you enable this function, each time an AEP packet is encountered, a message appears on the console indicating whether the packet was transmitted or received, the port on which it was transmitted or received, and the contents of the packet's Datagram Delivery Protocol (DDP) header.

EXAMPLE:

```
BigIron# ptrace appletalk aep
```

Syntax: ptrace appletalk aep

Possible values: N/A

Default value: N/A

ptrace appletalk nbp

Toggles tracing for Appletalk Name Binding Protocol (NBP) packets. When you enable this function, each time an NBP packet is encountered, a message appears on the console indicating whether the packet was transmitted or received, the port on which it was transmitted or received, and the contents of the packet's DDP header.

EXAMPLE:

```
BigIron# ptrace appletalk nbp
```

Syntax: ptrace appletalk nbp

Possible values: N/A

Default value: N/A

ptrace appletalk none

Disables tracing for all Appletalk packets.

EXAMPLE:

```
BigIron# ptrace appletalk none
```

Syntax: ptrace appletalk none

Possible values: N/A

Default value: N/A

ptrace appletalk rtmp

Toggles tracing for Appletalk Routing Table Maintenance Protocol (RTMP) packets. When you enable this function, each time an RTMP packet is encountered, a message appears on the console indicating whether the packet was transmitted or received, the port on which it was transmitted or received, and the contents of the packet's DDP header.

EXAMPLE:

```
BigIron# ptrace appletalk rtmp
```

Syntax: ptrace appletalk rtmp

Possible values: N/A

Default value: N/A

ptrace appletalk states

Toggles tracing for Appletalk state transition packets.

EXAMPLE:

```
BigIron# ptrace appletalk states
```

Syntax: ptrace appletalk states

Possible values: N/A

Default value: N/A

ptrace appletalk zip

Toggles tracing for Appletalk Zone Information Protocol (ZIP) packets. When you enable this function, each time a ZIP packet is encountered, a message appears on the console indicating whether the packet was transmitted or received, the port on which it was transmitted or received, and the contents of the packet's DDP header.

EXAMPLE:

```
BigIron# ptrace appletalk zip
```

Syntax: ptrace appletalk zip

Possible values: N/A

Default value: N/A

ptrace arp

Toggles tracing for ARP packets.

EXAMPLE:

```
BigIron# ptrace arp
```

Syntax: ptrace arp

Possible values: N/A

Default value: N/A

ptrace bootp

Toggles tracing for BOOTP packets.

EXAMPLE:

```
BigIron# ptrace bootp
```

Syntax: ptrace bootp

Possible values: N/A

Default value: N/A

ptrace dvmrp graft

Toggles tracing for DVMRP graft packets.

EXAMPLE:

```
BigIron# ptrace dvmrp graft
```

Syntax: ptrace dvmrp graft

Possible values: N/A

Default value: N/A

ptrace dvmrp graft-ack

Toggles tracing for DVMRP graft-ack packets.

EXAMPLE:

```
BigIron# ptrace dvmrp graft-ack
```

Syntax: ptrace dvmrp graft-ack

Possible values: N/A

Default value: N/A

ptrace dvmrp mcache

Toggles tracing for DVMRP mcache packets.

EXAMPLE:

```
BigIron# ptrace dvmrp mcache
```

Syntax: ptrace dvmrp mcache

Possible values: N/A

Default value: N/A

ptrace dvmrp message

Toggles tracing for DVMRP message packets.

EXAMPLE:

```
BigIron# ptrace dvmrp message
```

Syntax: ptrace dvmrp message

Possible values: N/A

Default value: N/A

ptrace dvmrp none

Disables tracing for DVMRP packets.

EXAMPLE:

```
BigIron# ptrace dvmrp none
```

Syntax: ptrace dvmrp none

Possible values: N/A

Default value: N/A

ptrace dvmrp probe

Toggles tracing for DVMRP probe packets.

EXAMPLE:

```
BigIron# ptrace dvmrp probe
```

Syntax: ptrace dvmrp probe

Possible values: N/A

Default value: N/A

ptrace dvmrp prune

Toggles tracing for DVMRP prune packets.

EXAMPLE:

```
BigIron# ptrace dvmrp prune
```

Syntax: ptrace dvmrp prune

Possible values: N/A

Default value: N/A

ptrace dvmrp route-table

Toggles tracing for DVMRP route-table packets.

EXAMPLE:

```
BigIron# ptrace dvmrp route-table
```

Syntax: ptrace dvmrp route-table

Possible values: N/A

Default value: N/A

ptrace icmp

Toggles tracing for ICMP packets.

EXAMPLE:

```
BigIron# ptrace icmp
```

Syntax: ptrace icmp

Possible values: N/A

Default value: N/A

ptrace igmp

Toggles tracing for IGMP packets.

EXAMPLE:

```
BigIron# ptrace igmp
```

Syntax: ptrace igmp

Possible values: N/A

Default value: N/A

ptrace ip

Toggles tracing for IP packets.

EXAMPLE:

```
BigIron# ptrace ip
```

Syntax: ptrace ip

Possible values: N/A

Default value: N/A

ptrace mpls rsvp

Displays information about the RSVP messages exchanged between this device and the next hop.

EXAMPLE:

```
BigIron# ptrace mpls rsvp
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages by entering the **ptrace mpls rsvp** command again.

```
Receive RSVP Path: src 20.1.1.1, dst 3.3.3.3 on 3/1
Send RSVP Path: src 20.1.1.1, dst 3.3.3.3 on 3/2, next hop 11.1.1.2
Receive RSVP Resv: src 11.1.1.2, dst 11.1.1.1 on 3/2
Send RSVP Resv: src 10.1.1.2, dst 10.1.1.1 on 3/1, next hop 10.1.1.1
Send RSVP Path: src 20.1.1.1, dst 3.3.3.3 on 3/2, next hop 11.1.1.2
Send RSVP Resv: src 10.1.1.2, dst 10.1.1.1 on 3/1, next hop 10.1.1.1
Receive RSVP Resv: src 11.1.1.2, dst 11.1.1.1 on 3/2
Receive RSVP Path: src 20.1.1.1, dst 3.3.3.3 on 3/1
Receive RSVP PathTear: src 20.1.1.1, dst 3.3.3.3 on 3/1
Send RSVP PathTear: src 20.1.1.1, dst 3.3.3.3 on 3/2, next hop 11.1.1.2
```

Syntax: ptrace mpls rsvp

Possible values: N/A

Default value: N/A

ptrace mpls rsvp detail-of-received

Displays the contents of each RSVP message received from next-hop devices.

EXAMPLE:

```
BigIron# ptrace mpls rsvp detail-of-received
```

After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages by entering the **ptrace mpls rsvp detail-of-received** command again.

```
Receive RSVP Path: src 20.1.1.1, dst 3.3.3.3 on 3/1
  version: 1, flags: 00000010, cksum: 0000571c, ttl: 62, length: 180
 SESSION                type 7 length 16:
  destination 3.3.3.3, tunnelid 1, ext tunnelid (source) 20.1.1.1
 RSVP_HOP               type 1 length 12:
  address: 11.1.1.1, LIH: 00000000
 TIME_VALUE             type 1 length 8: 30000
 LABEL_REQUEST          type 1 length 8: 00000800
 SESSION_ATTRIBUTE      type 7 length 12:
  setup_pri: 7, hold_pri: 0 flags: 00000001
  session name: test
 SENDER_TSPEC           type 2 length 36:
  max rate: 0, mean rate: 0, max burst: 0
 SENDER_TEMPLATE        type 7 length 12:
  source 20.1.1.1, tunnel_id 1

Receive RSVP PathTear: src 20.1.1.1, dst 3.3.3.3 on 3/1
  version: 1, flags: 00000010, cksum: 00006685, ttl: 62, length: 48
 SESSION                type 7 length 16:
  destination 3.3.3.3, tunnelid 1, ext tunnelid (source) 20.1.1.1
 RSVP_HOP               type 1 length 12:
  address: 11.1.1.1, LIH: 00000000
 SENDER_TEMPLATE        type 7 length 12:
  source 20.1.1.1, tunnel_id 1
```

Syntax: ptrace mpls rsvp detail-of-received

Possible values: N/A

Default value: N/A

ptrace mpls rsvp extensive

Displays messages about internal RSVP functions.

EXAMPLE:

```
BigIron# ptrace mpls rsvp extensive
```


After you enter this command, messages such as the following appear at the destination specified for debugging output. You can turn off these messages by entering the **ptrace mpls rsvp extensive** command again.

```
DC-RSVP instance 1 is becoming the fault tolerant primary instance.
DC-RSVP instance 1 is configured, and is activating.
DC-RSVP registered with an Interface Partner: DC-RSVP instance = 1
DC-RSVP registered with an Interface Partner: DC-RSVP instance = 1
DC-RSVP registered with an Interface Partner: DC-RSVP instance = 1

LSP up: sess_tnnl_ID = 0X0001, sess_exten_ID = 0X0A010101, sess_dest_addr =
0X03030303

DC-RSVP has received a Path Tear: sess_tnnl_ID = 0X0001, sess_exten_ID =
0X0A010101, lsp_ID = 0X0001, lsp_src_addr = 0X0A010101, sess_dest_addr =
0X03030303

LSP brought dn: release_flags = 0X004C, sess_tnnl_ID = 0X0001, sess_exten_ID =
0X0A010101, sess_dest_addr = 0X03030303

DC-RSVP is building a Resv Err: err_code = 3, err_val = 0, sess_tnnl_ID = 0X0001,
sess_exten_ID = 0X14010101, lsp_ID = 0X0101, lsp_src_addr = 0X00000001,
sess_dest_addr = 0X03030303

DC-RSVP is setting error code 0X00000003 and value 0X00000000 for session, address
0X03030303.

DC-RSVP failed to locate session matching SESSION object, address 0X03030303.
```

Syntax: ptrace mpls rsvp extensive

Possible values: N/A

Default value: N/A

ptrace none

Disables all packet tracing.

EXAMPLE:

```
BigIron# ptrace none
```

Syntax: ptrace ip

Possible values: N/A

Default value: N/A

ptrace ospf

Toggles tracing for OSPF packets.

EXAMPLE:

```
BigIron# ptrace ospf
```

Syntax: ptrace ospf

Possible values: N/A

Default value: N/A

ptrace pim fcache

Toggles tracing for PIM fcache packets.

EXAMPLE:

```
BigIron# ptrace pim fcache
```


Syntax: ptrace pim fcache

Possible values: N/A

Default value: N/A

ptrace pim mcache

Toggles tracing for PIM mcache packets.

EXAMPLE:

```
BigIron# ptrace pim mcache
```

Syntax: ptrace pim mcache

Possible values: N/A

Default value: N/A

ptrace pim message

Toggles tracing for PIM message packets.

EXAMPLE:

```
BigIron# ptrace pim message
```

Syntax: ptrace pim message

Possible values: N/A

Default value: N/A

ptrace pim none

Disables tracing for PIM packets.

EXAMPLE:

```
BigIron# ptrace pim none
```

Syntax: ptrace pim none

Possible values: N/A

Default value: N/A

ptrace ppp

Toggles tracing for PPP packets.

EXAMPLE:

```
BigIron# ptrace ppp
```

Syntax: ptrace ppp

Possible values: N/A

Default value: N/A

ptrace rarp

Toggles tracing for RARP packets.

EXAMPLE:

```
BigIron# ptrace rarp
```

Syntax: ptrace rarp

Possible values: N/A

Default value: N/A

ptrace rip

Toggles tracing for RIP packets.

EXAMPLE:

```
BigIron# ptrace rip
```

Syntax: ptrace rip

Possible values: N/A

Default value: N/A

ptrace snmp

Toggles tracing for SNMP packets.

EXAMPLE:

```
BigIron# ptrace snmp
```

Syntax: ptrace snmp

Possible values: N/A

Default value: N/A

ptrace switch none

Disables packet tracing started with the **ptrace switch stp** command.

EXAMPLE:

```
BigIron# ptrace switch none
```

Syntax: ptrace switch none

Possible values: N/A

Default value: N/A

ptrace switch stp

Toggles tracing for STP packets.

EXAMPLE:

```
BigIron# ptrace switch stp
```

Syntax: ptrace switch stp

Possible values: N/A

Default value: N/A

ptrace tcp

Toggles tracing for TCP packets.

EXAMPLE:

```
BigIron# ptrace tcp
```

Syntax: ptrace tcp

Possible values: N/A

Default value: N/A

ptrace telnet

Toggles tracing for Telnet packets.

EXAMPLE:

```
BigIron# ptrace telnet
```


Syntax: ptrace telnet

Possible values: N/A

Default value: N/A

ptrace term

Sends packet tracing output to the current terminal.

EXAMPLE:

```
BigIron# ptrace term
debug output is now sent to this terminal
```

Syntax: ptrace term

Possible values: N/A

Default value: Packet tracing output is sent to the console by default.

ptrace tftp

Toggles tracing for TFTP packets.

EXAMPLE:

```
BigIron# ptrace tftp
```

Syntax: ptrace tftp

Possible values: N/A

Default value: N/A

ptrace udp

Toggles tracing for UDP packets.

EXAMPLE:

```
BigIron# ptrace udp
```

Syntax: ptrace udp

Possible values: N/A

Default value: N/A

show ip bgp debug

Displays BGP debugging information for the router.

EXAMPLE:

```
BigIron# show ip bgp debug
  BGP4 Debug Information
Pid SBlock TBlocks UBlocks FBlocks EBlocks SAddress CAddress
0   16     10000   26      9973    0       04e6c16a 04e6c372
1   32     10000   9240    758     0       04e9cec2 04ebd0be
2   64     10000   41      9958    0       04ef4d1a 04ef504a
3   150    200     2       197     0       04f9ad72 04f9ae0c
4   22     67000   64404   2596    0       04fa25da 05030d1e
5   30     144000  131768  12228   0       0514baa2 0537b84e
6   74     67000   65886   1113    0       055f6fba 059d3c52
7   72     10000   9309    689     0       05af2de2 05b90822

  Total Memory Use for Route and Attributes Tables   : 13894800
  Memory Block Not Available Count   : 0
  Maximum Number of Attribute Entries Supported : 10000
  Maximum Number of Routes Supported : 67000
  Maximum Number of Peers Supported  : 3
  BGP Route Table Full Count         : 0
  Bad Memory Pool ID Count           : 0
  Bad Memory Address Count           : 0
  debug ip bgp errors
  debug ip bgp event
  debug ip bgp state
```

ALTERNATE OUTPUT:

```
BigIron 8000#sh ip bgp debug
  BGP4 Debug Information
Pid SBlock TBlocks UBlocks FBlocks Failure p_alloc #_pools p_unit
0   8      0       0       0       0       0       0       100
1   16     0       0       0       0       0       0       100
2   24     0       0       0       0       0       0       100
3   32     0       0       0       0       0       0       40
4   48     0       0       0       0       0       0       20
5   64     0       0       0       0       0       0       10
6   96     0       0       0       0       0       0       10
7   128    0       0       0       0       0       0       10
8   256    0       0       0       0       0       0       10
9   22     0       0       0       0       0       0       200
10  36     0       0       0       0       0       0       400
11  80     0       0       0       0       0       0       200
12  73     0       0       0       0       0       0       200
  Total Memory Use for Route and Attributes Tables   : 0
  Memory Block Not Available Count   : 0
  Bad Memory Pool ID Count           : 0
  Maximum Peer Index Number          : 0
  Number Of Peers Configured         : 0
  Malloc count for route info        : 0
  TCP transmit buffers               : 128  0
  Schedule BGP route calculation     : 6
```

The following table describes the output from the **show ip bgp debug** command:

Table 3.6: Output from the show ip bgp debug command

Statistic	Description
Pid	Memory pool ID 0 – 7
SBlock	Size of the memory blocks in the memory pool.
TBlocks	Total number of blocks in the memory pool.
UBlocks	Number of used blocks in the memory pool.
FBlocks	Number of free blocks in the memory pool.
EBlocks	Number of error blocks
SAddress	Starting address of the memory pool.
CAddress	Ending address of the memory pool.
Total Memory Use for Route and Attributes Tables	Amount of memory available for the BGP4 route and attributes tables.
Memory Block Not Available Count	Number of times that a memory block was not available.
Maximum Number of Attribute Entries Supported	Number of attribute entries the router's memory can hold. An attribute entry is a set of route attributes that are associated with one or more routes.
Maximum Number of Routes Supported	Number of BGP4 routes the router's memory can hold.
Maximum Number of Peers Supported	Number of BGP4 peers the router can have.
BGP Route Table Full Count	How many times a route could not be added to the BGP route table because the route table was full.
Bad Memory Pool ID Count	Number of times a memory pool was reported as bad. If there is a non-zero value in this field, contact Foundry technical support.
Bad Memory Address Count	Number of times a memory address was reported as bad. If there is a non-zero value in this field, contact Foundry technical support.
debug ip bgp errors debug ip bgp event debug ip bgp state	The **debug ip bgp** options that are currently in effect.

Syntax: show ip bgp debug

Possible values: N/A

Default value: N/A

show debug

Lists the debugging options currently in effect on the device.

EXAMPLE:

```
BigIron# debug all
```

```
BigIron# show debug
Debug message destination:  Console
IP Routing:
          BGP:  bgp debugging is on
          BGP:  neighbor 0.0.0.0 debugging is on
          BGP:  dampening debugging is on
          BGP:  events debugging is on
          BGP:  inbound information debugging is on
          BGP:  keepalives debugging is on
          BGP:  outbound information debugging is on
          BGP:  updates debugging is on
         OSPF:  adjacency events debugging is on
         OSPF:  database timer debugging is on
         OSPF:  events debugging is on
         OSPF:  flooding debugging is on
         OSPF:  lsa generation debugging is on
         OSPF:  packet debugging is on
         OSPF:  retransmission debugging is on
         OSPF:  spf debugging is on
         OSPF:  tree debugging is on
          RIP:  rip debugging is on
          RIP:  database debugging is on
          RIP:  events debugging is on
          RIP:  trigger debugging is on
         VRRP:  events debugging is on
         VRRP:  packet debugging is on
IP Multicast:
        DVMRP:  dvmrp debugging is on
        DVMRP:  detail debugging is on
        DVMRP:  pruning debugging is on
          PIM:  pim debugging is on
          PIM:  events debugging is on
          PIM:  group 0.0.0.0 debugging is on
         VRRP:  events debugging is on
         VRRP:  packet debugging is on
         IGMP:  IGMP debugging is on
Generic IP:
          TCP:  driver debugging is on
          TCP:  intercept debugging is on
          TCP:  packet debugging is on
          TCP:  rcmd debugging is on
          TCP:  sack debugging is on
          TCP:  transactions debugging is on
          UDP:  debugging is on
         IGMP:  IGMP debugging is on
         ICMP:  events debugging is on
         ICMP:  packets debugging is on
```

Syntax: show debug

Possible values: N/A

Default value: N/A

Chapter 4
Using the Backplane Debugging Commands

For debugging purposes, you can monitor information about the backplane hardware on a Chassis device. When the backplane debugging feature is enabled, every 30 seconds the device checks the following counters: SMC DMA Drop counters (DMADrop), SMC Backplane Drop counters (BPDrop), BM Free Queue Depth counters (FreeDepth), and BM Write Sequence Drop counters (WriteDrop). The device generates a Syslog message when any of the following conditions are true:

- DMADrop count is non-zero
- BPDrop count is non-zero
- WriteDrop count is greater than or equal to 1,500 increments per 30 seconds
- If the queue depth indicated by the FreeDepth counters is 120 less than the management module's approximate maximum free queue depth for 3 consecutive measurements.
    - On Management V modules, the maximum free queue depth is approximately 4000.
    - On Management IV modules, the maximum free queue depth is approximately 3960.
    - On Management III modules, the maximum free queue depth is approximately 1970.
    - On Management I, Management II, and Switch modules, the maximum free queue depth is approximately 890.

Table 4.1 describes the Syslog messages that can appear when the backplane debugging feature is enabled.

Table 4.1: Syslog messages generated by the backplane debugging feature

Message Level	Message	Explanation
Alert	Slot <num> SMC <num> Drop counter is <num>	When the backplane debugging feature is enabled, the first time the SMC DMA Drop (DMADrop) counter is non-zero, the device generates a Syslog message and an SNMP trap. When the first Syslog message indicating a non-zero DMADrop count is generated, the device starts a five-minute timer. After five minutes, the device generates a Syslog message if the DMADrop count is non-zero at least once during this five-minute period. Slot <num> is the slot number that contains the module. SMC <num> indicates the Strip Memory Controller (SMC) ASIC. Drop counter is <num> indicates the total number of SMC DMA drops during the five-minute period.
Alert	Slot <num> BP <num> Drop counter is <num>	When the backplane debugging feature is enabled, the first time the SMC Backplane Drop (BPDrop) counter is non-zero, the device generates a Syslog message and an SNMP trap. When the first Syslog message indicating a non-zero BPDrop count is generated, the device starts a five-minute timer. After five minutes, the device generates a Syslog message if the BPDrop count is non-zero at least once during this five-minute period. Slot <num> is the slot number that contains the module. BP <num> is the current value of the BPDrop counter. Drop counter is <num> indicates the total number of SMC backplane drops during the five-minute period.

Table 4.1: Syslog messages generated by the backplane debugging feature

Message Level	Message	Explanation
Warning	Slot <num> <module> Free Queue decreases less than the desirable values 3 consecutive times.	The module's BM Free Queue Depth (FreeDepth) has been recorded at 120 less than the maximum for the module for three consecutive measurements. • On Management V modules, the maximum free queue depth is approximately 4000. • On Management IV modules, the maximum free queue depth is approximately 3960. • On Management III modules, the maximum free queue depth is approximately 1970. • On Management I, Management II, and Switch modules, the maximum free queue depth is approximately 890. Slot <num> <module> is the slot number that contains the module and the kind of module.
Informational	Slot <num> Write Sequence Drop <num> within 30 seconds	The BM Write Sequence Drop (WriteDrop) counter is greater or equal to 1,500 increments per 30 seconds. Slot <num> is the slot number that contains the module. Write Sequence Drop <num> is the current value of the WriteDrop counter.

To enable the backplane debugging feature, enter the following command:

```
BigIron# debug hw
```

Syntax: [no] debug hw

To disable the backplane debugging feature, enter one of the following commands:

```
BigIron# no debug hw
```

or

```
BigIron# undebug hw
```

Syntax: undebug hw

Entering the **no debug hw** or **undebug hw** commands stops the backplane debugging feature, but does not clear the WriteDrop counters (the other counters are cleared once they are read). To clear the WriteDrop counters, you can either reboot the device, or enter the following command:

```
BigIron# clear hw writedrop
```

Syntax: clear hw writedrop

To display the status of the backplane counters, enter the following command:

```
BigIron# show backplane
_____________________________________________________________
Slot  Mod     FreeQ    DMADrop    BPDrop    WriteDrop    Last
-------------------------------------------------------------
3     BxGMR4  3988     0          0         252          D:0  H:0 M:20S:5
4     B24E    900      0          0         0            NEVER
```

Syntax: show backplane

The **show backplane** command displays the status of the backplane counters since the last boot (for the WriteDrop counters, either the last boot or the last time the counters were cleared with the **clear hw writedrop** command). Table 4.2 describes the output from the **show backplane** command.

Table 4.2: Output from the show backplane command

This Field...	Displays...
Slot	The slot number for the module.
Mod	The module type.
FreeQ	The module's BM free queue depth counter.
DMADrop	The sum of the module's four SMC DMA drop counters.
BPDrop	The sum of the module's four SMC backplane drop counters.
WriteDrop	The module's BM write sequence drop counter.
Last	The last time an event was recorded. If any SMC DMA drops or SMC backplane drops have occurred, the time of the last drop is displayed. If there have been no SMC DMA drops or SMC backplane drops, the time of the BM write sequence drop is displayed. If there have been no drops at all, then NEVER is displayed.

Chapter 5
Changing CAM Partitions

You can adjust the percentage of a module's CAM that can store Layer 2, Layer 3, or Layer 4 entries. In releases prior to 07.6.01, CAM partitioning was not configurable. Starting in release 07.6.01, you can specify the percentage of CAM assigned to each of the CAM entry types, both on a global and per-module basis. After you reboot the Foundry device, the user-specified CAM partitions take effect.

This chapter is divided into the following sections:

CAM Overview

Content Addressable Memory (CAM) is a component of Foundry modules that facilitates hardware forwarding. As packets flow through the Foundry device from a given source to a given destination, the management processor records forwarding information about the flow in CAM entries. A CAM entry generally contains next-hop information, such as the outgoing port, the MAC address of the next-hop router, VLAN tag, and so on. Once the Foundry device has this information in its CAM, packets with the same source and destination can be forwarded by hardware without the aid of the management processor, speeding up forwarding time.

CAM entries can contain Layer 2, Layer 3, or Layer 4 information. Each type of CAM entry has its own format. Layer 2 CAM entries contain destination MAC information; Layer 3 CAM entries contain destination IP information; Layer 4 CAM entries contain destination IP, destination TCP/UDP port, source IP, and source TCP/UDP port information. Layer 2 entries also deal with 802.1p (priority), and VLAN information.

When the Foundry device is initialized, the software partitions the available CAM into segments for Layer 2, Layer 3, or Layer 4 information. The percentage of CAM devoted to each type of CAM entry depends on the software image running on the device. For example, switch software may assign all of the available CAM to Layer 2 entries, while router software may assign a percentage of CAM to Layer 3 and a percentage to Layer 2/4.

On BigIron routers, the CAM lookup mechanism involves longest prefix match with up to three levels of overlapping prefixes. The Layer 3 CAM partition on these devices is divided into three levels of "supernet" host routes, designated Level1, Level2, and Level3. For Layer 3 IP network routes, Level1 routes precede Level2 routes, and Level2 routes precede Level3 routes. For example, given three routes to program into the CAM, 110.23.24.0/24, 110.23.0.0/16 and 110.0.0.0/8, the device programs 110.23.24.0/24 in Level1, 110.23.0.0/16 in Level2, and 110.0.0.0/8 in Level3.

The Layer 4 CAM partition is divided into four pools, designated Pool0, Pool1, Pool2, and Pool3. Pools 1 – 3 store Layer 4 session CAM entries. When no match for an IP packet is found in Pools 1 – 3, an entry for the packet is made in Pool0. IP packets with CAM entries in Pool0 are sent to the CPU. By default, entries for all packet types

except TCP are programmed into Pool0. When strict ACL TCP mode is enabled (with the **ip strict-acl-tcp** command) TCP packets are also programmed into Pool0.

CAM partitioning also depends on the device type and module used: the BigIron and FastIron families each have different amounts of CAM available, and IronCore, JetCore, POS OC-48, and 10 Gigabit Ethernet modules use different CAM partitioning mechanisms. The following sections list the CAM entry size, amount of CAM, and default CAM partition size for each of these modules for switch and router software images.

CAM Partitioning on IronCore Modules

In the IronCore architecture, all CAM entries are 64-bits wide, regardless of type.

BigIron Gigabit modules have 1 Mbit of CAM for each set of four ports, for a total of 2 Mbits. B24E modules have 1 Mbit of CAM for all 24 ports. FastIron Gigabit modules have 128Kbits of CAM for every four ports (256 Kbits total), and FI24E modules have 128 Kbit CAM total.

For switch software images, by default the CAM is partitioned 100 percent for Layer 2 entries. For router software images, the default CAM partition is 50 percent Layer 2 entries and 50 percent Layer 3 entries. In unicast high-performance mode (the default for release 7.5.00 and above) the CAM partition is 75 percent Layer 3 entries and 25 percent Layer 2 entries. On IronCore modules, Layer 4 CAM entries are part of the Layer 2 partition.

CAM Partitioning on JetCore Modules

On JetCore modules, CAM entries can be 64 bits (for Layer 2 entries) 64 bits (for Layer 3 entries), or 128 bits (for Layer 4 entries). Each 64-bit Layer 3 CAM entry contains two 32-bit IP route entries.

JetCore module ports are managed by two kinds of custom ASICs:

- Integrated Gigabit Controllers (IGCs) – Ethernet packet controllers for Gigabit ports. Each Gigabit Ethernet module contains two IGCs.
- Integrated Packet Controllers (IPCs) – Ethernet packet controllers for 10/100 ports. Each 10/100 Ethernet module contains two IPCs.

Each IGC or IPC has its own CAM space. An IPC or IGC has 1Mbit of CAM for FastIron modules or 2 Mbits for BigIron modules. A J-BxG module has 4 Mbits of CAM, a J-FI48E module has 2 Mbits, and a J-B16GC module has 8Mbits.

For switch software images, the default CAM partition is 75 percent Layer 2 entries and 25 percent Layer 4 entries. For router software images, the default CAM partition is 50 percent Layer 3 entries, 25 percent Layer 2 entries, and 25 percent Layer 4 entries. Note that these percentages refer to the amount of CAM space allotted to each type of CAM entry, not to the actual number of CAM entries, since on JetCore modules CAM entries of different types can be different sizes.

CAM Partitioning on POS OC-48 Modules

The CAM on POS OC-48 modules is divided into Layer 2, Layer 3 Net, Layer 3 Host, Layer 4 In, Layer 4 Out, MPLS In, and MPLS Out partitions. Layer 2, MPLS In, and MPLS Out CAM entries are 64 bits wide. Layer 4 In and Layer 4 Out CAM entries are 128 bits wide. Each Layer 3 CAM entry is 64 bits wide and contains two 32-bit IP route entries.

POS OC-48 modules have either 4 Mbits or 8 Mbits of CAM space. The default CAM partition is 12.5 percent Layer 2 entries, 12.5 percent Layer 3 entries, 6.25 percent Layer 4 In entries, 6.25 percent Layer 4 Out entries, 37.5 percent MPLS In entries, and 25 percent MPLS Out entries.

User-configurable CAM partitions are not supported on POS OC-48 modules.

CAM Partitioning on 10 Gigabit Ethernet Modules

As with other JetCore modules, CAM entries on 10 Gigabit Ethernet modules are 64 bits (for Layer 2 entries) 64 bits (for Layer 3 entries), or 128 bits (for Layer 4 entries). Unlike the other JetCore modules, 10 Gigabit Ethernet modules have two CAM banks of 4 Mbits each. One CAM bank is used for Layer 2 destination address entries and Layer 3 entries, and the other CAM bank is used for Layer 2 source address entries and Layer 4 entries.

The amount of CAM space allotted to Layer 2 source address entries must be equal to the amount allotted to Layer 2 destination address entries. Consequently, if you increase the amount of Layer 2 CAM space, it will reduce the amount of CAM space for both Layer 3 and Layer 4 entries.

For switch software images, one bank of CAM is divided into 50 percent Layer 2 destination address entries and 50 percent Layer 3 entries (even though the Layer 3 entries are not used by the switch software). The other CAM bank is divided into 50 percent Layer 2 source address entries and 50 percent Layer 4 entries. For router software images, one bank of CAM is divided into 25 percent Layer 2 destination address entries and 75 percent Layer 3 entries. The other CAM bank is divided into 25 percent Layer 2 source address entries and 75 percent Layer 4 entries.

Using the CLI to Configure CAM Partitioning

You can configure CAM partitioning on a global or per-module basis. On a router image, you can specify percentages for Layer 2, Layer 3, and Layer 4 CAM entries. On a switch image, you can specify percentages for Layer 2 and Layer 4 CAM entries.

For example, the following command specifies CAM percentages to be applied to all the modules on a Foundry device running a router image.

```
BigIron(config)# cam-partition l2 0 l3 100 l4 0
Slot 1 (DMA 0) CAM Partition:
    IronCore Module, Total Size 1Mbits
    L2 232.530029Mbits 88789.002929%, L3 0.75Mbits 75%, L4 232.655029Mbits 88801
.502929%
    L3 = 12288 (level2 = 2048, level3 = 2048), Pool0 = 2048, Pool1 = 2048, Pool2
 = 544488408, Pool3 = 0
Slot 1 (DMA 2) CAM Partition:
    IronCore Module, Total Size 1Mbits
    L2 232.530029Mbits 88789.002929%, L3 0.75Mbits 75%, L4 232.655029Mbits 88801
.502929%
    L3 = 12288 (level2 = 2048, level3 = 2048), Pool0 = 2048, Pool1 = 2048, Pool2
 = 544488408, Pool3 = 0
Cold start required.  Please write memory and then reload or power cycle.
```

Syntax: cam-partition l2 <percent> l3 <percent> l4 <percent>

When you enter the **cam-partition** command, the Foundry device attempts to partition the available CAM into the percentages you specify. Due to internal hardware restrictions, the resulting CAM partitions may not exactly match the percentages you specify. The device attempts to come as close as possible to match the user-specified partitions. The new CAM partitioning takes effect after you enter the **write memory** command and restart the Foundry device.

The percentages you specify must add up to 100 percent. When you are globally setting CAM partitions on 10 Gigabit Ethernet Modules, the percentage assigned to Layer 3 must equal the percentage assigned to Layer 4.

The following command specifies CAM percentages to be applied to all the modules on a Foundry device running a switch image.

```
FastIron(config)# cam-partition l2 60 l4 40
Slot 1 (DMA 0) CAM Partition:
    IronCore Module, Total Size 1Mbits (16384 HW entries)
    L3 0Bits 0%, L2+L4 1Mbits 100%
   L3 = 0 (level2 = 0, level3 = 0), Pool0 = 2048, Pool1 = 2048, Pool2 = 2048, Pool3
= 10240
Note: For IronCore, L2 && L4 share pool1, pool2 and pool3.
Slot 1 (DMA 2) CAM Partition:
    IronCore Module, Total Size 1Mbits (16384 HW entries)
    L3 0Bits 0%, L2+L4 1Mbits 100%
   L3 = 0 (level2 = 0, level3 = 0), Pool0 = 2048, Pool1 = 2048, Pool2 = 2048, Pool3
= 10240
Note: For IronCore, L2 && L4 share pool1, pool2 and pool3.
Reload required.  Please write memory and then reload or power cycle.
```

Syntax: cam-partition l2 <percent> l4 <percent>

To specify CAM partitions on an individual module, enter commands such as the following:

```
BigIron(config)# hw-module 3
BigIron(config-module-3/8)# cam-part l2 10 l3 70 l4 20
Slot 3 (DMA 8) CAM Partition:
    IronCore Module, Total Size 1Mbits
    L2 232.530029Mbits 88789.002929%, L3 0.75Mbits 75%, L4 232.655029Mbits 88801
.502929%
    L3 = 12288 (level2 = 2048, level3 = 2048), Pool0 = 2048, Pool1 = 2048, Pool2
 = 544488408, Pool3 = 0
Cold start required.  Please write memory and then reload or power cycle.
```

Syntax: hw-module <module>

Displaying CAM Partitioning Information

CAM is shared among multiple DMAs on a Foundry module. The CAM is accessible by one of the DMAs, called a master DMA. The **show version** command displays which DMAs are master DMAs. For example:

```
BigIron# show version
  SW: Version 07.6.01b139T51 Copyright (c) 1996-2002 Foundry Networks, Inc.
      Compiled on Sep 18 2002 at 03:16:53 labeled as B2S07601b139
      (2413622 bytes) from Secondary B2S07601b139.bin
  HW: BigIron 8000 Router, SYSIF version 21
==========================================================================
SL 1: B8GMR Fiber Management Module, SYSIF 2, M2, ACTIVE
      Serial #: Non-exist.
 2048 KB BRAM, SMC version 1, ICBM version 21
  512 KB PRAM(512K+0K) and 2048*8 CAM entries for DMA  0, version 0209
  512 KB PRAM(512K+0K) and shared CAM entries for DMA  1, version 0209
  512 KB PRAM(512K+0K) and 2048*8 CAM entries for DMA  2, version 0209
  512 KB PRAM(512K+0K) and shared CAM entries for DMA  3, version 0209
==========================================================================
SL 3: B24E Copper Switch Module
      Serial #: Non-exist.
 2048 KB BRAM, SMC version 2, ICBM version 21
  256 KB PRAM(256K+0K) and 2048*8 CAM entries for DMA  8, version 0808
  256 KB PRAM(256K+0K) and shared CAM entries for DMA  9, version 0808
  256 KB PRAM(256K+0K) and shared CAM entries for DMA 10, version 0808
==========================================================================
Active management module:
  240 MHz Power PC processor 603 (version 7/1201) 63 MHz bus
  512 KB boot flash memory
 8192 KB code flash memory
  256 KB SRAM
  128 MB DRAM
The system uptime is 2 hours 11 minutes 56 seconds
The system : started=cold start
```

Syntax: show version

In the previous example, on the module in slot 1, DMAs 0 and 2 are master DMAs, and on the module in slot 3, DMA 8 is a master DMA. You can display CAM partitioning information for each master DMA. For example:

```
SW-BI8000-doc# show cam-partition brief

==== SLOT 1 CAM PARTITION ====

DMA: 0 (0x00)
Number of CAM devices per DMA:  8
Number of hw entries per CAM:  0x00800
Total size of CAM = 1Mbits
complete CAM index range per DMA:
  (sw) 1 - 16383  (1 - 0x03fff), total entries: 16383 (0x03fff)
  (hw) 0 - 16383  (0 - 0x03fff), total entries: 16384 (0x04000)
Percentage of CAM hardware entries for each partition:
  Level3 l3 = 2047 (0.124938Mbits)      (12.493896%)
  Level3 l3 = 2048 (0.125Mbits) (12.5%)
  Level3 l3 = 8192 (0.5Mbits)   (50%)
  Level4    = 4096 (0.25Mbits)  (25%)

DMA: 2 (0x02)
Number of CAM devices per DMA:  8
Number of hw entries per CAM:  0x00800
Total size of CAM = 1Mbits
complete CAM index range per DMA:
  (sw) 1 - 16383  (1 - 0x03fff), total entries: 16383 (0x03fff)
  (hw) 0 - 16383  (0 - 0x03fff), total entries: 16384 (0x04000)
Percentage of CAM hardware entries for each partition:
  Level3 l3 = 2047 (0.124938Mbits)      (12.493896%)
  Level3 l3 = 2048 (0.125Mbits) (12.5%)
  Level3 l3 = 8192 (0.5Mbits)   (50%)
  Level4    = 4096 (0.25Mbits)  (25%)
```

Syntax: show cam-partition brief

To display the index range for each kind of CAM entry, enter the following command:

```
BigIron# show cam-partition detail

==== SLOT 1 CAM PARTITION ====

DMA: 0 (0x00)
Number of CAM devices per DMA:  8
Number of hw entries per CAM:  0x00800
Total size of CAM = 1Mbits
complete CAM index range per DMA:
  (sw) 1 - 16383  (1 - 0x03fff), total entries: 16383 (0x03fff)
  (hw) 0 - 16383  (0 - 0x03fff), total entries: 16384 (0x04000)
Percentage of CAM hardware entries for each partition:
  Level3 l3 = 2047 (0.124938Mbits)      (12.493896%)
  Level3 l3 = 2048 (0.125Mbits) (12.5%)
  Level3 l3 = 8192 (0.5Mbits)   (50%)
  Level4    = 4096 (0.25Mbits)  (25%)

L3 level 3 index range:
  (sw) 1 - 2047         (0x00001 - 0x007ff), free 2047 (0x007ff)
  (hw) 1 - 2047         (0x00001 - 0x007ff)
L3 level 2 index range:
  (sw) 2048 - 4095      (0x00800 - 0x00fff), free 2048 (0x00800)
  (hw) 2048 - 4095      (0x00800 - 0x00fff)
L3 index range:
  (sw) 4096 - 12287     (0x01000 - 0x02fff), free 8189 (0x01ffd)
  (hw) 4096 - 12287     (0x01000 - 0x02fff)
L4 pool 0 index range:
  (sw) 12288 - 14335    (0x03000 - 0x037ff), free 2044 (0x007fc)
  (hw) 12288 - 14335    (0x03000 - 0x037ff)
L2/L4 pool 1 index range:
  (sw) 14336 - 16383    (0x03800 - 0x03fff), free 2047 (0x007ff)
  (hw) 14336 - 16383    (0x03800 - 0x03fff)
```

Syntax: show cam-partition detail

To display CAM partitioning information for a specified module, enter a command such as the following:

```
BigIron# show cam-partition module 3 brief

==== SLOT 3 CAM PARTITION ====

DMA: 8 (0x08)
Number of CAM devices per DMA:  8
Number of hw entries per CAM:  0x00800
Total size of CAM = 0.9375Mbits
complete CAM index range per DMA:
  (sw) 1 - 15359  (1 - 0x03bff), total entries: 15359 (0x03bff)
  (hw) 0 - 15359  (0 - 0x03bff), total entries: 15360 (0x03c00)
Percentage of CAM hardware entries for each partition:
  Level3 l3 = 2047 (0.124938Mbits)      (13.326822%)
  Level3 l3 = 2048 (0.125Mbits) (13.333333%)
  Level3 l3 = 8192 (0.5Mbits)   (53.333333%)
  Level4    = 3072 (0.1875Mbits)        (20%)
```

Syntax: show cam-partition module <module> brief | detail

Chapter 6
Using the ServerIron Packet Capture Utility

The ServerIron 400 and ServerIron 800 feature a utility that captures packets directed to the ServerIron's CPU, based on user-defined filters. The captured packets are stored in a capture buffer and can be displayed on-screen or transferred to a TFTP server.

Using the Packet Capture Utility

To use the packet capture utility, you configure the capture buffer, specify filters, apply filters, then start and stop the utility.

Configuring the Capture Buffer

Prior to starting the packet capture utility, you enter the debug filter level of the CLI and set the size of the capture buffer and the number of bytes from each captured packet to be stored.

To enter the debug filter level of the CLI:

```
ServerIron> enable
ServerIron# debug filter
ServerIron(debug-filter-all-all)#
```

Syntax: debug filter

NOTE: The `all-all` in the debug-filter prompt refers to the WSM CPU currently selected for viewing captured packet information. By default, the debug-filter prompt refers to all WSM modules and all WSM CPUs. To view captured packet information, you must select a specific WSM and specific WSM CPU, or select the Management Processor. See "Viewing Captured Packets" on page 6-6 for information on how to do this.

Setting the Capture Buffer Size

By default, the size of the capture buffer is set to zero. To use the packet capture utility, you can set the capture buffer to between 1 – 1024 Kilobytes.

For example, to set the capture buffer to 128 Kilobytes:

```
ServerIron(debug-filter-all-all)# buffer-size 128
```

Syntax: buffer-size <kbytes>

To display the capture buffer size:

```
ServerIron(debug-filter-all-all)# show buffer-size
Capture buffer size: 131072 bytes
```


Syntax: show buffer-size

Setting the Number of Bytes to be Stored in the Capture Buffer

By default 64 bytes from each captured packet are stored in the capture buffer. You can specify the number of bytes from a captured packet that can be stored, or specify that the entire packet be stored. For example, to specify that 128 bytes of a captured packet be stored in the capture buffer:

```
ServerIron(debug-filter-all-all)# packet-size 128
```

Syntax: packet-size <bytes> | whole

The <bytes> parameter can be between 64 – 1518 bytes. The **whole** parameter specifies that the entire packet be stored in the capture buffer.

To show the packet-size setting:

```
ServerIron(debug-filter-all-all)# show packet-size
Max bytes stored from a filtered pkt: 128
```

Syntax: show packet-size

Specifying Packet Capture Filters

You specify the packets to store in the capture buffer by configuring one or more ***filter IDs***. A filter ID consists of a set of filters that specify the attributes of packets to be stored in the capture buffer. You can configure up to 16 filter IDs.

Within a filter ID, you can specify filters for Layer 1 – 4 information in a packet. In addition, you can set up filters to capture packets that contain a specified pattern within the packet.

By default, a filter ID is configured to match any packet. Within a filter ID, all the filters must match a received packet in order for the packet to be captured. The filters not explicitly configured have “don’t care” values, which are ignored during the matching process.

To enter the configuration level for filter ID 1:

```
ServerIron(debug-filter-all-all)# specify 1
ServerIron(debug-filter-spec-1)#
```

Syntax: specify <filter-id>

At the filter ID configuration level, you can specify individual filters to be included in the filter ID as well as display the current settings for the filter ID.

Ethernet Filters

To specify an Ethernet filter, enter one of the following CLI commands:

Table 6.1: Ethernet Filters

CLI command	Filter type
mac bcast	Ethernet broadcast packets
mac dest <mac-address>	Packets with the specified destination MAC address
mac mcast	Ethernet multicast packets
mac src <mac-address>	Packets with the specified source MAC address
mac type <type-in-hex>	Packets of the specified Layer 3 type

IP Filters

To specify an IP filter, enter one of the following CLI commands:

Table 1: IP Filters

CLI command	Filter type
ip bcast	IP broadcast packets
ip dest <ip-address>	Packets with the specified destination IP address
ip mcast	IP multicast packets
ip protocol <protocol-in-hex>	Packets with the specified Layer 4 protocol
ip src <ip-address>	Packets with the specified source IP address

TCP Filters

To specify a TCP filter, enter one of the following CLI commands:

Table 2: TCP Filters

CLI command	Filter type
tcp src <port-number>	Packets with the specified source TCP port
tcp dest <port-number>	Packets with the specified destination TCP port
tcp syn	TCP packets with the SYN flag on
tcp reset	TCP packets with the RST flag on
tcp fin	TCP packets with the FIN flag on
tcp ack	TCP packets with the ACK flag on
tcp push	TCP packets with the PSH flag on
tcp urgent	TCP packets with the URG flag on

UDP Filters

To specify a UDP filter, enter one of the following CLI commands:

Table 3: UDP Filters

CLI command	Filter type
udp src <port-number>	Packets with the specified source UDP port
udp dest <port-number>	Packets with the specified destination UDP port

HTTP Filters

To specify an HTTP filter, enter one of the following CLI commands:

Table 4: HTTP Filters

CLI command	Filter type
url <url-string>	Packets that contain the specified URL string.
cookie <cookie-string>	Packets that contain the specified cookie.

Specifying a Pattern Matching Filter

You can set up a filter to capture packets that contain a pattern of a specified length, starting from a specified offset from the beginning of the packet. For example:

```
ServerIron(debug-filter-spec-1)# pattern 24 2 1203
```

Syntax: pattern <offset> <length> <pattern-in-hex>

The <offset> is the number of bytes from the start of the packet.

The <length> is the length of the pattern in bytes. You can specify between 1 – 32 bytes.

The <pattern-in-hex> is the pattern to match. The length of the pattern must be equal to the number of bytes specified with the <length> parameter.

Displaying Current Filter Settings

To display the current filter settings for the filter ID:

```
ServerIron(debug-filter-spec-1)# show

Filter-ID: 1

                MAC filters:
                Src  MAC : ANY
                Dest MAC : ANY
                MAC Type : ANY
        IP filters:
                Src  IP  : ANY
                Dest IP  : ANY
                Protocol : ANY
        TCP filters:
                Src  port: ANY
                Dest port: ANY
                Flags    : None
        UDP filters:
                Src  port: ANY
                Dest port: ANY
        HTTP filters:
                Url      : ANY
                Cookie   : ANY
        Pattern filters:
                Pattern  : ANY
```

Syntax: show

Additionally, you can display settings for a filter ID at the debug filter level. For example, to display settings for filter ID 1:

```
ServerIron(debug-filter-all-all)# show 1
```


Syntax: show <filter-id>

Disabling a Filter

To disable a filter within a filter ID, use the **no** form of the CLI command. For example:

```
ServerIron(debug-filter-spec-1)# no ip bcast
```

Syntax: no <filter-command>

Resetting a Filter ID to Default Values

To reset all the filters in a filter ID to their default values:

```
ServerIron(debug-filter-spec-1)# reset
```

Syntax: reset

Applying Packet Capture Filters

After you specify a filter ID, it takes effect when you apply it. A filter ID should be applied globally or on an individual port. You can apply a filter ID so that filters inbound traffic only, outbound traffic only, or both.

For example, to apply filter ID 1 globally for inbound and outbound traffic on all ports:

```
ServerIron(debug-filter-all-all)# apply 1
```

Syntax: apply <filter-id>

To apply filter ID 1 so it filters inbound traffic on port 3/11:

```
ServerIron(debug-filter-all-all)# apply 1 3/11 in
```

Syntax: apply <filter-id> <port-number> [in | out]

You can apply multiple filter IDs and specify an and/or relationship between them. For example, to apply filter IDs 1 and 2, enter the following command. Packets that match the filters in both filter IDs are stored in the capture buffer.

```
ServerIron(debug-filter-all-all)# apply "1 and 2"
```

Syntax: apply <expression> [<port-number> [in | out]]

You can use an OR expression to specify multiple filter IDs. Packets that match the filters in either filter ID are stored in the capture buffer. To apply filter IDs 1 or 2, enter the following command.

```
ServerIron(debug-filter-all-all)# apply "1 or 2"
```

To apply filter IDs 1, 2, and 3 so that packets must match the filters in 1 and match the filters in either 3 or 4, enter the following command:

```
ServerIron(debug-filter-all-all)# apply "(1 and (3 or 4))"
```

To view the currently applied expressions:

```
ServerIron(debug-filter-all-all)# show apply
Filter ID apply expression: ( 1 and ( 3 or 4 ) )
```

Syntax: show apply

Starting and Stopping the Packet Capture Utility

After specifying and applying one or more filter IDs, you can start the packet capture utility. Once you start the packet capture utility, filtered packets are stored in the capture buffer and are available for viewing until you restart the utility.

To start the packet capture utility, enter the following command:

```
ServerIron(debug-filter-all-all)# start
```

Syntax: start

The packet capture utility runs until the capture buffer is full or until it is manually stopped. You can stop the packet capture utility with the following command:

```
ServerIron(debug-filter-all-all)# stop
Number of packets captured: 0
```

Syntax: stop

The **start** and **stop** commands start or stop a new capture session on all WSM modules. A capture session cannot be started or stopped on an on individual WSM module. Similarly, when the capture buffer is full on a WSM module, the capture session ends on that WSM module while capture sessions on other WSM modules continue.

Configuring Event-Based Filters

You can trigger a capture session to automatically start and stop at a specified time. If a stop time is not specified, the capture session ends when the buffer is full.

Note that to use event-based filters, the ServerIron's clock must be set, either manually or by synchronizing with an SNTP server.

For example, to specify that a capture session start at 1:10:11 a.m., enter the following commands:

```
ServerIron(debug-filter-all-all)# event start
ServerIron(debug-filter-event-start)# time 01:10:11
```

Syntax: event start

Syntax: time <hh:mm:ss>

To specify that the capture session stop at 2:10:11 a.m., enter the following commands:

```
ServerIron(debug-filter-all-all)# event stop
ServerIron(debug-filter-event-stop)# time 02:10:11
```

Syntax: event stop

To display the time settings for start or stop events:

```
ServerIron(debug-filter-event-stop)# show

Event triggers:
        Stop triggers:
                Time : 02:10:11
```

Syntax: show

To view event settings at the debug filter level:

```
ServerIron(debug-filter-all-all)# show events

Event triggers:
        Start triggers:
                Time : 01:10:11
        Stop triggers:
                Time : 02:10:11
```

Syntax: show events

Viewing Captured Packets

You can view the packets in the capture buffer in ASCII format or hex format, on a packet ID basis. The ASCII format is a decoded version of the packet. Additionally, you can display a summary of all packets captured, with a one-line description of each packet.

You view packets based on the WSM CPU that handled them. To determine the WSM CPU that handles traffic for a specific module, use the **show wsm-map** command. For example:

```
ServerIron# show wsm-map
slot 3 (weight 24 x 100M) is processed by WSM 2/2 (weight 24)
slot 4 (weight 80 x 100M) is processed by WSM 2/1 (weight 80)
```

Syntax: show wsm-map

In this example, traffic for the module in slot 4 is handled by the WSM in slot 2, using CPU 1. You can select this WSM and WSM CPU by entering the following command:

```
ServerIron(debug-filter-all-all)# view bp 2 1
ServerIron(debug-filter-2-1)#
```

After you enter this command, packet information will be from the capture buffer for the WSM module in slot 2, WSM CPU 1.

To select the Management Processor capture buffer to view, enter the following command:

```
ServerIron(debug-filter-all-all)# view mp
ServerIron(debug-filter-mp)#
```

Syntax: view [mp | bp <slot-number> <cpu-number>]

To view a captured packet in ASCII format:

```
ServerIron(debug-filter-2-1)# ascii-dump 1
```

Syntax: ascii-dump <packet-number>

To view a captured packet in hex format:

```
ServerIron(debug-filter-2-1)# hex-dump 1
```

Syntax: hex-dump <packet-number>

To view a summary of all captured packets:

```
ServerIron(debug-filter-2-1)# summary

  1> 80   TCP :80   ->4628  Seq:63438631   Ack:12382816   ACK
  2> 80   TCP :80   ->4628  Seq:63440091   Ack:12382816   ACK
  3> 76   TCP :80   ->4629  Seq:161663377  Ack:12382728   SYN ACK
  4> 80   TCP :80   ->4629  Seq:161663378  Ack:12382792   ACK
  5> 80   TCP :80   ->4629  Seq:161664838  Ack:12382792   ACK
  6> 80   TCP :80   ->4629  Seq:161666298  Ack:12382792   ACK PSH
  7> 80   TCP :80   ->4629  Seq:161667758  Ack:12382792   ACK
  8> 80   TCP :80   ->4629  Seq:161669218  Ack:12382792   ACK PSH FIN
  9> 76   TCP :80   ->4630  Seq:63438637   Ack:12382737   SYN ACK
 10> 76   TCP :3140 ->80    Seq:12252912   Ack:12252912   RST
```

Syntax: summary

To view a summary for a range of packets in the capture buffer:

```
ServerIron(debug-filter-2-1)# summary 1 5

  1> 80   TCP :80   ->4628  Seq:63438631   Ack:12382816   ACK
  2> 80   TCP :80   ->4628  Seq:63440091   Ack:12382816   ACK
  3> 76   TCP :80   ->4629  Seq:161663377  Ack:12382728   SYN ACK
  4> 80   TCP :80   ->4629  Seq:161663378  Ack:12382792   ACK
  5> 80   TCP :80   ->4629  Seq:161664838  Ack:12382792   ACK
```

Syntax: summary <starting-packet-number> <ending-packet-number>

You can display headers up to and including a specified network layer. For example, to limit the display to Layer 2 and Layer 3 headers, enter the following command:

```
ServerIron(debug-filter-2-1)# summary headers 3
```

Syntax: summary headers 2 | 3 | 4 | all | default

The **2** parameter includes Layer 2 specific headers in the display.

The **3** parameter includes Layer 2 and Layer 3 specific headers in the display.

The **4** parameter includes Layer 2, Layer 3, and Layer 4 specific headers in the display.

The **all** parameter includes all packet headers in the display.

The **default** parameter includes headers from the highest decodable layer in the display.

To display the selected summary headers:

```
ServerIron(debug-filter-2-1)# show headers
Summary headers selected: Layer-3
```

Syntax: show headers

Using TFTP to Transfer Information from the Capture Buffer

You can use TFTP to transfer the contents of the capture buffer to a TFTP server. For example, to send an ASCII dump of all captured packets to a TFTP server:

```
ServerIron(debug-filter-2-1)# tftp ascii-dump all 192.168.9.210 ascii-dump.txt
```

Syntax: tftp ascii-dump all | <packet-number> <ip-addr> <target-file-name>

The **all** parameter transfers the entire contents of the capture buffer.

The <packet-number> parameter specifies an individual packet to be transferred.

The <ip-addr> parameter is the IP address of the TFTP server.

The <target-file-name> specifies a filename on the TFTP server.

To send a hex dump of all captured packets to a TFTP server:

```
ServerIron(debug-filter-2-1)# tftp hex-dump all 192.168.9.210 hex-dump.txt
```

Syntax: tftp hex-dump all | <packet-number> <ip-addr> <target-file-name>

To send a summary of all captured packets to a TFTP server:

```
ServerIron(debug-filter-2-1)# tftp summary 192.168.9.210 summary-dump.txt
```

Syntax: tftp summary <ip-addr> <target-file-name>

Filter Examples

Below are two examples of how to configure and start the packet capture utility.

The following commands set up the capture buffer to store 32 Kbytes of data and capture entire packets. A single filter ID is configured that specifies filters for packets whose source IP address is 10.10.10.10 and whose destination IP address is 20.20.20.20. After the filter ID is specified, it is applied and a capture session is started.

```
ServerIron> enable
ServerIron# debug filter
ServerIron(debug-filter-all-all)# buffer-size 32
ServerIron(debug-filter-all-all)# packet-size whole

ServerIron(debug-filter-all-all)# specify 1
ServerIron(debug-filter-spec-1)# ip src 10.10.10.10
ServerIron(debug-filter-spec-1)# ip dest 20.20.20.20
ServerIron(debug-filter-spec-1)# exit

ServerIron(debug-filter-all-all)# apply 1
ServerIron(debug-filter-all-all)# start
```

The following commands configure the ServerIron to capture packets whose source IP address is 10.10.10.10 (any destination), as well as packets whose destination IP address is 20.20.20.20 (any source). Two filter IDs are specified and applied. Packets that match either of the filter IDs are stored in the capture buffer.

```
ServerIron> enable
ServerIron# debug filter
ServerIron(debug-filter-all-all)# buffer-size 32
ServerIron(debug-filter-all-all)# packet-size whole

ServerIron(debug-filter-all-all)# specify 1
ServerIron(debug-filter-spec-1)# ip src 10.10.10.10
ServerIron(debug-filter-spec-1)# exit

ServerIron(debug-filter-all-all)# specify 2
ServerIron(debug-filter-spec-2)# ip dest 20.20.20.20
ServerIron(debug-filter-spec-2)# exit

ServerIron(debug-filter-all-all)# apply "1 or 2"
ServerIron(debug-filter-all-all)# start
```

