

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

Figure 2-92 shows sample **debug ip drp** output. This example shows the packet origination, the IP address that information is routed to, and the route metrics that were returned.

Figure 2-92 Sample Debug IP DRP Output

```
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 2-49 describes the fields shown in Figure 2-92.

Table 2-49 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

Figure 2-93 shows sample **debug ip dvmrp** output.

Figure 2-93 Sample Debug IP DVMRP Output

```
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
```

Explanations for individual lines of output from Figure 2-93 follow.

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 2-50 provides a list of internet multicast addresses supported for host IP implementations.

Table 2-50 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 mrouterd. 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
```

Figure 2-94 shows sample **debug ip dvmrp detail** output.

Figure 2-94 Sample Debug IP DVMRP Detail Output

```
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
```

Explanations for individual lines of output from Figure 2-94 follow.

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 IGRP protocol 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

Figure 2-95 shows sample **debug ip eigrp** output.

Figure 2-95 Sample Debug IP EIGRP Output

```
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 2-51 describes significant fields in the debug messages shown in Figure 2-95.

Table 2-51 Debug IP EIGRP Field Descriptions

Field	Description
IP-EIGRP:	Indicates that this is an IP Enhanced IGRP packet.
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 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.
DRP: RTQUERY for 172.31.58.94 returned internal=0, external=0
```

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

Figure 2-96 shows sample **debug ip http authentication** output.

Figure 2-96 Sample Debug IP HTTP Authentication Output

```
Router# debug ip http authentication

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

Table 2-52 describes the output fields created by the **debug ip http authentication** command.

Table 2-52 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

Figure 2-97 shows sample **debug ip http ezsetup** output. This example shows the configuration changes for the router when the EZ Setup form has been submitted.

Figure 2-97 Sample Debug IP HTTP EZ Setup Output

```
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
```

```
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

Figure 2-98 shows sample **debug ip http ssi** command output.

Figure 2-98 Sample Debug IP HTTP SSI Output

```
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
```

Explanations for individual lines of output from Figure 2-98 follow.

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

Sample Display

Figure 2-99 shows a partial sample of **debug ip http token** output. 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.

Figure 2-99 Sample Debug IP HTTP Token Output

```
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: 'foo'
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

Figure 2-100 shows sample **debug ip http transaction** output. In this example, the browser accessed the router's home page *http://router-name/*.

Figure 2-100 Sample Debug IP HTTP Transaction Output

```
Router# debug ip http transaction

HTTP: parsed uri '/'
HTTP: client version 1.0
HTTP: parsed extension Referer
HTTP: parsed line http://www.foo.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 2-53 lists describes some of the fields in Figure 2-100.

Table 2-53 Debut 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.foo.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

Figure 2-101 shows a partial sample of **debug ip http url** output. 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.

Figure 2-101 Sample Debug IP HTTP URL Output

```
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

Figure 2-102 shows sample **debug ip icmp** output.

Figure 2-102 Sample Debug IP ICMP Output

```
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 2-54 describes significant fields in the first line of **debug ip icmp** output shown in Figure 2-102.

Table 2-54 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 2-54 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 2-55 describes significant fields in the second line of **debug ip icmp** output in Figure 2-102.

Table 2-55 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 2-54 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

Figure 2-103 shows sample **debug ip igmp** output.

Figure 2-103 Sample Debug IP IGMP Output

```
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 pim

debug ip mrouting

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

Figure 2-104 shows sample **debug ip igrp events** output.

Figure 2-104 Sample Debug IP IGRP Events Output

```

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

```

S2548

Figure 2-104 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

Figure 2-105 shows sample **debug ip igrp transactions** output.

Figure 2-105 Sample Debug IP IGRP Transactions Output

```
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
```

64828

Figure 2-105 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.

The first line in Figure 2-105 is self-explanatory.

On the second line in Figure 2-105, 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 Figure 2-105 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 mcache

Use the **debug ip mcache** command to display IP multicast fast-switching events. 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

Figure 2-106 shows sample **debug ip mcache** output when an IP multicast route is cleared.

Figure 2-106 Sample Debug IP Mcache Output

```
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 2-56 provides explanations for representative lines of the **debug ip mcache** output shown in Figure 2-106.

Table 2-56 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 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

Figure 2-107 shows sample **debug ip mpacket** output.

Figure 2-107 Sample Debug IP Mpacket Output

```
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 2-57 defines fields shown in Figure 2-107.

Table 2-57 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.

Table 2-57 Debug IP Mpacket Field Descriptions (Continued)

Field	Description
(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.
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 tells 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* to limit the output to a single multicast group.

Sample Display

Figure 2-108 shows sample **debug ip mrouting** output.

Figure 2-108 Sample Debug IP Mrouting Output

```
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
```

Explanations for individual lines of output from Figure 2-108 follow.

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 packet 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

Figure 2-109 shows sample **debug ip nat** output. 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 (*).

Figure 2-109 Sample Debug IP NAT Output

```
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 2-58 describes the fields and messages shown in Figure 2-109.

Table 2-58 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.

Figure 2-110 shows sample **debug ip nat detailed** output. 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.

Figure 2-110 Sample Debug IP NAT Detailed Output

```
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 2-59 describes the fields and messages shown in Figure 2-110.

Table 2-59 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

Figure 2-111 shows sample **debug ip ospf events** output.

Figure 2-111 Sample Debug IP OSPF Events Output

```
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 in Figure 2-111 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

Figure 2-112 shows sample **debug ip ospf packet** output.

Figure 2-112 Sample Debug IP OSPF Packet Output

```
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. Figure 2-113 shows sample **debug ip ospf packet** output when MD5 authentication is used.

Figure 2-113 Sample Debug IP OSPF Packet Output—MD5 Authentication

```
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 2-60 describes the fields shown in Figure 2-112 and Figure 2-113.

Table 2-60 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

Table 2-60 Debug IP OSPF Packet Field Descriptions (Continued)

Field	Description
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]
```

Syntax Description

access-list-number (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.

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 debugging information includes packets received, generated, and forwarded. Fast-switched packets do not generate messages.

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.

Note Because the **debug ip packet** 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 Display

Figure 2-114 shows sample **debug ip packet** output.

Figure 2-114 Sample Debug IP Packet Output

```
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
```

Figure 2-114 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 2-61 describes the fields shown in the first line of Figure 2-114.

Table 2-61 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 in Figure 2-114.

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 generer*. Then the label is compared with the interface range, and the appropriate action is taken as Table 2-62 describes.

Table 2-62 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

debug ip pim

Use the **debug ip pim EXEC** command to display Protocol Independent Multicast (PIM) packets received and transmitted as well as PIM related events. The **no** form of this command disables debugging output.

```
[no] debug ip pim [group]
```

Syntax Description

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

Usage Guidelines

PIM uses IGMP packets to communicate between routers and advertise reachability information.

Use this command with **debug ip igmp** and **debug ip mrouting** to observe additional multicast routing information.

Sample Display

Figure 2-115 shows sample **debug ip pim** output.

Figure 2-115 Sample Debug IP PIM Output

```
Router# debug ip pim 224.2.0.1

PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Received Join/Prune on Tunnel0 from 10.3.84.1
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Received RP-Reachable on Ethernet1 from 172.16.20.31
PIM: Update RP expiration timer for 224.2.0.1
PIM: Forward RP-reachability packet for 224.2.0.1 on Tunnel0
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Prune-list (10.221.196.51/32, 224.2.0.1)
PIM: Set join delay timer to 2 seconds for (10.221.0.0/16, 224.2.0.1) on Ethernet1
PIM: Received Join/Prune on Ethernet1 from 172.24.37.6
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Received Join/Prune on Tunnel0 from 10.3.84.1
PIM: Join-list: (*, 224.2.0.1) RP 172.16.20.31
PIM: Add Tunnel0 to (*, 224.2.0.1), Forward state
PIM: Join-list: (10.0.0.0/8, 224.2.0.1)
PIM: Add Tunnel0 to (10.0.0.0/8, 224.2.0.1), Forward state
PIM: Join-list: (10.4.0.0/16, 224.2.0.1)
PIM: Prune-list (172.24.84.16/28, 224.2.0.1) RP-bit set RP 172.24.84.16
PIM: Send Prune on Ethernet1 to 172.24.37.6 for (172.24.84.16/28, 224.2.0.1), RP
PIM: For RP, Prune-list: 10.9.0.0/16
PIM: For RP, Prune-list: 10.16.0.0/16
PIM: For RP, Prune-list: 10.49.0.0/16
PIM: For RP, Prune-list: 10.84.0.0/16
PIM: For RP, Prune-list: 10.146.0.0/16
PIM: For 10.3.84.1, Join-list: 172.24.84.16/28
PIM: Send periodic Join/Prune to RP via 172.24.37.6 (Ethernet1)
```

Explanations for individual lines of output from Figure 2-115 follow.

The following lines appear periodically when PIM is running in sparse mode and indicate to this router which multicast groups and multicast sources other routers are interested in:

```
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
PIM: Received Join/Prune on Ethernet1 from 172.24.37.33
```

The following lines appear when a rendezvous point (RP) message is received and the RP timer is reset. The expiration timer sets a checkpoint to make sure the RP still exists; otherwise a new RP must be discovered.

```
PIM: Received RP-Reachable on Ethernet1 from 172.16.20.31
PIM: Update RP expiration timer for 224.2.0.1
PIM: Forward RP-reachability packet for 224.2.0.1 on Tunnel0
```

The prune-list message in the following line states that this router is not interested in the source address information. The prune message tells an upstream router to stop forwarding multicast packets from this source.

```
PIM: Prune-list (10.221.196.51/32, 224.2.0.1)
```

In the following line, a second router on the network wants to override the prune message that the upstream router just received. The timer is set at a random value so that if there are additional routers on the network that still want to receive multicast packets for the group, only one will actually send the message. The other routers will receive the join message and then suppress sending their own message.

```
PIM: Set join delay timer to 2 seconds for (10.221.0.0/16, 224.2.0.1) on Ethernet1
```

In the following line, a join message is sent towards the RP for all sources:

```
PIM: Join-list: (*, 224.2.0.1) RP 172.16.20.31
```

In the following lines, the interface is being added to the outgoing interface (OIF) of the *,G and S,G mroute table entry so that packets from the source will be forwarded out that particular interface:

```
PIM: Add Tunnel0 to (*, 224.2.0.1), Forward state
PIM: Add Tunnel0 to (10.0.0.0/8, 224.2.0.1), Forward state
```

The following line appears in sparse mode only. There are two trees on which data may be received: the RP tree and the source tree. In dense mode there is no RP. After the source and the receiver have discovered one another at the RP, the first-hop router for the receiver will usually join to the source tree rather than the RP tree.

```
PIM: Prune-list (172.24.84.16/28, 224.2.0.1) RP-bit set RP 172.24.84.16
```

The Send Prune message in the next line shows that a router is sending a message to a second router saying that the first router no longer wants to receive multicast packets for the S,G. The “RP” at the end of the message indicates that the router is pruning the RP tree and is most likely joining the source tree, although the router may not have downstream members for the group or downstream routers with members of the group. The output shows which specific sources this router no longer wants to receive multicast from.

```
PIM: Send Prune on Ethernet1 to 172.24.37.6 for (172.24.84.16/28, 224.2.0.1), RP
```

The following lines indicate a prune message is sent toward the RP so that router can join the source tree rather than the RP tree:

```
PIM: For RP, Prune-list: 10.9.0.0/16
PIM: For RP, Prune-list: 10.16.0.0/16
PIM: For RP, Prune-list: 10.49.0.0/16
```

In the following line, a periodic message is sent towards the RP. The default period is once per minute. Prune and join messages are sent toward the RP or source rather than directly to the RP or source. It is the responsibility of the next-hop router to take proper action with this message, such as continuing to forward it to the next router in the tree.

```
PIM: Send periodic Join/Prune to RP via 172.24.37.6 (Ethernet1)
```

Related Commands

debug ip dvmrp

debug ip igmp

debug ip igmp transactions

debug ip mrouting

debug ip sd

debug ip pim atm

To log PIM ATM signaling activity, use the **debug ip pim atm EXEC** command. To disable debugging output, use the **no** form of this command.

```
[no] debug ip pim atm
```

Sample Displays

The sample display in Figure 2-116 shows a new group being created and the router toward the RP opening a new VC. Since there are now two groups on this router, there are two virtual circuits open, as reflected by the “current count.”

Figure 2-116 shows sample **debug ip pim atm** output.

Figure 2-116 Sample Debug IP PIM ATM Output

```
Router# debug ip pim atm

Jan 28 19:05:51: PIM-ATM: Max VCs 200, current count 1
Jan 28 19:05:51: PIM-ATM: Send SETUP on ATM2/0 for 239.254.254.253/171.69.214.43
Jan 28 19:05:51: PIM-ATM: Received CONNECT on ATM2/0 for 239.254.254.253, vcd 19
Jan 28 19:06:35: PIM-ATM: Max VCs 200, current count 2
```

Table 2-63 describes the significant fields in Figure 2-116.

Table 2-63 Debug IP PIM ATM Field Descriptions

Field	Description
Jan 28 19:05:51	Current date and time in hours:minutes:seconds.
PIM-ATM	Indicates what PIM is doing to set up or monitor an ATM connection (vc).
current count	Current number of open virtual circuits.

The resulting **show ip mroute** output follows:

```
Router# show ip mroute 239.254.254.253

IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
      R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.254.254.253), 00:00:04/00:02:53, RP 171.69.214.50, flags: S
  Incoming interface: Ethernet1/1, RPF nbr 171.69.214.50
  Outgoing interface list:
    ATM2/0, VCD 19, Forward/Sparse-Dense, 00:00:04/00:02:52
```

debug ip pim auto-rp

Use the **debug ip pim auto-rp EXEC** command to display the contents of each Protocol Independent Multicast (PIM) packet used in the automatic discovery of group-to-rendezvous point (RP) mapping as well as the actions taken on the address-to-RP mapping database. The **no** form of this command disables debugging output.

[no] debug ip pim auto-rp

Sample Displays

Figure 2-117 shows sample **debug ip pim auto-rp** output.

Figure 2-117 Sample Debug IP PIM Auto-RP Output

```
Router# debug ip pim auto-rp

Auto-RP: Received RP-announce, from 172.31.214.66, RP_cnt 1, holdtime 180 secs
Auto-RP: update (192.168.248.0/24, RP:172.31.214.66)
Auto-RP: Build RP-Discovery packet
Auto-RP: Build mapping (192.168.248.0/24, RP:172.31.214.66),
Auto-RP: Build mapping (192.168.250.0/24, RP:172.31.214.26).
Auto-RP: Build mapping (192.168.254.0/24, RP:172.31.214.2).
Auto-RP: Send RP-discovery packet (3 RP entries)
Auto-RP: Build RP-Announce packet for 172.31.214.2
Auto-RP: Build announce entry for (192.168.254.0/24)
Auto-RP: Send RP-Announce packet, IP source 172.31.214.2, ttl 8
```

Explanations for individual lines of output from Figure 2-117 follow.

The first two lines show a packet received from 172.31.214.66 announcing that it is the rendezvous point (RP) for the groups in 192.168.248.0/24. This announcement contains one RP address and is valid for 180 seconds. The RP-mapping agent then updates its mapping database to include the new information.

```
Auto-RP: Received RP-announce, from 172.31.214.66, RP_cnt 1, holdtime 180 secs
Auto-RP: update (192.168.248.0/24, RP:172.31.214.66)
```

In the next five lines, the router creates an RP-discovery packet containing three RP mapping entries. The packet is sent to the well-known CISCO-RP-DISCOVERY group address (224.0.1.40).

```
Auto-RP: Build RP-Discovery packet
Auto-RP: Build mapping (192.168.248.0/24, RP:172.31.214.66),
Auto-RP: Build mapping (192.168.250.0/24, RP:172.31.214.26).
Auto-RP: Build mapping (192.168.254.0/24, RP:172.31.214.2).
Auto-RP: Send RP-discovery packet (3 RP entries)
```

The final three lines show the router announcing that it intends to be an RP for the groups in 192.168.254.0/24. Only routers inside the scope ttl 8 receive the advertisement and use the RP for these groups.

```
Auto-RP: Build RP-Announce packet for 172.31.214.2
Auto-RP: Build announce entry for (192.168.254.0/24)
Auto-RP: Send RP-Announce packet, IP source 172.31.214.2, ttl 8
```

Figure 2-118 shows sample **debug ip pim auto-rp** output when a router receives an update. In this example, the packet contains three group-to-RP mappings, which are valid for 180 seconds. The RP-mapping agent then updates its mapping database to include the new information.

Figure 2-118 Sample Debug IP PIM Auto-RP Output—Router Receiving Update

```
Router# debug ip pim auto-rp

Auto-RP: Received RP-discovery, from 172.31.214.17, RP_cnt 3, holdtime 180 secs
Auto-RP: update (192.168.248.0/24, RP:172.31.214.66)
Auto-RP: update (192.168.250.0/24, RP:172.31.214.26)
Auto-RP: update (192.168.254.0/24, RP:172.31.214.2)
```

debug ip policy

Use the **debug ip policy** EXEC command to display IP policy routing packet activity. The **no** form of this command disables debugging output.

[no] debug ip policy

Usage Guidelines

After you configure IP policy routing with the **ip policy** and **route map** commands, use the **debug ip policy** command to ensure that the IP policy is configured correctly.

Policy routing looks at various parts of the packet and then routes the packet based on certain user-defined attributes in the packet.

The **debug ip policy** command helps you determine what policy routing is doing. It displays information about whether a packet matches the criteria, and if so, the resulting routing information for the packet.



Caution Because the **debug ip policy** 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.

You can also use the **show ip local policy** command to obtain additional information.

Sample Display

Figure 2-119 shows sample **debug ip policy** output. Line 1 indicates that a packet with the given source and destination addresses matched a policy. Line 2 indicates the clause in the route map that the packet matched. In this case, the packet matches clause 20 in the route map. Line 3 indicates that a second packet did not match the policy.

Figure 2-119 Sample Debug IP Policy Output

```
Router# debug ip policy
IP: s=172.16.232.150 (local), d=172.16.2.75, len 100, policy match
IP: route map equal, item 20, permit
IP: s=172.16.232.150 (local), d=172.16.2.75, len 200, policy rejected -- normal
forwarding
```

debug ip rip

Use the **debug ip rip** EXEC command to display information on RIP routing transactions. The **no** form of this command disables debugging output.

[no] debug ip rip

Sample Display

Figure 2-120 shows sample **debug ip rip** output.

Figure 2-120 Sample Debug IP RIP Output

<p>Updates received from this source address</p> <p>Updates sent to these two destination addresses</p>	<p>—</p> <p>—</p> <p>—</p>	<pre> router# debug ip rip RIP: received update from 10.89.80.28 on Ethernet0 10.89.95.0 in 1 hops 10.89.81.0 in 1 hops 10.89.66.0 in 2 hops 172.31.0.0 in 16 hops (inaccessible) 0.0.0.0 in 7 hop RIP: sending update to 255.255.255.255 via Ethernet0 (10.89.64.31) subnet 10.89.94.0, metric 1 172.31.0.0 in 16 hops (inaccessible) RIP: sending update to 255.255.255.255 via Serial1 (10.89.94.31) subnet 10.89.64.0, metric 1 subnet 10.89.66.0, metric 3 172.31.0.0 in 16 hops (inaccessible) default 0.0.0.0, metric 8 </pre>	<p>92550</p>
---	----------------------------	---	--------------

Figure 2-120 shows that the router being debugged has received updates from one router at source address 160.89.80.28. That router sent information about five destinations in the routing table update. Notice that the fourth destination address in the update—131.108.0.0—is inaccessible because it is more than 15 hops away from the router sending the update. The router being debugged also sent updates, in both cases to broadcast address 255.255.255.255 as the destination.

The first line in Figure 2-120 is self-explanatory.

The second line in Figure 2-120 is an example of a routing table update. It shows how many hops a given Internet address is from the router.

The entries in Figure 2-120 show that the router is sending updates that are similar, except that the number in parentheses is the source address encapsulated into the IP header.

Examples of additional output that the **debug ip rip** command can generate follow.

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

```

RIP: broadcasting general request on Ethernet0
RIP: broadcasting general request on Ethernet1

```

The following line is self-explanatory:

```

RIP: received request from 160.89.80.207 on Ethernet0

```

An entry such as the following is most likely caused by a malformed packet from the transmitter:

```

RIP: bad version 128 from 160.89.80.43

```

debug ip routing

Use the **debug ip routing EXEC** command to display information on Routing Information Protocol (RIP) routing table updates and route-cache updates. The **no** form of this command disables debugging output.

[no] debug ip routing

Sample Display

Figure 2-121 shows sample **debug ip routing** output.

Figure 2-121 Sample Debug IP Routing Output

```
Router# debug ip routing

RT: add 172.25.168.0 255.255.255.0 via 172.24.76.30, igrp metric [100/3020]
RT: metric change to 172.25.168.0 via 172.24.76.30, igrp metric [100/3020]
    new metric [100/2930]
IP: cache invalidation from 0x115248 0x1378A, new version 5736
RT: add 172.26.219.0 255.255.255.0 via 172.24.76.30, igrp metric [100/16200]
RT: metric change to 172.26.219.0 via 172.24.76.30, igrp metric [100/16200]
    new metric [100/10816]
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5737
RT: 172.26.219.0 came out of holddown
RT: garbage collecting entry for 172.26.219.0
IP: cache invalidation from 0x115248 0x1378A, new version 5738
RT: add 172.26.219.0 255.255.255.0 via 172.24.76.30, igrp metric [100/10816]
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5739
RT: 172.26.219.0 came out of holddown
RT: garbage collecting entry for 172.26.219.0
IP: cache invalidation from 0x115248 0x1378A, new version 5740
RT: add 172.26.219.0 255.255.255.0 via 172.24.76.30, igrp metric [100/16200]
RT: metric change to 172.26.219.0 via 172.24.76.30, igrp metric [100/16200]
    new metric [100/10816]
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5741
```

Explanations for representative lines of output in Figure 2-121 follow.

In the following lines, a newly created entry has been added to the IP routing table. The “metric change” indicates that this entry existed previously, but its metric changed and the change was reported by means of IGRP. The metric could also be reported via RIP, OSPF, or another IP routing protocol. The numbers inside the brackets report the administrative distance and the actual metric.

```
RT: add 172.25.168.0 255.255.255.0 via 172.24.76.30, igrp metric [100/3020]
RT: metric change to 172.25.168.0 via 172.24.76.30, igrp metric [100/3020]
    new metric [100/2930]
IP: cache invalidation from 0x115248 0x1378A, new version 5736
```

“Cache invalidation” means that the fast switching cache was invalidated due to a routing table change. “New version” is the version number of the routing table. When the routing table changes, this number is incremented. The hexadecimal numbers are internal numbers that vary from version to version and software load to software load.

In the following output, the “holddown” and “cache invalidation” lines are displayed. Most of the distance vector routing protocols use “holddown” to avoid typical problems like counting to infinity and routing loops. If you look at the output of **show ip protocols** you will see what the timer values are for “holddown” and “cache invalidation.” “Cache invalidation” corresponds to “came out of holddown.” “Delete route” is triggered when a better path comes along. It gets rid of the old inferior path.

```
RT: delete route to 172.26.219.0 via 172.24.76.30, igmp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5737
RT: 172.26.219.0 came out of holddown
```

debug ip rsvp

Use the **debug ip rsvp** EXEC command to enable logging of significant Resource Reservation Protocol (RSVP) events. The **no** form of this command disables debugging output.

[no] debug ip rsvp [detail [access-list]]

Syntax Description

detail	(Optional) Displays debug information in a detailed format.
<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.

Usage Guidelines

The RSVP protocol permits end systems to request quality of service (QoS) guarantees from the network.

The **debug ip rsvp** command displays recognition of new senders, subsequent discontinuation of senders, and installation and removal of reservations.

The **detail** option displays recognition of new senders, subsequent discontinuation of senders, installation and removal of reservations, and messages sent and received that are permitted by the specified access list, or all messages if no access list is specified. The output from **detail** option is the same as the **debug ip rsvp** command but it also includes a hexadecimal-dump of the contents of the messages.

Sample Displays

In the following message, the router received a path message for destination 192.168.253.10 on interface Ethernet 0/2. The message was sent by 172.31.215.9—this indicates the previous hop and not the source.

```
RSVP: PATH message for 192.168.253.10(Ethernet0/2) from 172.31.215.9
```

In the following two messages, the router is sending a path message to destination 192.168.253.10 on interface Ethernet 0/0. The second message is a reassurance that the packet is sent.

```
RSVP: send path multicast about 192.168.253.10 on Ethernet0/0  
RSVP: IP to 192.168.253.10 length=52 checksum=720C Ethernet0/0
```

In the following message, the router received an reservation (RESV) message for destination 192.168.253.10 on interface serial 0. The message was sent by 172.31.215.113, which is the next hop toward the destination.

```
RSVP: RESV message for 192.168.253.10(Serial0) from 172.31.215.113
```

In the following messages, the router is removing the sender state for destination 1.1.1.1. The destination port is (1234), the source is 172.31.215.11, and the protocol and source port are (17:1234). Removing the sender state triggers a teardown message to be sent toward destination 1.1.1.1 on interface Ethernet 0/2 (located in the routing table).

```
RSVP: remove Ethernet0/1 PATH 1.1.1.1(1234) <- 172.31.215.11(17:1234)  
RSVP: send path teardown multicast about 1.1.1.1 on Ethernet0/2
```

The following messages occur when a reservation (RESV) message is received but no sender state (path message) exists that matches it. In this case, the router drops the reservation and sends a reservation error message to destination 192.168.253.10.

```
RSVP: RESV message for 192.168.253.10(Serial0) from 172.31.215.113
RSVP: send reservation error to 172.31.215.113 about 192.168.253.10
RSVP: IP to 172.31.215.113 length=40 checksum=1A84 if Serial0
RSVP RESV: no path information for 192.168.253.10
```

In the following messages, the router tears down a reservation because it received a teardown message or because the message timed out. Tearing down a reservation triggers the removal of the existing installed reservation, as shown in the second message. The teardown also triggers the removal of this reservation from upstream routers. After the router removes the reservation, an RESV teardown message is sent to the source to tear down the reservation on the next hop, as shown in the fourth message. This process continues until the source is reached.

```
RSVP: RESV TEAR message for 192.168.253.10(Ethernet0/0) from 172.31.215.98
RSVP: remove Ethernet0/0 RESV 192.168.253.10(18004) <- 172.31.60.189(17:18004)
RSVP: remove Ethernet0/2 RESV request 192.168.253.10(18004) <- 172.31.60.189(17:18004)
RSVP: send reservation teardown to 172.31.215.9 about 192.168.253.10
```

In the following messages, the router received and accepted a new reservation request. This request triggers the router to send a reservation request message to the source.

```
RSVP: Reservation is new
RSVP: start requesting 30 kbps SE reservation for 172.31.60.189(18004) UDP->
192.168.253.10(18004) on Ethernet0 neighbor 172.31.215.97
```

In the following message, the router received an RSVP message on an interface with the RSVP feature disabled:

```
RSVP: Input packet while RSVP disabled on Serial2/0
```

debug ip rtp header-compression

Use the **debug ip rtp header-compression** EXEC command to display events specific to RTP header compression. Use the **no** form of this command to disable debugging output.

[no] debug ip rtp header-compression

Sample Display

Figure 2-122 shows sample **debug ip rtp header-compression** output.

Figure 2-122 Sample Debug IP RTP Header-Compression Output

```
Router# debug ip rtp header-compression

RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 0, received sequence 0
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 1, received sequence 1
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 2, received sequence 2
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 3, received sequence 3
```

Table 2-64 describes the significant fields in Figure 2-122.

Table 2-64 Debug IP RTP Header-Compression Field Descriptions

Field	Description
context 0	Compression state for a connection 0.
expected sequence	RTP header compression link sequence (expected).
received sequence	RTP header compression link sequence (actually received).

Related Command

debug ip rtp packets

debug ip rtp packets

Use the **debug ip rtp packets** EXEC command to display a detailed dump of packets specific to RTP header compression. Use the **no** form of this command to disable debugging output.

[no] debug ip rtp packets

Sample Display

Figure 2-123 shows sample **debug ip rtp packets** output.

Figure 2-123 Sample Debug IP RTP Packets Output

```
Router# debug ip rtp packets

RTP packet dump:
  IP:  source: 171.68.8.10, destination: 224.2.197.169, id: 0x249B, ttl: 9,
      TOS: 0 prot: 17,
  UDP: source port: 1034, destination port: 27404, checksum: 0xB429, len: 152
  RTP: version: 2, padding: 0, extension: 0, marker: 0,
      payload: 3, ssrc 2369713968,
      sequence: 2468, timestamp: 85187180, csrc count: 0
```

Table 2-65 describes the significant fields in Figure 2-123.

Table 2-65 Debug IP RTP Packets Field Descriptions

Field	Description
id	IP identification.
ttl	IP time to live (TTL).
len	Total UDP length.

Related Command

debug ip rtp header-compression

debug ip sd

Use the **debug ip sd** command to display all session directory (SD) announcements received. The **no** form of this command disables debugging output.

[no] debug ip sd

Usage Guidelines

This command shows session directory announcements for multicast IP. Use it to observe multicast activity.

Sample Display

Figure 2-124 shows sample **debug ip sd** output.

Figure 2-124 Sample Debug IP SD Output

```
Router# debug ip sd

SD: Announcement from 172.31.58.81 on Serial0.1, 146 bytes
s=*cisco: CBONE Audio
i=cisco internal-only audio conference
o=dino@dino-ss20.cisco.com
c=224.0.255.1 16 2891478496 2892688096
m=audio 31372 1700

SD: Announcement from 172.22.246.68 on Serial0.1, 147 bytes
s=IMS: U.S. Senate
i=U.S. Senate at http://town.hall.org/radio/live.html
o=carl@also.radio.com
c=224.2.252.231 95 0 0
m=audio 36572 2642
a=fmt:gsm
```

Table 2-66 provides explanations for representative lines of the **debug ip sd** output shown in Figure 2-124.

Table 2-66 Debug IP SD Output Descriptions

Field	Description
SD	Session directory event.
Announcement from	Address sending the SD announcement.
on Serial0.1	Interface receiving the announcement.
146 bytes	Size of the announcement event.
s=	Session name being advertised.
i=	Information providing a descriptive name for the session.
o=	Origin of the session, either an IP address or a name.
c=	Connect description showing address and number of hops.
m=	Media description that includes media type, port number, and ID.

Related Commands

debug ip dvmrp
debug ip igmp
debug ip mcache
debug ip mrouting
debug ip pim

debug ip security

Use the **debug ip security** EXEC command to display IP security option processing. The **no** form of this command disables debugging output.

[no] debug ip security

Usage Guidelines

The **debug ip security** command displays information for both basic and extended IP security options. For interfaces where **ip security** is configured, each IP packet processed for that interface results in debugging output regardless of whether the packet contains IP security options. IP packets processed for other interfaces that also contain IP security information also trigger debugging output. Some additional IP security debugging information is also controlled by the **debug ip packet** EXEC command.

Note Because the **debug ip security** command generates a significant amount of output for every IP packet processed, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Sample Display

Figure 2-125 shows sample **debug ip security** output.

Figure 2-125 Sample Debug IP Security Output

```
Router# debug ip security

IP Security: src 172.24.72.52 dst 172.24.72.53, number of BSO 1
      idb: NULL
      pak: insert (0xFF) 0x0
IP Security: BSO postroute: SECINSERT changed to secret (0x5A) 0x10
IP Security: src 172.24.72.53 dst 172.24.72.52, number of BSO 1
      idb: secret (0x6) 0x10 to secret (0x6) 0x10, no implicit
      def secret (0x6) 0x10
      pak: secret (0x5A) 0x10
IP Security: checking BSO 0x10 against [0x10 0x10]
IP Security: classified BSO as secret (0x5A) 0x10
```

Table 2-67 describes significant fields shown in Figure 2-125.

Table 2-67 Debug IP Security Field Descriptions

Field	Description
number of BSO	Indicates the number of basic security options found in the packet.
idb	Provides information on the security configuration for the incoming interface.
pak	Provides information on the security classification of the incoming packet.
src	Indicates the source IP address.
dst	Indicates the destination IP address.

Explanations for representative lines of output in Figure 2-125 follow.

The following line indicates that the packet was locally generated, and it has been classified with the internally significant security level “insert” (0xff) and authority 0x0:

```
idb: NULL
pak: insert (0xff) 0x0
```

The following line indicates that the packet was received via an interface with dedicated IP security configured. Specifically, the interface is configured at security level “secret” and with authority information of 0x0. The packet itself was classified at level “secret” (0x5a) and authority 0x10.

```
idb: secret (0x6) 0x10 to secret (0x6) 0x10, no implicit
     def secret (0x6) 0x10
pak: secret (0x5A) 0x10
```

debug ip socket

Use the **debug ip socket EXEC** command to display all state change information for all sockets. Use the **no** form of this command to disable debugging output.

[no] debug ip socket

Usage Guidelines

Use this command to collect information on the socket interface. To get more complete information on a socket/TCP port pair, use this command in conjunction with **debug ip tcp transactions**.

Because the socket debugging information is state-change oriented, you will not see the debug message on a per packet basis. However, if the connections normally have very short lives (few packet exchanges during the life cycle of a connection), then socket debugging could become expensive because of the state changes involved during connection setup and teardown.

Sample Displays

Figure 2-126 shows sample **debug ip socket** output from a server process.

Figure 2-126 Sample Debug IP Socket Output—Server Process

```
Router# debug ip socket

Added socket 0x60B86228 to process 40
SOCKET: set TCP property TCP_PID, socket 0x60B86228, TCB 0x60B85E38
Accepted new socket fd 1, TCB 0x60B85E38
Added socket 0x60B86798 to process 40
SOCKET: set TCP property TCP_PID, socket 0x60B86798, TCB 0x60B877C0
SOCKET: set TCP property TCP_BIT_NOTIFY, socket 0x60B86798, TCB 0x60B877C0
SOCKET: created new socket to TCP, fd 2, TCB 0x60B877C0
SOCKET: bound socket fd 2 to TCB 0x60B877C0
SOCKET: set TCP property TCP_WINDOW_SIZE, socket 0x60B86798, TCB 0x60B877C0
SOCKET: listen on socket fd 2, TCB 0x60B877C0
SOCKET: closing socket 0x60B86228, TCB 0x60B85E38
SOCKET: socket event process: socket 0x60B86228, TCB new state --> FINWAIT1
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING
SOCKET: Removed socket 0x60B86228 from process 40 socket list
```

Figure 2-127 shows sample **debug ip socket** output from a client process.

Figure 2-127 Sample Debug IP Socket Output—Client Process

```
Router# debug ip socket

Added socket 0x60B70220 to process 2
SOCKET: set TCP property TCP_PID, socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: set TCP property TCP_BIT_NOTIFY, socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: created new socket to TCP, fd 0, TCB 0x60B6CFDC
SOCKET: socket event process: socket 0x60B70220, TCB new state --> SYNSENT
socket state: SS_ISCONNECTING
SOCKET: socket event process: socket 0x60B70220, TCB new state --> ESTAB
socket state: SS_ISCONNECTING
SOCKET: closing socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: socket event process: socket 0x60B70220, TCB new state --> FINWAIT1
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING
SOCKET: Removed socket 0x60B70220 from process 2 socket list
```

Table 2-68 describes the significant fields in Figure 2-126 and Figure 2-127.

Table 2-68 Debug IP Socket Field Descriptions

Field	Description
Added socket 0x60B86228 process 40	New socket is opened for process 40.
SOCKET	Indicates that this is a SOCKET transaction.
set TCP property TCP_PID	Set process ID to the TCP associated with the socket.
socket 0x60B86228, TCB 0x60B85E38	Address for the socket/TCP pair.
set TCP property TCP_BIT_NOTIFY	Set the method for how the socket wants to be notified for an event.
created new socket to TCP, fd 2	Opened a new socket referenced by file descriptor 2 to TCP.
bound socket fd 2 to TCB	Bound the socket referenced by file descriptor 2 to TCP.
listen on socket fd 2	Indicates which file descriptor the application is listening to.
closing socket	Indicates the socket is being closed.
socket event process	Processed a state change event occurred in the transport layer.
TCB new state --> FINWAIT1	TCP state machine changed to FINWAIT1. (See the debug ip tcp transaction command for more information on TCP state machines.)
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING	<p>New SOCKET state flags after the transport event processing. This socket is still connected, but disconnecting is in progress, and it will not send more data to peer.</p> <p>Possible SOCKET state flags follow:</p> <p>SS_NOFDREF No file descriptor reference for this socket.</p> <p>SS_ISCONNECTING Socket connecting is in progress.</p> <p>SS_ISBOUND Socket is bound to TCP.</p> <p>SS_ISCONNECTED Socket is connected to peer.</p> <p>SS_ISDISCONNECTING Socket disconnecting is in progress.</p> <p>SS_CANTSENDMORE Can't send more data to peer.</p> <p>SS_CANTRCVMORE Can't receive more data from peer.</p> <p>SS_ISDISCONNECTED Socket is disconnected. Connection is fully closed.</p>
Removed socket 0x60B86228 from process 40 socket list	Connection is closed, and the socket is removed from the process socket list.

Related Command

debug ip tcp transactions

debug ip tcp driver

Use the **debug ip tcp driver EXEC** command to display information on Transmission Control Protocol (TCP) driver events; for example, connections opening or closing, or packets being dropped because of full queues. The **no** form of this command disables debugging output.

[no] debug ip tcp driver

Usage Guidelines

The TCP driver is the process that the router software uses to send packet data over a TCP connection. Remote source-route bridging, STUN (serial tunneling), and X.25 switching currently use the TCP driver.

Using the **debug ip tcp driver** command together with the **debug ip tcp driver-pak** command provides the most verbose debugging output concerning TCP driver activity.

Sample Display

Figure 2-128 shows sample **debug ip tcp driver** output.

Figure 2-128 Sample Debug IP TCP Driver Output

```
Router# debug ip tcp driver

TCPDRV359CD8: Active open 172.21.80.26:0 --> 172.21.80.25:1996 OK, lport 36628
TCPDRV359CD8: enable tcp timeouts
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 Abort
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 DoClose tcp abort
```

Explanations for individual lines of output from Figure 2-128 follow.

Table 2-69 describes the fields in the first line of output in Figure 2-128.

Table 2-69 Debug IP TCP Driver Field Descriptions

Field	Description
TCPDRV359CD8:	Unique identifier for this instance of TCP driver activity.
Active open 172.21.80.26	Indication that the router at IP address 172.21.80.26 has initiated a connection to another router.
:0	The TCP port number the initiator of the connection uses to indicate that any port number can be used to set up a connection.
--> 172.21.80.25	The IP address of the remote router to which the connection has been initiated.
:1996	The TCP port number that the initiator of the connection is requesting that the remote router use for the connection. (1996 is a private TCP port number reserved in this implementation for remote source-route bridging.)
OK,	Indication that the connection has been established. If the connection has not been established, this field and the following field do not appear in this line of output.
lport 36628	The TCP port number that has actually been assigned for the initiator to use for this connection.

The following line indicates that the TCP driver user (remote source-route bridging, in this case) will allow TCP to drop the connection if excessive retransmissions occur:

```
TCPDRV359CD8: enable tcp timeouts
```

The following line indicates that the TCP driver user (in this case, remote source-route bridging) at IP address 172.21.80.26 (and using TCP port number 36628) is requesting that the connection to IP address 172.21.80.25 using TCP port number 1996 be aborted:

```
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 Abort
```

The following line indicates that this connection was in fact closed due to an abnormal termination:

```
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 DoClose tcp abort
```

debug ip tcp driver-pak

Use the **debug ip tcp driver-pak** EXEC command to display information on every operation that the Transmission Control Protocol (TCP) driver performs. The **no** form of this command disables debugging output.

[no] debug ip tcp driver-pak

Usage Guidelines

This command turns on a verbose debugging by logging at least one debugging message for every packet sent or received on the TCP driver connection.

The TCP driver is the process that the router software uses to send packet data over a TCP connection. Remote source-route bridging, STUN (serial tunneling), and X.25 switching currently use the TCP driver.

To observe the context within which certain **debug ip tcp driver-pak** messages occur, turn on this command in conjunction with the **debug ip tcp driver** command.

Note Because the **debug ip tcp driver-pak** command generates so many messages, use it only on lightly loaded systems. This command not only places a significant load on the system processor, but it may even change the symptoms of any unexpected behavior that occur.

Sample Display

Figure 2-129 shows sample **debug ip tcp driver-pak** output.

Figure 2-129 Sample Debug IP TCP Driver-Pak Output

```
Router# debug ip tcp driver-pak

TCPDRV359CD8: send 2E8CD8 (len 26) queued
TCPDRV359CD8: output pak 2E8CD8 (len 26) (26)
TCPDRV359CD8: readf 42 bytes (Thresh 16)
TCPDRV359CD8: readf 26 bytes (Thresh 16)
TCPDRV359CD8: readf 10 bytes (Thresh 10)
TCPDRV359CD8: send 327E40 (len 4502) queued
TCPDRV359CD8: output pak 327E40 (len 4502) (4502)
```

Explanations for individual lines of output from Figure 2-129 follow.

Table 2-70 describes the fields shown in the first line of output of Figure 2-129.

Table 2-70 Debug TCP Driver-Pak Field Descriptions

Field	Description
TCPDRV359CD8	Unique identifier for this instance of TCP driver activity.
send	Indication that this event involves the TCP driver sending data.
2E8CD8	Address in memory of the data the TCP driver is sending.
(len 26)	Length of the data (in bytes).
queued	Indication that the TCP driver user process (in this case, remote source-route bridging) has transferred the data to the TCP driver to send.

The following line indicates that the TCP driver has sent the data that it had received from the TCP driver user, as shown in the previous line of output. The last field in the line (26) indicates that the 26 bytes of data were sent out as a single unit.

```
TCPDRV359CD8: output pak 2E8CD8 (len 26) (26)
```

The following line indicates that the TCP driver has received 42 bytes of data from the remote IP address. The TCP driver user (in this case, remote source-route bridging) has established an input threshold of 16 bytes for this connection. (The input threshold instructs the TCP driver to transfer data to the TCP driver user only when at least 16 bytes are present.)

```
TCPDRV359CD8: readf 42 bytes (Thresh 16)
```

debug ip tcp intercept

To display TCP intercept statistics, use the **debug ip tcp intercept EXEC** command. The **no** form of this command disables debugging output.

[no] debug ip tcp intercept

Sample Display

Figure 2-130 illustrates a scenario in which a router configured with TCP intercept operates between a client and a server.

Figure 2-130 TCP Intercept Debug Scenario

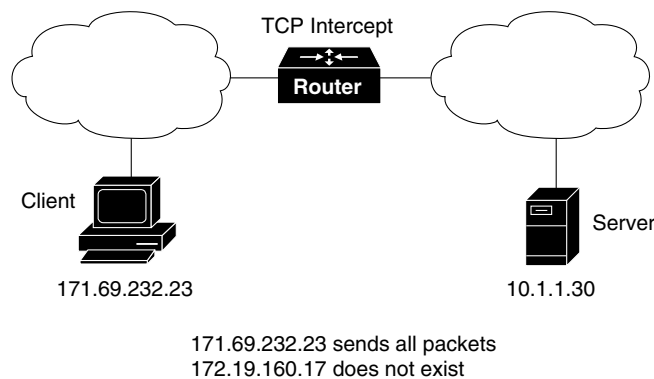


Figure 2-131 shows sample **debug ip tcp intercept** output, with interspersed commentary. The sample output is based on Figure 2-130.

Figure 2-131 Sample Debug IP TCP Intercept Output

```
Router# debug ip tcp intercept
```

A connection attempt arrives.

```
INTERCEPT: new connection (172.19.160.17:61774) => (10.1.1.30:23)
INTERCEPT: 172.19.160.17:61774 <- ACK+SYN (10.1.1.30:61774)
```

A second connection attempt arrives.

```
INTERCEPT: new connection (172.19.160.17:62030) => (10.1.1.30:23)
INTERCEPT: 172.19.160.17:62030 <- ACK+SYN (10.1.1.30:62030)
```

The router retransmits to both apparent clients.

```
INTERCEPT: retransmit 2 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 2 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
```

A third connection attempt arrives.

```
INTERCEPT: new connection (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: 171.69.232.23:1048 <- ACK+SYN (10.1.1.30:1048)
```

The router sends more retransmissions trying to establish connections with the apparent clients.

```
INTERCEPT: retransmit 4 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 4 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 2 (171.69.232.23:1048) <- (10.1.1.30:23) SYNRCVD
```

The router establishes the connection with the third client and retransmits to the server.

```
INTERCEPT: 1st half of connection is established (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: (171.69.232.23:1048) SYN -> 10.1.1.30:23
INTERCEPT: retransmit 2 (171.69.232.23:1048) -> (10.1.1.30:23) SYNSENT
```

The server responds; the connection is established.

```
INTERCEPT: 2nd half of connection established (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: (171.69.232.23:1048) ACK -> 10.1.1.30:23
```

The router retransmits to the first two apparent clients, times out, and sends resets.

```
INTERCEPT: retransmit 8 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 8 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 16 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 16 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmitting too long (172.19.160.17:61774) => (10.1.1.30:23) SYNRCVD
INTERCEPT: 172.19.160.17:61774 <- RST (10.1.1.30:23)
INTERCEPT: retransmitting too long (172.19.160.17:62030) => (10.1.1.30:23) SYNRCVD
INTERCEPT: 172.19.160.17:62030 <- RST (10.1.1.30:23)
```

debug ip tcp transactions

Use the **debug ip tcp transactions** EXEC command to display information on significant Transmission Control Protocol (TCP) transactions such as state changes, retransmissions, and duplicate packets. The **no** form of this command disables debugging output.

[no] debug ip tcp transactions

Usage Guidelines

This command is particularly useful for debugging a performance problem on a TCP/IP network that you have isolated above the data link layer.

The **debug ip tcp transactions** command displays output for packets the router sends and receives, but does not display output for packets it forwards.

Sample Display

Figure 2-132 shows sample **debug ip tcp transactions** output.

Figure 2-132 Sample Debug IP TCP Output

```
Router# debug ip tcp transactions

TCP: sending SYN, seq 168108, ack 88655553
TCP0: Connection to 10.9.0.13:22530, advertising MSS 966
TCP0: state was LISTEN -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: state was SYNSENT -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: Connection to 10.9.0.13:22530, received MSS 956
TCP0: restart retransmission in 5996
TCP0: state was SYNRCVD -> ESTAB [23 -> 10.9.0.13(22530)]
TCP2: restart retransmission in 10689
TCP2: restart retransmission in 10641
TCP2: restart retransmission in 10633
TCP2: restart retransmission in 13384 -> 10.0.0.13(16151)]
TCP0: restart retransmission in 5996 [23 -> 10.0.0.13(16151)]
```

Table 2-71 describes significant fields shown in Figure 2-132.

Table 2-71 Debug IP TCP Transactions Field Descriptions

Field	Description
TCP:	Indicates that this is a TCP transaction.
sending SYN	Indicates that a synchronize packet is being sent.
seq 168108	Indicates the sequence number of the data being sent.
ack 88655553	Indicates the sequence number of the data being acknowledged.
TCP0:	Indicates the TTY number (0, in this case) with which this TCP connection is associated.
Connection to 10.9.0.13:22530	Indicates the remote address with which a connection has been established.
advertising MSS 966	Indicates the maximum segment size this side of the TCP connection is offering to the other side.

Table 2-71 Debug IP TCP Transactions Field Descriptions (Continued)

Field	Description
state was LISTEN -> SYNRCVD	<p>Indicates that the TCP state machine changed state from LISTEN to SYNSENT. Possible TCP states follow:</p> <p>CLOSED—Connection closed.</p> <p>CLOSEWAIT—Received a FIN segment.</p> <p>CLOSING—Received a FIN/ACK segment.</p> <p>ESTAB—Connection established.</p> <p>FINWAIT 1—Sent a FIN segment to start closing the connection.</p> <p>FINWAIT 2—Waiting for a FIN segment.</p> <p>LASTACK—Sent a FIN segment in response to a received FIN segment.</p> <p>LISTEN—Listening for a connection request.</p> <p>SYNRCVD—Received a SYN segment, and responded.</p> <p>SYNSENT—Sent a SYN segment to start connection negotiation.</p> <p>TIMEWAIT—Waiting for network to clear segments for this connection before the network no longer recognizes the connection as valid. This must occur before a new connection can be set up.</p>
[23 -> 10.9.0.13(22530)]	<p>Within these brackets:</p> <p>The first field (23) indicates local TCP port.</p> <p>The second field (10.9.0.13) indicates the destination IP address.</p> <p>The third field (22530) indicates the destination TCP port.</p>
restart retransmission in 5996	<p>Indicates the number of milliseconds until the next retransmission takes place.</p>