

debug ip drp

Use the **debug ip drp** EXEC command to display Director Response Protocol (DRP) information. The **no** form of this command disables debugging output.

[no] debug ip drp

Usage Guidelines

The **debug ip drp** command is used to debug the director response agent used by the Distributed Director product. The Distributed Director can be used to dynamically respond to Domain Name System (DNS) queries with the IP address of the “best” host based on various criteria.

Sample Display

The following is sample output from the **debug ip drp** command. This example shows the packet origination, the IP address that information is routed to, and the route metrics that were returned.

```
Router# debug ip drp

DRP: received v1 packet from 172.31.232.8, via Ethernet0
DRP: RTQUERY for 172.31.58.94 returned internal=0, external=0
```

Table 51 describes the fields in the output.

Table 51 **Debug IP DRP Field Descriptions**

Field	Description
DRP: received v1 packet from 172.31.232.8, via Ethernet0	The router received a version 1 DRP packet from the IP address shown, via the interface shown.
DRP: RTQUERY for 172.31.58.94	The DRP packet contained two Route Query requests. The first request was for the distance to the IP address 171.69.113.50.
internal	If nonzero, the metric for the internal distance of the route that the router uses to send packets in the direction of the client. The internal distance is the distance within the router’s autonomous system.
external	If nonzero, the metric for the Border Gateway Protocol (BGP) or external distance used to send packets to the client. The external distance is the distance outside the router’s autonomous system.

debug ip dvmrp

Use the **debug ip dvmrp** EXEC command to display information on Distance Vector Multiprotocol Routing Protocol (DVMRP) packets received and transmitted. The **no** form of this command disables debugging output.

[no] debug ip dvmrp [detail [access-list] [in | out]]

Syntax Description

detail	(Optional) Enables a more detailed level of output and displays packet contents.
<i>access-list</i>	(Optional) Causes the debug ip dvmrp command to restrict output to one access list.
in	(Optional) Causes the debug ip dvmrp command to output packets received in DVMRP reports.
out	(Optional) Causes the debug ip dvmrp command to output packets transmitted in DVMRP reports.

Usage Guidelines

Use the **debug ip dvmrp detail** command with care. This command generates a great deal of output and can interrupt other activity on the router when it is invoked.

Sample Displays

The following is sample output from the **debug ip dvmrp** command:

```
Router# debug ip dvmrp

DVMRP: Received Report on Ethernet0 from 172.19.244.10
DVMRP: Received Report on Ethernet0 from 172.19.244.11
DVMRP: Building Report for Ethernet0 224.0.0.4
DVMRP: Send Report on Ethernet0 to 224.0.0.4
DVMRP: Sending IGMP Reports for known groups on Ethernet0
DVMRP: Received Report on Ethernet0 from 172.19.244.10
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Building Report for Tunnel0 224.0.0.4
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Radix tree walk suspension
DVMRP: Send Report on Tunnel0 to 192.168.199.254
```

The following lines show that the router received DVMRP routing information and placed it in the mroute table:

```
DVMRP: Received Report on Ethernet0 from 172.19.244.10
DVMRP: Received Report on Ethernet0 from 172.19.244.11
```

The following lines show that the router is creating a report to send to another DVMRP router:

```
DVMRP: Building Report for Ethernet0 224.0.0.4
DVMRP: Send Report on Ethernet0 to 224.0.0.4
```

Table 52 provides a list of internet multicast addresses supported for host IP implementations.

Table 52 Internet Multicast Addresses

Address	Description	RFC
224.0.0.0	Base address (Reserved)	RFC 1112
224.0.0.1	All systems on this subnet	RFC 1112
224.0.0.2	All routers on this subnet	
224.0.0.3	Unassigned	
224.0.0.4	DVMRP routers	RFC 1075
224.0.0.5	OSPF/IGP all routers	RFC 1583

The following lines show that a protocol update report has been sent to all known multicast groups. Hosts use IGMP reports to communicate with routers and to request to join a multicast group. In this case, the router is sending an IGMP report for every known group to the host, which is running mroute. The host then responds as though the router was a host on the LAN segment that wants to receive multicast packets for the group.

```
DVMRP: Sending IGMP Reports for known groups on Ethernet0
```

The following is sample output from the **debug ip dvmrp detail** command:

```
Router# debug ip dvmrp detail

DVMRP: Sending IGMP Reports for known groups on Ethernet0
DVMRP: Advertise group 224.2.224.2 on Ethernet0
DVMRP: Advertise group 224.2.193.34 on Ethernet0
DVMRP: Advertise group 224.2.231.6 on Ethernet0
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Origin 150.166.53.0/24, metric 13, distance 0
DVMRP: Origin 150.166.54.0/24, metric 13, distance 0
DVMRP: Origin 150.166.55.0/24, metric 13, distance 0
DVMRP: Origin 150.166.56.0/24, metric 13, distance 0
DVMRP: Origin 150.166.92.0/24, metric 12, distance 0
DVMRP: Origin 150.166.100.0/24, metric 12, distance 0
DVMRP: Origin 150.166.101.0/24, metric 12, distance 0
DVMRP: Origin 150.166.142.0/24, metric 8, distance 0
DVMRP: Origin 150.166.200.0/24, metric 12, distance 0
DVMRP: Origin 150.166.237.0/24, metric 12, distance 0
DVMRP: Origin 150.203.5.0/24, metric 8, distance 0
```

The following lines show that this group is available to the DVMRP router. The mrouterd process on the host will forward the source and multicast information for this group through the DVMRP cloud to other members.

```
DVMRP: Advertise group 224.2.224.2 on Ethernet0
```

The following lines show the DVMRP route information:

```
DVMRP: Origin 150.166.53.0/24, metric 13, distance 0  
DVMRP: Origin 150.166.54.0/24, metric 13, distance 0
```

The *metric* is the number of hops the route has covered, and the *distance* is the administrative distance.

debug ip eigrp

Use the **debug ip eigrp** EXEC command to display information on Enhanced Interior Gateway Routing Protocol (EIGRP) packets. The **no** form of this command disables debugging output.

[no] debug ip eigrp

Usage Guidelines

This command helps you analyze the packets that are sent and received on an interface. Because the **debug ip eigrp** command generates large amounts of output, only use it when traffic on the network is light.

Sample Display

The following is sample output from the **debug ip eigrp** command:

```
Router# debug ip eigrp

IP-EIGRP: Processing incoming UPDATE packet
IP-EIGRP: Ext 192.168.3.0 255.255.255.0 M 386560 - 256000 130560 SM 360960 - 256000
104960
IP-EIGRP: Ext 192.168.0.0 255.255.255.0 M 386560 - 256000 130560 SM 360960 - 256000
104960
IP-EIGRP: Ext 192.168.3.0 255.255.255.0 M 386560 - 256000 130560 SM 360960 - 256000
104960
IP-EIGRP: 172.24.43.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 172.24.43.0 255.255.255.0 metric 371200 - 256000 115200
IP-EIGRP: 192.135.246.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 192.135.246.0 255.255.255.0 metric 46310656 - 45714176 596480
IP-EIGRP: 172.24.40.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 172.24.40.0 255.255.255.0 metric 2272256 - 1657856 614400
IP-EIGRP: 192.135.245.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 192.135.245.0 255.255.255.0 metric 40622080 - 40000000 622080
IP-EIGRP: 192.135.244.0 255.255.255.0, - do advertise out Ethernet0/1
```

Table 53 describes significant fields in the debug messages.

Table 53 Debug IP EIGRP Field Descriptions

Field	Description
IP-EIGRP:	Indicates EIGRP packet information.
Ext	Indicates the following address is an external destination rather than an internal destination, which would be labeled as Int.
M	Shows the computed metric, which includes SM and the cost between this router and the neighbor. The first number is the composite metric. The next two numbers are the inverse bandwidth and the delay, respectively.
SM	Shows the metric as reported by the neighbor.

debug ip error

To display IP errors, use the **debug ip error** command in privileged EXEC mode. To disable debugging errors, use the **no** form of this command.

debug ip error *access-list-number* [**detail**] [**dump**]

no debug ip error

Syntax Description

<i>access-list-number</i>	(Optional) The IP access list number that you can specify. If the datagram is not permitted by that access list, the related debugging output (or IP error) is suppressed. Standard, extended, and expanded access lists are supported. The range of standard and extended access lists is from 1 to 199. The range of expanded access lists is from 1300 to 2699.
detail	(Optional) Displays detailed IP error debugging information.
dump	(Hidden) Displays IP error debugging information along with raw packet data in hexadecimal and ASCII forms. This keyword can be enabled with individual access lists and also with the detail keyword. The dump keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution notes below, in the usage guidelines, for more specific information.

Usage Guidelines

This command is used for IP error debugging. The output displays IP errors which are locally detected by this router.

Caution Enabling this command will generate output only if IP errors occur. However, if the router starts to receive many packets that contain errors, substantial output may be generated and severely affect system performance. This command should be used with caution in production networks. It should only be enabled when traffic on the IP network is low, so other activity on the system is not adversely affected. Enabling the **detail** and **dump** keywords use the highest level of system resources of the available configuration options for this command, so a high level of caution should be applied when enabling either of these keywords.

Caution The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. Because of the risk of using significant CPU utilization, the dump keyword is hidden from the user and cannot be seen using the “?” prompt. The length of the displayed packet information may exceed the actual packet length and include additional padding bytes that do not belong to the IP packet. Also note that the beginning of a packet may start at different locations in the dump output depending on the specific router, interface type, and packet header processing that may have occurred before the output is displayed.

Sample Output

The following is sample output from the **debug ip error** command:

```
debug ip error

IP packet errors debugging is on

04:04:45:IP:s=10.8.8.1 (Ethernet0/1), d=10.1.1.1, len 28, dispose ip.hopcount
```

The IP error in the above output was caused when the router attempted to forward a packet with a time-to-live (TTL) value of 0. The “ip.hopcount” traffic counter is incremented when a packet is dropped because of an error. This error is also displayed in the output of the **show ip traffic** command by the “bad hop count” traffic counter.

Table 54 Table 54 describes the significant fields shown in the display.

Table 54 debug ip error Field Descriptions

Field	Description
IP:s=10.8.8.1 (Ethernet0/1)	The packet source IP address and interface.
d=10.1.1.1, len 28	The packet destination IP address and prefix length.
dispose ip.hopcount	This traffic counter increments when an IP packet is dropped because of an error.

The following is sample output from the **debug ip error** command enabled with the **detail** keyword:

```
debug ip error detail

IP packet errors debugging is on (detailed)

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.1.1.1, len 28, dispose udp.noport
1d08h:   UDP src=41921, dst=33434

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.2.2.2, len 28, dispose ip.hopcount
1d08h:   UDP src=33691, dst=33434
```

The detailed output includes layer 4 information in addition to the standard output. The IP error in the above output was caused when the router received a UDP packet when no application was listening to the UDP port. The “udp.noport” traffic counter is incremented when the router drops a UDP packet because of this error. This error is also displayed in the output of the **show ip traffic** command by the “no port” traffic counter under “UDP statistics.”

Table 55 Table 55 describes the significant fields shown in the display.

Table 55 debug ip error detail Field Descriptions

Field	Description
IP:s=10.0.19.100 (Ethernet0/1)	The IP packet source IP address and interface.
d=10.1.1.1, len 28	The IP packet destination and prefix length.
dispose udp.noport	The traffic counter that is incremented when a UDP packet is dropped because of this error.

The following is sample output from the **debug ip error** command enabled with the **detail** and **dump** keywords:

```
debug ip error detail dump

IP packet errors debugging is on (detailed) (dump)

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.1.1.1, len 28, dispose udp.noport
1d08h:   UDP src=37936, dst=33434
03D72360:           0001 42AD4242           ..B-BB
03D72370:0002FCA5 DC390800 4500001C 30130000 ..|\9..E...0...
03D72380:01116159 0A001364 0A010101 9430829A ..aY...d....0..
03D72390:0008C0AD           ..@-

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.2.2.2, len 28, dispose ip.hopcount
1d08h:   UDP src=41352, dst=33434
03C01600:           0001 42AD4242           ..B-BB
03C01610:0002FCA5 DC390800 4500001C 302A0000 ..|\9..E...0*..
03C01620:01116040 0A001364 0A020202 A188829A ..`@...d....!...
03C01630:0008B253           ..2S
```

Note The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution in the usage guidelines section of this command reference page for more specific information.

The output from the **debug ip error** command, when the **dump** keyword is enabled, provides raw packet data in hexadecimal and ASCII forms. This additional output is displayed in addition to the standard output. The **dump** keyword can be used with all of the available configuration options of this command.

Table 55 Table 56 describes the standard output fields shown in the display.

Table 56 debug ip error detail dump Field Descriptions

Field	Description
IP:s=10.0.19.100 (Ethernet0/1)	The IP packet source IP address and interface.
d=10.1.1.1, len 28	The IP packet destination and prefix length.
dispose udp.noport	The traffic counter that is incremented when a UDP packet is dropped because of this error.

Related Commands

show ip traffic

debug ip ftp

To activate the debugging option to track the transactions submitted during an FTP session, use the **debug ip ftp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

```
[no] debug ip ftp
```

Usage Guidelines

The **debug ip ftp** command is useful for debugging problems associated with File Transfer Protocol (FTP).

Sample Display

The following is an example of the **debug ip ftp** command:

```
Router# debug ip ftp
FTP transactions debugging is on
```

The following is sample output from the **debug ip ftp** command:

```
FTP: 220 ProFTPD 1.2.0pre8 Server (DFW Nostrum FTP Server) [defiant.dfw.nostrum.com]
Dec 27 22:12:09.133: FTP: ---> USER router
Dec 27 22:12:09.133: FTP: 331 Password required for router.
Dec 27 22:12:09.137: FTP: ---> PASS WQHK5JY2
Dec 27 22:12:09.153: FTP: 230 Anonymous access granted, restrictions apply.
Dec 27 22:12:09.153: FTP: ---> TYPE I
Dec 27 22:12:09.157: FTP: 200 Type set to I.
Dec 27 22:12:09.157: FTP: ---> PASV
.....
.....
Dec 27 22:12:09.173: FTP: ---> QUIT
Dec 27 22:12:09.181: FTP: 221 Goodbye.
```

debug ip http authentication

Use the **debug ip http authentication** privileged EXEC command to troubleshoot HTTP authentication problems. This command displays the authentication method the router attempted and authentication-specific status messages. The **no** form of this command disables debugging output.

[no] debug ip http authentication

Sample Display

The following is sample output from the **debug ip http authentication** command:

```
Router# debug ip http authentication

Authentication for url '/' '/' level 15 privless '/'
Authentication username = 'local15' priv-level = 15 auth-type = local
```

Table 57 describes the output fields created by the **debug ip http authentication** command.

Table 57 Debug IP HTTP Authentication Descriptions

Field	Description
Authentication for url	Provides information about the URL in different forms.
Authentication username	Identifies the user.
priv-level	Indicates the user privilege level.
auth-type	Indicates the authentication method.

debug ip http ezsetup

Use the **debug ip http ezsetup** EXEC command to display the configuration changes that occur during the EZ Setup process. The **no** form of this command disables debugging output.

[no] debug ip http ezsetup

Usage Guidelines

Use the **debug ip http ezsetup** command to verify the EZ Setup actions without changing the router's configuration.

EZ Setup is a form you fill out to perform basic router configuration from most Hypertext Markup Language (HTML) browsers.

Sample Display

The following is sample output from the **debug ip http ezsetup** that shows the configuration changes for the router when the EZ Setup form has been submitted:

```
Router# debug ip http ezsetup

service timestamps debug
service timestamps log
service password-encryption
!
hostname router-name
!
enable secret router-pw
line vty 0 4
password router-pw
!
interface ethernet 0
 ip address 172.21.52.9 255.255.255.0
 no shutdown
 ip helper-address 172.31.2.132
 ip name-server 172.31.2.132
 isdn switch-type basic-5ess
 username Remote-name password Remote-chap
interface bri 0
 ip unnumbered ethernet 0
 encapsulation ppp
 no shutdown
 dialer map ip 192.168.254.254 speed 56 name Remote-name Remote-number
 isdn spid1 spid1
 isdn spid2 spid2
 ppp authentication chap callin
 dialer-group 1
!
ip classless
access-list 101 deny udp any any eq snmp
access-list 101 deny udp any any eq ntp
access-list 101 permit ip any any
dialer-list 1 list 101
ip route 0.0.0.0 0.0.0.0 192.168.254.254
ip route 192.168.254.254 255.255.255.255 bri 0
logging buffered
snmp-server community public RO
```

debug ip http ezsetup

```
ip http server
ip classless
ip subnet-zero
!
end
```

Related Commands

debug ip http token
debug ip http transaction
debug ip http url

debug ip http ssi

Use the **debug ip http ssi** EXEC command to display information about the HTML SSI EXEC command or HTML SSI ECHO command. The **no** form of this command disables debugging output.

[no] debug ip http ssi

Sample Display

The following is sample output from the **debug ip http ssi** command:

```
Router# debug ip http ssi

HTML: filtered command `exec cmd="show users"`
HTML: SSI command `exec`
HTML: SSI tag `cmd` = "show users"
HTML: Executing CLI `show users` in mode `exec` done
```

The following line shows the contents of the Server Side Include (SSI) EXEC command:

```
HTML: filtered command `exec cmd="show users"``
```

The following line indicates the type of SSI command that was requested:

```
HTML: SSI command `exec`
```

The following line shows the argument “*show users*” assigned to the tag ‘*cmd*’:

```
HTML: SSI tag `cmd` = "show users"
```

The following line indicates that the Cisco IOS **show users** command is being executed in EXEC mode:

```
HTML: Executing CLI `show users` in mode `exec` done
```

debug ip http token

Use the **debug ip http token** EXEC command to display individual tokens parsed by the Hypertext Transfer Protocol (HTTP) server. The **no** form of this command disables debugging output.

[no] debug ip http token

Usage Guidelines

Use the **debug ip http token** command to display low-level HTTP server parsings. To display high-level HTTP server parsings, use the **debug ip http transaction** command.

Sample Display

The following is part of a sample output from the **debug ip http token** command. In this example, the browser accessed the router's home page `http://router-name/`. The output gives the token parsed by the HTTP server and its length.

```
Router# debug ip http token

HTTP: token len 3: 'GET'
HTTP: token len 1: ' '
HTTP: token len 1: '/'
HTTP: token len 1: ' '
HTTP: token len 4: 'HTTP'
HTTP: token len 1: '/'
HTTP: token len 1: '1'
HTTP: token len 1: '.'
HTTP: token len 1: '0'
HTTP: token len 2: '\15\12'
HTTP: token len 7: 'Referer'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 4: 'http'
HTTP: token len 1: ':'
HTTP: token len 1: '/'
HTTP: token len 1: '/'
HTTP: token len 3: 'www'
HTTP: token len 1: '.'
HTTP: token len 3: 'thesite'
HTTP: token len 1: '.'
HTTP: token len 3: 'com'
HTTP: token len 1: '/'
HTTP: token len 2: '\15\12'
HTTP: token len 10: 'Connection'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 4: 'Keep'
HTTP: token len 1: '-'
HTTP: token len 5: 'Alive'
HTTP: token len 2: '\15\12'
HTTP: token len 4: 'User'
HTTP: token len 1: '-'
HTTP: token len 5: 'Agent'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 7: 'Mozilla'
HTTP: token len 1: '/'
HTTP: token len 1: '2'
HTTP: token len 1: '.'
...
```

Related Commands

debug ip http ezsetup
debug ip http transaction
debug ip http url

debug ip http transaction

Use the **debug ip http transaction** EXEC command to display Hypertext Transfer Protocol (HTTP) server transaction processing. The **no** form of this command disables debugging output.

[no] debug ip http transaction

Usage Guidelines

Use the **debug ip http transaction** command to display what the HTTP server is parsing at a high level. To display what the HTTP server is parsing at a low level, use the **debug ip http token** command.

Sample Display

The following is sample output from the **debug ip http transaction** command. In this example, the browser accessed the router's home page `http://router-name/`.

```
Router# debug ip http transaction

HTTP: parsed uri '/'
HTTP: client version 1.0
HTTP: parsed extension Referer
HTTP: parsed line http://www.company.com/
HTTP: parsed extension Connection
HTTP: parsed line Keep-Alive
HTTP: parsed extension User-Agent
HTTP: parsed line Mozilla/2.01 (X11; I; FreeBSD 2.1.0-RELEASE i386)
HTTP: parsed extension Host
HTTP: parsed line router-name
HTTP: parsed extension Accept
HTTP: parsed line image/gif, image/x-xbitmap, image/jpeg, image/
HTTP: parsed extension Authorization
HTTP: parsed authorization type Basic
HTTP: received GET ''
```

Table 58 lists describes some of the fields in the output.

Table 58 Debug IP HTTP Transaction Field Descriptions

Field	Description
HTTP: parsed uri '/'	Uniform resource identifier that is requested.
HTTP: client version 1.0	Client HTTP version.
HTTP: parsed extension Referer	HTTP extension.
HTTP: parsed line http://www.company.com/	Value of HTTP extension.
HTTP: received GET ''	HTTP request method.

Related Commands

debug ip http ezsetup
debug ip http token
debug ip http url

debug ip http url

Use the **debug ip http url** EXEC command to show the uniform resource locators (URLs) accessed from the router. The **no** form of this command disables debugging output.

[no] debug ip http url

Usage Guidelines

Use the **debug ip http url** command to keep track of the URLs that are accessed and to determine from which hosts the URLs are accessed.

Sample Display

The following is sample output from the shows a partial sample of **debug ip http url** command. In this example, the HTTP server accessed the URLs */* and */exec*. The output shows the URL being requested and the IP address of the host requesting the URL.

```
Router# debug ip http url

HTTP: processing URL '/' from host 172.31.2.141
HTTP: processing URL '/exec' from host 172.31.2.141
```

Related Commands

debug ip http ezsetup
debug ip http token
debug ip http transaction

debug ip icmp

Use the **debug ip icmp** EXEC command to display information on Internal Control Message Protocol (ICMP) transactions. The **no** form of this command disables debugging output.

[no] debug ip icmp

Usage Guidelines

This command helps you determine whether the router is sending or receiving ICMP messages. Use it, for example, when you are troubleshooting an end-to-end connection problem.

Note For more information about the fields in **debug ip icmp** output, see RFC-792, “Internet Control Message Protocol”; Appendix I of RFC-950, “Internet Standard Subnetting Procedure”; and RFC-1256, “ICMP Router Discovery Messages.”

Sample Display

The following is sample output from the **debug ip icmp** command:

```
Router# debug ip icmp

ICMP: rcvd type 3, code 1, from 10.95.192.4
ICMP: src 10.56.0.202, dst 172.16.16.1, echo reply
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: src 172.16.12.35, dst 172.16.20.7, echo reply
ICMP: dst (255.255.255.255) protocol unreachable rcv from 10.31.7.21
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: dst (255.255.255.255) protocol unreachable rcv from 10.31.7.21
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: src 10.56.0.202, dst 172.16.16.1, echo reply
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: dst (255.255.255.255) protocol unreachable rcv from 10.31.7.21
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
```

Table 59 describes significant fields in the first line of **debug ip icmp** output.

Table 59 **Debug IP ICMP Field Descriptions—Part 1**

Field	Description
ICMP:	Indication that this message describes an ICMP packet.
rcvd type 3	<p>The type field can be one of the following:</p> <ul style="list-style-type: none"> • 0—Echo Reply • 3—Destination Unreachable • 4—Source Quench • 5—Redirect • 8—Echo • 9—Router Discovery Protocol Advertisement • 10—Router Discovery Protocol Solicitations • 11—Time Exceeded • 12—Parameter Problem • 13—Timestamp • 14—Timestamp Reply • 15—Information Request • 16—Information Reply • 17—Mask Request • 18—Mask Reply
code 1	<p>This field is a code. The meaning of the code depends upon the type field value:</p> <ul style="list-style-type: none"> • Echo and Echo Reply—The code field is always zero. • Destination Unreachable—The code field can have the following values: <ul style="list-style-type: none"> — 0—Network unreachable — 1—Host unreachable — 2—Protocol unreachable — 3—Port unreachable — 4—Fragmentation needed and DF bit set — 5—Source route failed • Source Quench—The code field is always 0. • Redirect—The code field can have the following values: <ul style="list-style-type: none"> — 0—Redirect datagrams for the network — 1—Redirect datagrams for the host — 2—Redirect datagrams for the command mode of service and network — 3—Redirect datagrams for the command mode of service and host • Router Discovery Protocol Advertisements and Solicitations—The code field is always zero.

Table 59 Debug IP ICMP Field Descriptions—Part 1 (Continued)

Field	Description
code 1 (continued)	<ul style="list-style-type: none"> • Time Exceeded—The code field can have the following values: <ul style="list-style-type: none"> — 0—Time to live exceeded in transit — 1—Fragment reassembly time exceeded • Parameter Problem—The code field can have the following values: <ul style="list-style-type: none"> — 0—General problem — 1—Option is missing — 2—Option missing, no room to add • Timestamp and Timestamp Reply—The code field is always zero. • Information Request and Information Reply—The code field is always zero. • Mask Request and Mask Reply—The code field is always zero.
from 10.95.192.4	Source address of the ICMP packet.

Table 60 describes significant fields in the second line of **debug ip icmp** output.

Table 60 Debug IP ICMP Field Descriptions—Part 2

Field	Description
ICMP:	Indication that this message describes an ICMP packet
src 10.5610.120.0.202	The address of the sender of the echo
dst 172.16.16.1	The address of the receiving router
echo reply	Indication the router received an echo reply

Other messages that the **debug ip icmp** command can generate follow.

When an IP router or host sends out an ICMP mask request, the following message is generated when the router sends a mask reply:

```
ICMP: sending mask reply (255.255.255.0) to 172.21.80.23 via Ethernet0
```

The following two lines are examples of the two forms of this message. The first form is generated when a mask reply comes in after the router sends out a mask request. The second form occurs when the router receives a mask reply with a nonmatching sequence and ID. See Appendix I of RFC 950, “Internet Standard Subnetting Procedures,” for details.

```
ICMP: mask reply 255.255.255.0 from 172.21.80.31
ICMP: unexpected mask reply 255.255.255.0 from 172.21.80.32
```

The following output indicates that the router sent a redirect packet to the host at address 172.21.80.31, instructing that host to use the gateway at address 172.21.80.23 in order to reach the host at destination address 172.16.1.111:

```
ICMP: redirect sent to 172.21.80.31 for dest 172.16.1.111 use gw 172.21.80.23
```

The following message indicates that the router received a redirect packet from the host at address 172.21.80.23, instructing the router to use the gateway at address 172.21.80.28 in order to reach the host at destination address 172.21.81.34:

```
ICMP: redirect rcvd from 172.21.80.23 -- for 172.21.81.34 use gw 172.21.80.28
```

The following message is displayed when the router sends an ICMP packet to the source address (172.21.94.31 in this case), indicating that the destination address (172.16.13.33 in this case) is unreachable:

```
ICMP: dst (172.16.13.33) host unreachable sent to 172.21.94.31
```

The following message is displayed when the router receives an ICMP packet from an intermediate address (172.21.98.32 in this case), indicating that the destination address (172.16.13.33 in this case) is unreachable:

```
ICMP: dst (172.16.13.33) host unreachable rcv from 172.21.98.32
```

Depending on the code received (as Table 59 describes), any of the unreachable messages can have any of the following “strings” instead of the “host” string in the message:

```
net
protocol
port
frag. needed and DF set
source route failed
prohibited
```

The following message is displayed when the TTL in the IP header reaches zero and a time exceed ICMP message is sent. The fields are self-explanatory.

```
ICMP: time exceeded (time to live) send to 10.95.1.4 (dest was 172.16.1.111)
```

The following message is generated when parameters in the IP header are corrupted in some way and the parameter problem ICMP message is sent. The fields are self-explanatory.

```
ICMP: parameter problem sent to 128.121.1.50 (dest was 172.16.1.111)
```

Based on the preceding information, the remaining output can be easily understood.

```
ICMP: parameter problem rcvd 172.21.80.32
ICMP: source quench rcvd 172.21.80.32
ICMP: source quench sent to 128.121.1.50 (dest was 172.16.1.111)
ICMP: sending time stamp reply to 172.21.80.45
ICMP: sending info reply to 172.21.80.12
ICMP: rdp advert rcvd type 9, code 0, from 172.21.80.23
ICMP: rdp solicit rcvd type 10, code 0, from 172.21.80.43
```

debug ip igmp

Use the **debug ip igmp** EXEC command to display Internet Group Management Protocol (IGMP) packets received and transmitted, as well as IGMP-host related events. The **no** form of this command disables debugging output.

[no] debug ip igmp

Usage Guidelines

This command helps discover whether the IGMP processes are functioning. In general, if IGMP is not working, the router process never discovers that there is another host on the network that is configured to receive multicast packets. In dense mode this means the packets will be delivered intermittently (a few every 3 minutes). In sparse mode they will never be delivered.

Use this command in conjunction with **debug ip pim** and **debug ip mrouting** to observe additional multicast activity and to see what is happening to the multicast routing process, or why packets are forwarded out of particular interfaces.

Sample Display

The following is sample output from the **debug ip igmp** command:

```
Router# debug ip igmp

IGMP: Received Host-Query from 172.24.37.33 (Ethernet1)
IGMP: Received Host-Report from 172.24.37.192 (Ethernet1) for 224.0.255.1
IGMP: Received Host-Report from 172.24.37.57 (Ethernet1) for 224.2.127.255
IGMP: Received Host-Report from 172.24.37.33 (Ethernet1) for 225.2.2.2
```

The messages displayed by the **debug ip igmp** command show query and report activity received from other routers and multicast group addresses.

Related Commands

debug ip mrouting
debug ip pim

debug ip igrp events

Use the **debug ip igrp events** EXEC command to display summary information on Interior Gateway Routing Protocol (IGRP) routing messages that indicates the source and destination of each update, as well as the number of routes in each update. Messages are not generated for each route. The **no** form of this command disables debugging output.

```
[no] debug ip igrp events [ip-address]
```

Syntax Description

ip-address (Optional) IP address of an IGRP neighbor.

Usage Guidelines

If the IP address of an IGRP neighbor is specified, the resulting **debug ip igrp events** output includes messages describing updates from that neighbor and updates that the router broadcasts toward that neighbor.

This command is particularly useful when there are many networks in your routing table. In this case, using **debug ip igrp transactions** could flood the console and make the router unusable. Use **debug ip igrp events** instead to display summary routing information.

Sample Display

The following is sample output from the **debug ip igrp events** command:

```
router# debug ip igrp events
Updates sent to these two destination addresses — IGRP: sending update to 255.255.255.255 via Ethernet1 (160.89.33.8)
                                                    IGRP: Update contains 26 interior, 40 system, and 3 exterior routes.
                                                    IGRP: Total routes in update: 69
Updates received from these source addresses — IGRP: sending update to 255.255.255.255 via Ethernet0 (160.89.32.8)
                                                    IGRP: Update contains 1 interior, 0 system, and 0 exterior routes.
                                                    IGRP: Total routes in update: 1
                                                    IGRP: received update from 160.89.32.24 on Ethernet0
                                                    IGRP: Update contains 17 interior, 1 system, and 0 exterior routes.
                                                    IGRP: Total routes in update: 18
                                                    IGRP: received update from 160.89.32.7 on Ethernet0
                                                    IGRP: Update contains 5 interior, 1 system, and 0 exterior routes.
                                                    IGRP: Total routes in update: 6
```

This shows that the router has sent two updates to the broadcast address 255.255.255.255. The router also received two updates. Three lines of output describe each of these updates.

The first line indicates whether the router sent or received the update packet, the source or destination address, and the interface through which the update was sent or received. If the update was sent, the IP address assigned to this interface is shown (in parentheses).

```
IGRP: sending update to 255.255.255.255 via Ethernet1 (160.89.33.8)
```

The second line summarizes the number and types of routes described in the update:

```
IGRP: Update contains 26 interior, 40 system, and 3 exterior routes.
```

The third line indicates the total number of routes described in the update:

```
IGRP: Total routes in update: 69
```

debug ip igrp transactions

Use the **debug ip igrp transactions** EXEC command to display transaction information on Interior Gateway Routing Protocol (IGRP) routing transactions. The **no** form of this command disables debugging output.

[no] debug ip igrp transactions [*ip-address*]

Syntax Description

ip-address (Optional) IP address of an IGRP neighbor.

Usage Guidelines

If the IP address of an IGRP neighbor is specified, the resulting **debug ip igrp transactions** output includes messages describing updates from that neighbor and updates that the router broadcasts toward that neighbor.

When there are many networks in your routing table, **debug ip igrp transactions** can flood the console and make the router unusable. In this case, use **debug ip igrp events** instead to display summary routing information.

Sample Display

The following is sample output from the **debug ip igrp transactions** command:

```

Router# debug ip igrp transactions

Updates sent to these two source addresses
-----
IGRP: received update from 160.89.80.240 on Ethernet
 subnet 160.89.66.0, metric 1300 (neighbor 1200)
 subnet 160.89.56.0, metric 8676 (neighbor 8576)
 subnet 160.89.48.0, metric 1200 (neighbor 1100)
 subnet 160.89.50.0, metric 1300 (neighbor 1200)
 subnet 160.89.40.0, metric 8676 (neighbor 8576)
 network 192.82.152.0, metric 158550 (neighbor 158450)
 network 192.68.151.0, metric 1115511 (neighbor 1115411)
 network 150.136.0.0, metric 16777215 (inaccessible)
 exterior network 129.140.0.0, metric 9676 (neighbor 9576)
 exterior network 140.222.0.0, metric 9676 (neighbor 9576)
IGRP: received update from 160.89.80.28 on Ethernet
 subnet 160.89.95.0, metric 180671 (neighbor 180571)
 subnet 160.89.81.0, metric 1200 (neighbor 1100)
 subnet 160.89.15.0, metric 16777215 (inaccessible)

Updates received from these two destination addresses
-----
IGRP: sending update to 255.255.255.255 via Ethernet0 (160.89.64.31)
 subnet 160.89.94.0, metric=847
-----
IGRP: sending update to 255.255.255.255 via Serial11 (160.89.94.31)
 subnet 160.89.80.0, metric=16777215
 subnet 160.89.64.0, metric=1100

```

S2549

The output shows that the router being debugged has received updates from two other routers on the network. The router at source address 160.89.80.240 sent information about ten destinations in the update; the router at source address 160.89.80.28 sent information about three destinations in its update. The router being debugged also sent updates—in both cases to the broadcast address 255.255.255.255 as the destination address.

On the second line the first field refers to the type of destination information: “subnet” (interior), “network” (system), or “exterior” (exterior). The second field is the Internet address of the destination network. The third field is the metric stored in the routing table and the metric advertised by the neighbor sending the information. “Metric... inaccessible” usually means that the neighbor router has put the destination in holddown.

The entries in show that the router is sending updates that are similar, except that the numbers in parentheses are the source addresses used in the IP header. A metric of 16777215 is inaccessible.

Other examples of output that the **debug ip igrp transactions** command can produce follow.

The following entry indicates that the routing table was updated and shows the new edition number (97 in this case) to be used in the next IGRP update:

```
IGRP: edition is now 97
```

Entries such as the following occur on startup or when some event occurs such as an interface transitioning or a user manually clearing the routing table:

```
IGRP: broadcasting request on Ethernet0  
IGRP: broadcasting request on Ethernet1
```

The following type of entry can result when routing updates become corrupted between sending and receiving routers:

```
IGRP: bad checksum from 172.21.64.43
```

An entry such as the following should never appear. If it does, the receiving router has a bug in the software or a problem with the hardware. In either case, contact your technical support representative.

```
IGRP: system 45 from 172.21.64.234, should be system 109
```

debug ip inspect

Use the **debug ip inspect** EXEC command to display messages about CBAC events. The **no** form of this command disables debugging output.

```
[no] debug ip inspect {function-trace | object-creation | object-deletion | events | timers |
                        protocol | detail}
[no] debug ip inspect detail
```

Syntax Description

function-trace	Displays messages about software functions called by CBAC.
object-creation	Displays messages about software objects being created by CBAC. Object creation corresponds to the beginning of CBAC-inspected sessions.
object-deletion	Displays messages about software objects being deleted by CBAC. Object deletion corresponds to the closing of CBAC-inspected sessions.
events	Displays messages about CBAC software events, including information about CBAC packet processing.
timers	Displays messages about CBAC timer events such as when a CBAC idle timeout is reached.
<i>protocol</i>	Displays messages about CBAC-inspected protocol events, including details about the protocol's packets. Refer to Table 61 for a list of <i>protocol</i> keywords.
detail	Use this form of the command in conjunction with other CBAC debugging commands. This causes detailed information to be displayed for all the other enabled CBAC debugging.

Table 61 Protocol Keywords for the debug ip inspect Command

Application Protocol	<i>protocol</i> keyword
Transport Layer Protocols	
TCP	tcp
UDP	udp
Application-Layer Protocols	
CU-See-Me	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the ftp tokens parsed)	ftp-tokens
H.323	h323
UNIX r-commands (rlogin, rexec, rsh)	remd
RealAudio	realaudio
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks

Table 61 Protocol Keywords for the debug ip inspect Command (Continued)

Application Protocol	<i>protocol keyword</i>
TFTP	tftp
VDOLive	vdolive
RPC	rpc
Java applets	http

Sample Display

The following is sample output from the **debug ip inspect function-trace** command:

```
*Mar 2 01:16:16: CBAC FUNC: insp_inspection
*Mar 2 01:16:16: CBAC FUNC: insp_pre_process_sync
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar 2 01:16:16: CBAC FUNC: insp_find_pregen_session
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_get_irc_of_idb
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_create_sis
*Mar 2 01:16:16: CBAC FUNC: insp_inc_halfopen_sis
*Mar 2 01:16:16: CBAC FUNC: insp_link_session_to_hash_table
*Mar 2 01:16:16: CBAC FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC FUNC: insp_listen_state
*Mar 2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar 2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar 2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar 2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar 2 01:16:16: CBAC FUNC: insp_process_syn_packet
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar 2 01:16:16: CBAC FUNC: insp_create_tcp_host_entry
*Mar 2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar 2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar 2 01:16:16: CBAC FUNC: insp_dec_halfopen_sis
*Mar 2 01:16:16: CBAC FUNC: insp_remove_sis_from_host_entry
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
```

This output shows the functions called by CBAC as a session is inspected. Entries with an asterisk (*) after the word “CBAC” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect object-creation** and **debug ip inspect object-deletion** command:

```
*Mar 2 01:18:30: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar 2 01:18:30: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:30: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar 2 01:18:30: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item
25A3634
*Mar 2 01:18:31: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1
```

The following is sample output from the **debug ip inspect object-creation**, **debug ip inspect object-deletion**, and **debug ip inspect events** commands:

```
*Mar 2 01:18:51: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar 2 01:18:51: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:51: CBAC Src 10.1.0.1 Port [1:65535]
*Mar 2 01:18:51: CBAC Dst 10.0.0.1 Port [46406:46406]
*Mar 2 01:18:51: CBAC Pre-gen sis 25A3574 created: 10.1.0.1[1:65535]
30.0.0.1[46406:46406]
*Mar 2 01:18:51: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar 2 01:18:51: CBAC sis 25C1CC4 initiator_addr (10.1.0.1:20) responder_addr
(30.0.0.1:46406) initiator_alt_addr (40.0.0.1:20) responder_alt_addr (10.0.0.1:46406)
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar 2 01:18:51: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item
25A3634
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1
```

The following is sample output from the **debug ip inspect timers** command:

```
*Mar 2 01:19:15: CBAC Timer Init Leaf: Pre-gen sis 25A3574
*Mar 2 01:19:15: CBAC Timer Start: Pre-gen sis 25A3574 Timer: 25A35D8 Time: 30000
milisecs
*Mar 2 01:19:15: CBAC Timer Init Leaf: sis 25C1CC4
*Mar 2 01:19:15: CBAC Timer Stop: Pre-gen sis 25A3574 Timer: 25A35D8
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 30000 milisecs
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 3600000 milisecs
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 5000 milisecs
*Mar 2 01:19:15: CBAC Timer Stop: sis 25C1CC4 Timer: 25C1D5C
```

The following is sample output from the **debug ip inspect tcp** command:

```
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:43: CBAC sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 2544374 TCP P ack 4200176247 seq 4223720032(30)
(10.0.0.1:46409) <= (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 25412F8 TCP P ack 4223720062 seq 4200176247(15)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC sis 25C1CC4 pak 2544734 TCP S seq 4226992037(0) (10.1.0.1:20) =>
(10.0.0.1:46411)
*Mar 2 01:20:43: CBAC* sis 25C1CC4 pak 2541E38 TCP S ack 4226992038 seq 4203405054(0)
(10.1.0.1:20) <= (10.0.0.1:46411)
```

This sample shows TCP packets being processed, and lists the corresponding acknowledge (ACK) packet numbers and sequence (SEQ) numbers. The number of data bytes in the TCP packet is shown in parentheses—for example, (22). For each packet shown, the addresses and port numbers are shown separated by a colon. For example, (10.1.0.1:21) indicates an IP address of 10.1.0.1 and a TCP port number of 21.

Entries with an asterisk (*) after the word “CBAC” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect tcp** and **debug ip inspect detailed** commands:

```
*Mar 2 01:20:58: CBAC* Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar 2 01:20:58: P ack 4223720160 seq 4200176262(22)
*Mar 2 01:20:58: CBAC* Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar 2 01:20:58: CBAC* sis 25A3604 SIS_OPEN
*Mar 2 01:20:58: CBAC* Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1),
len 76,proto=6
*Mar 2 01:20:58: CBAC sis 25A3604 Saving State: SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt
4200176262 r_sndnxt 4223720160 r_rcvwnd 8760
*Mar 2 01:20:58: CBAC* sis 25A3604 pak 2541E38 TCP P ack 4223720160 seq 4200176262(22)
(30.0.0.1:46409) => (40.0.0.1:21)
*Mar 2 01:20:58: CBAC* sis 25A3604 pak 2541E38 SIS_OPEN/ESTAB TCP seq 4200176262(22)
Flags: ACK 4223720160 PSH
*Mar 2 01:20:58: CBAC* sis 25A3604 pak 2541E38 --> SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223720160 i_sndnxt 4200176284 i_rcvwnd 8760 risn 4223719771 r_rcvnxt
4200176262 r_sndnxt 4223720160 r_rcvwnd 8760
*Mar 2 01:20:58: CBAC* sis 25A3604 L4 inspect result: PASS packet 2541E38
(30.0.0.1:46409) (40.0.0.1:21) bytes 22 ftp
*Mar 2 01:20:58: CBAC sis 25A3604 Restoring State: SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223
720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262 r_sndnxt
4223720160 r_rcvwnd 8760
*Mar 2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:58: -----
*Mar 2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:58: -----
*Mar 2 01:20:58: CBAC* Bump up: inspection requires the packet in the process
path(30.0.0.1) (40.0.0.1)
*Mar 2 01:20:58: CBAC Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar 2 01:20:58: P ack 4223720160 seq 4200176262(22)
*Mar 2 01:20:58: CBAC Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar 2 01:20:58: CBAC sis 25A3604 SIS_OPEN
*Mar 2 01:20:58: CBAC Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1),
len 76, proto=6
```

debug ip mbgp dampening

To log route flap dampening activity related to MBGP, use the **debug ip mbgp dampening EXEC** command. The **no** form of this command disables debugging output.

[no] debug ip mbgp dampening [*access-list-number*]

Syntax Description

<i>access-list-number</i>	(Optional) Number of an access list in the range 1 to 99. If an access list number is specified, debugging occurs only for the routes permitted by the access list.
---------------------------	---

Sample Display

The following is sample output from the **debug ip mbgp dampening** command:

```
Router# debug ip mbgp dampening
BGP: charge penalty for 173.19.0.0/16 path 49 with halflife-time 15 reuse/suppress
750/2000
BGP: flapped 1 times since 00:00:00. New penalty is 1000
BGP: charge penalty for 173.19.0.0/16 path 19 49 with halflife-time 15 reuse/suppress
750/2000
BGP: flapped 1 times since 00:00:00. New penalty is 1000
```

debug ip mbgp updates

To log MBGP-related information passed in BGP Update messages, use the **debug ip mbgp updates** EXEC command. The **no** form of this command disables debugging output.

[no] debug ip mbgp updates

Sample Display

The following is sample output from the **debug ip mbgp updates** command

```
Router# debug ip mbgp updates

BGP: NEXT_HOP part 1 net 200.10.200.0/24, neigh 171.69.233.49, next 171.69.233.34
BGP: 171.69.233.49 send UPDATE 200.10.200.0/24, next 171.69.233.34, metric 0, path 33
34 19 49 109 65000 297 3561 6503
BGP: NEXT_HOP part 1 net 200.10.202.0/24, neigh 171.69.233.49, next 171.69.233.34
BGP: 171.69.233.49 send UPDATE 200.10.202.0/24, next 171.69.233.34, metric 0, path 33
34 19 49 109 65000 297 1239 1800 3597
BGP: NEXT_HOP part 1 net 200.10.228.0/22, neigh 171.69.233.49, next 171.69.233.34
BGP: 171.69.233.49 rcv UPDATE about 222.2.2.0/24, next hop 171.69.233.49, path 49 109
metric 0
BGP: 171.69.233.49 rcv UPDATE about 131.103.0.0/16, next hop 171.69.233.49, path 49 109
metric 0
BGP: 171.69.233.49 rcv UPDATE about 206.205.242.0/24, next hop 171.69.233.49, path 49
109 metric 0
BGP: 171.69.233.49 rcv UPDATE about 1.0.0.0/8, next hop 171.69.233.49, path 49 19
metric 0
BGP: 171.69.233.49 rcv UPDATE about 198.1.2.0/24, next hop 171.69.233.49, path 49 19
metric 0
BGP: 171.69.233.49 rcv UPDATE about 171.69.0.0/16, next hop 171.69.233.49, path 49
metric 0
BGP: 171.69.233.49 rcv UPDATE about 172.19.0.0/16, next hop 171.69.233.49, path 49
metric 0
BGP: nettable_walker 172.19.0.0/255.255.0.0 calling revise_route
BGP: revise route installing 172.19.0.0/255.255.0.0 -> 171.69.233.49
BGP: 171.69.233.19 computing updates, neighbor version 267099, table version 267100,
starting at 0.0.0.0
BGP: NEXT_HOP part 1 net 172.19.0.0/16, neigh 171.69.233.19, next 171.69.233.49
BGP: 171.69.233.19 send UPDATE 172.19.0.0/16, next 171.69.233.49, metric 0, path 33 49
BGP: 1 updates (average = 46, maximum = 46)
BGP: 171.69.233.19 updates replicated for neighbors : 171.69.233.34, 171.69.233.49,
171.69.233.56
BGP: 171.69.233.19 1 updates enqueued (average=46, maximum=46)
BGP: 171.69.233.19 update run completed, ran for 0ms, neighbor version 267099, start
version 267100, throttled to 267100, check point net 0.0.0.0
```

debug ip mcache

Use the **debug ip mcache** command to display IP multicast fast-switching events. The **no** form of this command disables debugging output. The **no** form of this command disables debugging output.

[no] debug ip mcache [*name* | *address*]

Syntax Description

name (Optional) Hostname.

address (Optional) Group address.

Usage Guidelines

Use this command when multicast fast-switching appears not to be functioning.

Sample Display

The following is sample output from the **debug ip mcache** command when an IP multicast route is cleared:

```
Router# debug ip mcache

IP multicast fast-switching debugging is on

Router#clear ip mroute *

MRC: Build MAC header for (172.31.60.185/32, 224.2.231.173), Ethernet0
MRC: Fast-switch flag for (172.31.60.185/32, 224.2.231.173), off -> on, caller
ip_mroute_replicate-1
MRC: Build MAC header for (172.31.191.10/32, 224.2.127.255), Ethernet0
MRC: Build MAC header for (172.31.60.152/32, 224.2.231.173), Ethernet0
```

Table 62 provides explanations for representative lines of the **debug ip mcache** output.

Table 62 Debug IP Mcache Descriptions

Field	Description
MRC	Multicast route cache.
Fast-switch flag	Route is fast-switched.
(<i>address</i> /32)	Host route with 32 bits of mask.
off -> on	State has changed.
caller <i>string</i>	The code function that activated the state change.

Related Commands

debug ip dvmrp
debug ip igmp
debug ip igrp transactions
debug ip mrouting
debug ip sd

debug ip mds ipc

To debug MDS interprocessor communication, that is, synchronization between the MFIB on the line card and the multicast routing table in the RP, use the **debug ip mds ipc** EXEC command. The **no** form of this command disables debugging output.

[no] debug ip mds ipc {event | packet}

Syntax Description

event	Displays MDS events when there is a problem.
packet	Displays MDS packets.

Usage Guidelines

Use this command on the line card or RP.

Sample Displays

The following is sample output from the **debug ip mds ipc packet** command:

```
VIP-Slot0# debug ip mds ipc packet
MDFS ipc packet debugging is on
VIP-Slot0#
MDFS: LC sending statistics message to RP with code 0 of size 36
MDFS: LC sending statistics message to RP with code 1 of size 680
MDFS: LC sending statistics message to RP with code 2 of size 200
MDFS: LC sending statistics message to RP with code 3 of size 152
MDFS: LC sending window message to RP with code 36261 of size 8
MDFS: LC received IPC packet of size 60 sequence 36212
```

The following is sample output from the **debug ip mds ipc event** command:

```
VIP-Slot0# debug ip mds ipc event
MDFS: LC received invalid sequence 21 while expecting 20
```

debug ip mds mevent

To debug MFIB route creation, route updates, and so on, use the **debug ip mds mevent EXEC** command. The **no** form of this command disables debugging output.

[no] debug ip mds mevent

Usage Guidelines

Use this command on the line card.

Sample Display

The following is sample output from the **debug ip mds mevent** command:

```
VIP-Slot0# debug ip mds mevent
MDFS mroute event debugging is on
VIP-Slot0#clear ip mdfs for *
VIP-Slot0#
MDFS: Create (*, 239.255.255.255)
MDFS: Create (192.168.1.1/32, 239.255.255.255), RPF POS2/0/0
MDFS: Add OIF for mroute (192.168.1.1/239.255.255.255) on Fddi0/0/0
MDFS: Create (*, 224.2.127.254)
MDFS: Create (192.168.1.1/32, 224.2.127.254), RPF POS2/0/0
MDFS: Add OIF for mroute (192.168.1.1/224.2.127.254) on Fddi0/0/0
MDFS: Create (128.9.160.67/32, 224.2.127.254), RPF POS2/0/0
```

debug ip mds mpacket

To debug multicast distributed switching (MDS) events, such as packet drops, interface drops, and switching failures, use the **debug ip mds mpacket EXEC** command. The **no** form of this command disables debugging output.

[no] debug ip mds mpacket

Usage Guidelines

Use this command on the line card.

Sample Display

```
Router# debug ip mds mpacket
```

debug ip mds process

To debug line card process level events, use the **debug ip mds process EXEC** command. The **no** form of this command disables debugging output.

[no] debug ip mds process

Usage Guidelines

Use this command on the line card or RP.

Sample Display

The following is sample output from the **debug ip mds process** command:

```
Router# debug ip mds process
MDFS process debugging is on
Mar 19 16:15:47.448: MDFS: RP queueing mdb message for (210.115.194.5, 224.2.127.254) to
all linecards
Mar 19 16:15:47.448: MDFS: RP queueing midb message for (210.115.194.5, 224.2.127.254)
to all linecards
Mar 19 16:15:47.628: MDFS: RP servicing low queue for LC in slot 0
Mar 19 16:15:47.628: MDFS: RP servicing low queue for LC in slot 2
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.68.224.10, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.68.224.10, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.69.67.106, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.69.67.106, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (206.14.154.181, 224.2.127.254)
to all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (206.14.154.181, 224.2.127.254)
to all linecards
Mar 19 16:15:48.233: MDFS: RP queueing mdb message for (210.115.194.5, 224.2.127.254) to
all linecards
```

debug ip mpacket

Use the **debug ip mpacket** EXEC command to display IP multicast packets received and transmitted. The **no** form of this command disables debugging output.

[no] debug ip mpacket [detail] [access-list] [group]

Syntax Description

detail	(Optional) Causes the debug ip mpacket command to display IP header information as well as MAC address information.
<i>access-list</i>	(Optional) Access list number.
<i>group</i>	(Optional) Group name or address.

Usage Guidelines

This command displays information for multicast IP packets that are forwarded from this router. By using the *access-list* or *group* argument, you can limit the display to multicast packets from sources described by the access list or a specific multicast group.

Use this command with **debug ip packet** to observe additional packet information.

Note The **debug ip mpacket** command generates lots of messages. Use this command with care so that performance on the network is not affected by the debug message traffic.

Sample Display

The following is sample output from the **debug ip mpacket** command:

```
Router# debug ip mpacket 224.2.0.1

IP: s=10.188.34.54 (Ethernet1), d=224.2.0.1 (Tunnel0), len 88, mforward
IP: s=10.188.34.54 (Ethernet1), d=224.2.0.1 (Tunnel0), len 88, mforward
IP: s=10.188.34.54 (Ethernet1), d=224.2.0.1 (Tunnel0), len 88, mforward
IP: s=10.162.3.27 (Ethernet1), d=224.2.0.1 (Tunnel0), len 68, mforward
```

Table 63 defines fields in the output.

Table 63 Debug IP Mpacket Field Descriptions

Field	Description
IP	An IP packet.
<i>s=address</i>	The source address of the packet.
(Ethernet1)	The name of the interface that received the packet.
<i>d=address</i>	The multicast group address that is the destination for this packet.
(Tunnel0)	The outgoing interface for the packet.
len 88	The number of bytes in the packet. This value will vary depending on the application and the media.

Table 63 **Debug IP Mpacket Field Descriptions (Continued)**

Field	Description
mforward	The packet has been forwarded.
not RPF interface	The interface is not a reverse packet forwarding interface. (See debug ip mrouting .)
RPF lookup failed	The reverse packet forwarding lookup failed. (See debug ip mrouting .)

Related Commands

debug ip dvmrp
debug ip igmp
debug ip mrouting
debug ip packet
debug ip sd

debug ip mrouting

Use the **debug ip mrouting** EXEC command to display changes to the IP multicast routing table. The **no** form of this command disables debugging output.

[no] debug ip mrouting [*group*]

Syntax Description

group (Optional) Group name or address to monitor a single group's packet activity.

Usage Guidelines

This command indicates when the router has made changes to the mroute table. Use the **debug ip pim** and **debug ip mrouting** commands at the same time to obtain additional multicast routing information. In addition, use the **debug ip igmp** command to see why an mroute message is being displayed.

This command generates a large amount of output. Use the optional *group* argument to limit the output to a single multicast group.

Sample Display

The following is sample output from the **debug ip mrouting** command:

```
Router# debug ip mrouting 224.2.0.1

MRT: Delete (10.0.0.0/8, 224.2.0.1)
MRT: Delete (10.4.0.0/16, 224.2.0.1)
MRT: Delete (10.6.0.0/16, 224.2.0.1)
MRT: Delete (10.9.0.0/16, 224.2.0.1)
MRT: Delete (10.16.0.0/16, 224.2.0.1)
MRT: Create (*, 224.2.0.1), if_input NULL
MRT: Create (172.24.15.0/24, 225.2.2.4), if_input Ethernet0, RPF nbr 172.16.61.15
MRT: Create (172.24.39.0/24, 225.2.2.4), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.0.0.0/8, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.4.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.6.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.9.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.16.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
```

The following lines show that multicast IP routes were deleted from the routing table:

```
MRT: Delete (10.0.0.0/8, 224.2.0.1)
MRT: Delete (10.4.0.0/16, 224.2.0.1)
MRT: Delete (10.6.0.0/16, 224.2.0.1)
```

The *,G entry in the following line is always null since it is a *,G. The *,G entries are generally created by receipt of an IGMP host-report from a group member on the directly connected LAN or by a PIM join message (in sparse mode) which this router receives from a router that is sending joins toward the RP. This router will in turn, send a join toward the RP which creates the shared tree (or RP tree).

```
MRT: Create (*, 224.2.0.1), if_input NULL
```

The following lines are an example of creating an S,G entry that show a mpacket was received on E0. The second line shows a route being created for a source that is on a directly connected LAN. The RPF means “reverse path forwarding,” whereby the router looks up the source address of the multicast packet in the unicast routing table and asks which interface will be used to send a packet to that source.

```
MRT: Create (172.24.15.0/24, 225.2.2.4), if_input Ethernet0, RPF nbr 172.16.61.15
MRT: Create (172.24.39.0/24, 225.2.2.4), if_input Ethernet1, RPF nbr 0.0.0.0
```

The following lines show that multicast IP routes were added to the routing table. Note the 0.0.0.0 as the RPF, which means the route was created by a source that is directly connected to this router.

```
MRT: Create (10.9.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.16.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
```

If the source is not directly connected, the nbr address shown in these lines will be the address of the router that forwarded the packet to this router.

The shortest path tree state maintained in routers consists of source (S), multicast address (G), outgoing interface (OIF), and incoming interface (IIF). The forwarding information is referred to as the multicast forwarding entry for (S,G).

An entry for a shared tree can match packets from any source for its associated group if the packets come through the proper incoming interface as determined by the RPF lookup. Such an entry is denoted as (*,G). A (*,G) entry keeps the same information a (S,G) entry keeps, except that it saves the rendezvous point (RP) address in place of the source address in sparse mode or 0.0.0.0 in dense mode.

Related Commands

```
debug ip dvmrp
debug ip igmp
debug ip pim
debug ip packet
debug ip sd
```

debug ip nat

Use the **debug ip nat** EXEC command to display information about IP packets translated by the IP network address translation (NAT) feature. The **no** form of this command disables debugging output.

[no] debug ip nat [*access-list* | **detailed**]

Syntax Description

<i>access-list</i>	(Optional) Standard IP access list number. If the datagram is not permitted by the specified access list, the related debugging output is suppressed.
detailed	(Optional) Displays debug information in a detailed format.

Usage Guidelines

The NAT feature reduces the need for unique, registered IP addresses. It can also save private network administrators from having to renumber hosts and routers that do not conform to global IP addressing.

Use the **debug ip nat** command to verify the operation of the NAT feature by displaying information about every packet that is translated by the router. The **debug ip nat detailed** command generates a description of each packet considered for translation. This command also outputs information about certain errors or exceptional conditions, such as the failure to allocate a global address.



Caution Because the **debug ip nat** command generates a significant amount of output, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Sample Displays

The following is sample output from the **debug ip nat** command. In this example, the first two lines show the debugging output that a Domain Name System (DNS) request and reply produced. The remaining lines show the debugging output from a Telnet connection from a host on the inside of the network to a host on the outside of the network. All Telnet packets, except for the first packet, were translated in the fast path, as indicated by the asterisk (*).

```
Router# debug ip nat

NAT: s=192.168.1.95->172.31.233.209, d=172.31.2.132 [6825]
NAT: s=172.31.2.132, d=172.31.233.209->192.168.1.95 [21852]
NAT: s=192.168.1.95->172.31.233.209, d=172.31.1.161 [6826]
NAT*: s=172.31.1.161, d=172.31.233.209->192.168.1.95 [23311]
NAT*: s=192.168.1.95->172.31.233.209, d=172.31.1.161 [6827]
NAT*: s=192.168.1.95->172.31.233.209, d=172.31.1.161 [6828]
NAT*: s=172.31.1.161, d=172.31.233.209->192.168.1.95 [23313]
NAT*: s=172.31.1.161, d=172.31.233.209->192.168.1.95 [23325]
```

Table 64 describes the fields and messages.

Table 64 Debug IP NAT Field Descriptions

Field	Description
NAT:	Indicates that the packet is being translated by the network address translation feature. An asterisk (*) indicates the translation is occurring in the fast path. The first packet in a conversation always goes through the slow path (that is, process-switched). The remaining packets go through the fast path if a cache entry exists.
s=192.168.1.95->172.31.233.209	Source address of the packet and how it is being translated.
d=172.31.2.132	Destination address of the packet.
[6825]	IP identification number of the packet. Might be useful in the debugging process to correlate with other packet traces from protocol analyzers.

The following is sample output from the **debug ip nat detailed** command. In this example, the first two lines show the debugging output that a Domain Name System (DNS) request and reply produced. The remaining lines show the debugging output from a Telnet connection from a host on the inside of the network to a host on the outside of the network. In this example, the inside host 192.168.1.95 was assigned the global address 172.31.233.193.

```
Router# debug ip nat detailed

NAT: i: udp (192.168.1.95, 1493) -> (172.31.2.132, 53) [22399]
NAT: o: udp (172.31.2.132, 53) -> (172.31.233.193, 1493) [63671]
NAT*: i: tcp (192.168.1.95, 1135) -> (172.31.2.75, 23) [22400]
NAT*: o: tcp (172.31.2.75, 23) -> (172.31.233.193, 1135) [22002]
NAT*: i: tcp (192.168.1.95, 1135) -> (172.31.2.75, 23) [22401]
NAT*: i: tcp (192.168.1.95, 1135) -> (172.31.2.75, 23) [22402]
NAT*: o: tcp (172.31.2.75, 23) -> (172.31.233.193, 1135) [22060]
NAT*: o: tcp (172.31.2.75, 23) -> (172.31.233.193, 1135) [22071]
```

Table 65 describes the fields and messages shown in the output.

Table 65 Debug IP NAT Detailed Field Descriptions

Field	Description
NAT:	Indicates that the packet is being translated by the network address translation feature. An asterisk (*) indicates the translation is occurring in the fast path.
i:	Indicates that the packet is moving from a host inside the network to one outside the network.
o:	Indicates that the packet is moving from a host outside the network to one inside the network.
udp	Protocol of the packet.
(192.168.1.95, 1493) -> (172.31.2.132, 53)	Indicates that the packet is sent from IP address 192.168.1.95 port number 1493 to IP address 172.31.2.132 port number 53.
[22399]	IP identification number of the packet.

debug ip ospf events

Use the **debug ip ospf events** EXEC command to display information on Open Shortest Path First (OSPF)-related events, such as adjacencies, flooding information, designated router selection, and shortest path first (SPF) calculation. The **no** form of this command disables debugging output.

[no] debug ip ospf events

Sample Display

The following is sample output from the **debug ip ospf events** command:

```
Router# debug ip ospf events

OSPF:hello with invalid timers on interface Ethernet0
hello interval received 10 configured 10
net mask received 255.255.255.0 configured 255.255.255.0
dead interval received 40 configured 30
```

The **debug ip ospf events** output shown might appear if any of the following occurs:

- The IP subnet masks for routers on the same network do not match.
- The OSPF hello interval for the router does not match that configured for a neighbor.
- The OSPF dead interval for the router does not match that configured for a neighbor.

If a router configured for OSPF routing is not seeing an OSPF neighbor on an attached network, do the following:

- Make sure that both routers have been configured with the same IP mask, OSPF hello interval, and OSPF dead interval.
- Make sure that both neighbors are part of the same area type.

In the following example line, the neighbor and this router are not part of a stub area (that is, one is a part of a transit area and the other is a part of a stub area, as explained in RFC 1247):

```
OSPF: hello packet with mismatched E bit
```

Related Command

debug ip ospf packet

debug ip ospf packet

Use the **debug ip ospf packet** EXEC command to display information about each Open Shortest Path First (OSPF) packet received. The **no** form of this command disables debugging output.

[no] debug ip ospf packet

Sample Display

The following is sample output from the **debug ip ospf packet** command:

```
Router# debug ip ospf packet

OSPF: rcv. v:2 t:1 l:48 rid:200.0.0.117
      aid:0.0.0.0 chk:6AB2 aut:0 auk:
```

The **debug ip ospf packet** command produces one set of information for each packet received. The output varies slightly depending on which authentication is used. The following is sample output from the **debug ip ospf packet** command when MD5 authentication is used.

```
Router# debug ip ospf packet

OSPF: rcv. v:2 t:1 l:48 rid:200.0.0.116
      aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x0
```

Table 66 describes the fields shown in the outputs.

Table 66 Debug IP OSPF Packet Field Descriptions

Field	Description
v:	OSPF version.
t:	OSPF packet type. Possible packet types follow: 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 follow: 0—No authentication 1—Simple password 2—MD5
auk:	OSPF authentication key.
keyid:	MD5 key ID.
seq:	Sequence number.

Related Command

debug ip ospf events

debug ip packet

Use the **debug ip packet** EXEC command to display general IP debugging information and IP security option (IPSO) security transactions. The **no** form of this command disables debugging output.

[no] debug ip packet [*access-list-number*] [**detail**] [**dump**]

Syntax Description

<i>access-list-number</i>	(Optional) IP access list number that you can specify. If the datagram is not permitted by that access list, the related debugging output is suppressed.
detail	(Optional) Displays detailed IP packet debugging information. This information includes the packet types and codes as well as source and destination port numbers.
dump	(Hidden) Displays IP packet debugging information along with raw packet data in hexadecimal and ASCII forms. This keyword can be enabled with individual access lists and also with the detail keyword.

Note The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution notes below, in the usage guidelines, for more specific information.

Usage Guidelines

If a communication session is closing when it should not be, an end-to-end connection problem can be the cause. The **debug ip packet** command is useful for analyzing the messages traveling between the local and remote hosts. IP packet debugging captures the packets that are process switched including received, generated and forwarded packets. IP packets that are switched in the fast path are not captured.

IPSO security transactions include messages that describe the cause of failure each time a datagram fails a security test in the system. This information is also sent to the sending host when the router configuration allows it.

Caution Because the **debug ip packet** command generates a substantial amount of output and uses a substantial amount of system resources, this command should be used with caution in production networks. It should only be enabled when traffic on the IP network is low, so other activity on the system is not adversely affected. Enabling the **detail** and **dump** keywords use the highest level of system resources of the available configuration options for this command, so a high level of caution should be applied when enabling either of these keywords.

Caution The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. Because of the risk of using significant CPU utilization, the dump keyword is hidden from the user and cannot be seen using the “?” prompt. The length of the displayed packet information may exceed the actual packet length and include additional padding bytes that do not belong to the IP packet. Also note that the beginning of a packet may start at different locations in the dump output depending on the specific router, interface type, and packet header processing that may have occurred before the output is displayed.

Sample Display

The following is sample output from the **debug ip packet** command:

```
Router# debug ip packet

IP: s=172.16.13.44 (Fddi0), d=10.125.254.1 (Serial2), g=172.16.16.2, forward
IP: s=172.16.1.57 (Ethernet4), d=10.36.125.2 (Serial2), g=172.16.16.2, forward
IP: s=172.16.1.6 (Ethernet4), d=255.255.255.255, rcvd 2
IP: s=172.16.1.55 (Ethernet4), d=172.16.2.42 (Fddi0), g=172.16.13.6, forward
IP: s=172.16.89.33 (Ethernet2), d=10.130.2.156 (Serial2), g=172.16.16.2, forward
IP: s=172.16.1.27 (Ethernet4), d=172.16.43.126 (Fddi1), g=172.16.23.5, forward
IP: s=172.16.1.27 (Ethernet4), d=172.16.43.126 (Fddi0), g=172.16.13.6, forward
IP: s=172.16.20.32 (Ethernet2), d=255.255.255.255, rcvd 2
IP: s=172.16.1.57 (Ethernet4), d=10.36.125.2 (Serial2), g=172.16.16.2, access denied
```

The output shows two types of messages that the **debug ip packet** command can produce; the first line of output describes an IP packet that the router forwards, and the third line of output describes a packet that is destined for the router. In the third line of output, “rcvd 2” indicates that the router decided to receive the packet.

Table 67 describes the fields shown in the first line.

Table 67 Debug IP Packet Field Descriptions

Field	Description
IP:	Indicates that this is an IP packet.
s = 172.16.13.44 (Fddi0)	Indicates the source address of the packet and the name of the interface that received the packet.
d = 10.125.254.1 (Serial2)	Indicates the destination address of the packet and the name of the interface (in this case, S2) through which the packet is being sent out on the network.
g = 172.16.16.2	Indicates the address of the next hop gateway.
forward	Indicates that the router is forwarding the packet. If a filter denies a packet, “access denied” replaces “forward,” as shown in the last line of output.

The following is sample output from the **debug ip packet** command enabled with the **detail** keyword:

debug ip packet detail

```
IP packet debugging is on (detailed)

001556: 19:59:30: CEF: Try to CEF switch 10.4.9.151 from FastEthernet0/0
001557: 19:59:30: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.151 (FastEthernet03
001558: 19:59:30:     TCP src=179, dst=11001, seq=3736598846, ack=2885081910, wH
001559: 20:00:09: CEF: Try to CEF switch 10.4.9.151 from FastEthernet0/0
001560: 20:00:09: IP: s=10.4.9.4 (FastEthernet0/0), d=10.4.9.151 (FastEthernet03
001561: 20:00:09:     TCP src=179, dst=11000, seq=163035693, ack=2948141027, wiH
001562: 20:00:14: CEF: Try to CEF switch 10.4.9.151 from FastEthernet0/0
001563: 20:00:14: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.151 (FastEthernet03
001564: 20:00:14:     ICMP type=8, code=0
001565: 20:00:14: IP: s=10.4.9.151 (local), d=10.4.9.6 (FastEthernet0/0), len 1g
001566: 20:00:14:     ICMP type=0, code=0
```

The format of the output with **detail** keyword provides additional information, such as the packet type, code, some field values, and source and destination port numbers.

Table 68 Table 68 describes the significant fields shown in the output.

Table 68 debug ip packet detail Field Descriptions

Field	Description
CEF:	Indicates that the IP packet is being processed by CEF.
IP:	Indicates that this is an IP packet.
s=10.4.9.6 (FastEthernet0/0)	Indicates the source address of the packet and the name of the interface that received the packet.
d=10.4.9.151 (FastEthernet03)	Indicates the destination address of the packet and the name of the interface through which the packet is being sent out on the network.
TCP src=	Indicates the source TCP port number.
dst=	Indicates the destination TCP port number.
seq=	Value from the TCP packet sequence number field./
ack=	Value from the TCP packet acknowledgement field.
ICMP type=	Indicates ICMP packet type.
code=	Indicates ICMP return code.

The following is sample output from the **debug ip packet** command enabled with the **dump** keyword:

```

debug ip packet dump

IP packet debugging is on (detailed) (dump)

21:02:42: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.4 (FastEthernet0/0), len 13
07003A00:          0005 00509C08          ...P..
07003A10: 0007855B 4DC00800 45000064 001E0000 ...[M@..E..d....
07003A20: FE019669 0A040906 0A040904 0800CF7C ~..i.....O|
07003A30: 0D052678 00000000 0A0B7145 ABCDABCD ..&x.....qE+M+M
07003A40: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
07003A50: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
07003A60: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
07003A70: ABCDABCD ABCDABCD ABCDABCD          +M+M+M+M+M+M
21:02:42: IP: s=10.4.9.4 (local), d=10.4.9.6 (FastEthernet0/0), len 100, sending
07003A00:          0005 00509C08          ...P..
07003A10: 0007855B 4DC00800 45000064 001E0000 ...[M@..E..d....
07003A20: FF019569 0A040904 0A040906 0000D77C ...i.....W|
07003A30: 0D052678 00000000 0A0B7145 ABCDABCD ..&x.....qE+M+M
07003A40: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
07003A50: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
07003A60: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
07003A70: ABCDABCD ABCDABCD ABCDABCD          +M+M+M+M+M+M
21:02:42: CEF: Try to CEF switch 10.4.9.4 from FastEthernet0/0
21:02:42: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.4 (FastEthernet0/0), len 13
07003380:          0005 00509C08          ...P..
07003390: 0007855B 4DC00800 45000064 001F0000 ...[M@..E..d....
070033A0: FE019668 0A040906 0A040904 0800CF77 ~..h.....Ow
070033B0: 0D062678 00000000 0A0B7149 ABCDABCD ..&x.....qI+M+M
070033C0: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
070033D0: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
070033E0: ABCDABCD ABCDABCD ABCDABCD ABCDABCD +M+M+M+M+M+M+M+M
070033F0: ABCDABCD ABCDABCD ABCDABCD          +M+M+M+M+M+M

```

Note The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution in the usage guidelines section of this command reference page for more specific information.

The output from the **debug ip packet** command, when the **dump** keyword is enabled, provides raw packet data in hexadecimal and ASCII forms. This additional output is displayed in addition to the standard output. The **dump** keyword can be used with all of the available configuration options of this command.

Table 69 Table 69 describes the standard output fields shown.

Table 69 debug ip packet dump Field Descriptions

Field	Description
IP:	Indicates that this is an IP packet.
s=10.4.9.6 (FastEthernet0/0)	Indicates the source address of the packet and the name of the interface that received the packet.
d=10.4.9.4 (FastEthernet0/0) len 13	Indicates destination address and length of the packet and the name of the interface through which the packet is being sent out on the network.
sending	Indicates that the router is sending the packet.

The calculation on whether to send a security error message can be somewhat confusing. It depends upon both the security label in the datagram and the label of the incoming interface. First, the label contained in the datagram is examined for anything obviously wrong. If nothing is wrong, assume it to be correct. If there is something wrong, the datagram is treated as *unclassified genser*. Then the label is compared with the interface range, and the appropriate action is taken as Table 70 describes.

Table 70 Security Actions

Classification	Authorities	Action Taken
Too low	Too low	No Response
	Good	No Response
	Too high	No Response
In range	Too low	No Response
	Good	Accept
	Too high	Send Error
Too high	Too low	No Response
	In range	Send Error
	Too high	Send Error

The security code can only generate a few types of ICMP error messages. The only possible error messages and their meanings follow:

- “ICMP Parameter problem, code 0”—Error at pointer
- “ICMP Parameter problem, code 1”—Missing option
- “ICMP Parameter problem, code 2”—See Note that follows
- “ICMP Unreachable, code 10”—Administratively prohibited

Note The message “ICMP Parameter problem, code 2” identifies a specific error that occurs in the processing of a datagram. This message indicates that the router received a datagram containing a maximum length IP header but no security option. After being processed and routed to another interface, it is discovered that the outgoing interface is marked with “add a security label.” Since the IP header is already full, the system cannot add a label and must drop the datagram and return an error message.

When an IP packet is rejected due to an IP security failure, an audit message is sent via DNSIX NAT. Also, any **debug ip packet** output is appended to include a description of the reason for rejection. This description can be any of the following:

- No basic
- No basic, no response
- Reserved class
- Reserved class, no response
- Class too low, no response
- Class too high
- Class too high, bad authorities, no response

- Unrecognized class
- Unrecognized class, no response
- Multiple basic
- Multiple basic, no response
- Authority too low, no response
- Authority too high
- Compartment bits not dominated by maximum sensitivity level
- Compartment bits do not dominate minimum sensitivity level
- Security failure: extended security disallowed
- NLESO source appeared twice
- ESO source not found
- Postroute, failed xfc out
- No room to add IPSO