debug ncia circuit through debug pxf tbridge

debug nat64

To enable stateless Network Address Translation 64 (NAT64) debugging, use the **debug nat64** command in privileged EXEC mode. To disable NAT64 debugging, use the **no** form of this command.

debug nat64 {**all**| {**aliases**| **ha** {**all**| **info**| **trace**| **warn**}}| **id-manager**| **info**| **intf-address**| **issu** {**all**| **message**| **trace**}| **memory**| **pool-routes**| **statistics**| **trace**| **warn**}

no debug nat64 {**all**| {**aliases**| **ha** {**all**| **info**| **trace**| **warn**}}| **id-manager**| **info**| **intf-address**| **issu** {**all**| **message**| **trace**}| **memory**| **pool-routes**| **statistics**| **trace**| **warn**}

Syntax Description

all	Enables information, trace, and warning level debugging.
aliases	Enables debugging of IP aliases created by NAT64.
ha	Enables high availability (HA) debugging.
all	Enables HA information, trace, and warning level debugging.
info	Enables HA information level debugging.
trace	Enables HA trace level debugging.
warn	Enables HA warning level debugging.
id-manager	Enables Interface Descriptor manager trace debugging.
info	Enables information level debugging.
intf-address	Enables interface address change events debugging.
issu	Enables In-Service Software Upgrade (ISSU) debugging.
all	Enables ISSU trace level and message debugging.
message	Enables ISSU message debugging.
trace	Enables ISSU trace level debugging.
memory	Enables memory trace debugging.
pool-routes	Enables the debugging of routes attached to a a pool address range.

statistics	Enables statistics debugging.
trace	Enables trace level debugging.
warn	Enables warning level debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.2S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was modified. The **aliases**, **intf-address**, and **pool-routes** keywords were added.

Usage Guidelines

The general debugging levels are information, trace, and warning. The **debug nat64 memory** and **debug nat64 id-manager** commands provide detailed traces related to resources and memory allocation. The **debug nat64 issu** command provides traces specific to the ISSU messages exchanged.

Examples

The following is sample output from the **debug nat64 statistics** command. The output fields are self-explanatory.

```
Router# debug nat64 statistics

NAT64 statistics debugging is on
Sep 16 18:26:24.537 IST: NAT64 (stats): Received stats update for IDB(FastEthernet0/3/5)
Sep 16 18:26:24.537 IST: NAT64 (stats): Updating pkts_translated_v4v6 from 94368894 to
95856998 (is_delta(TRUE) value(1488104))
Sep 16 18:26:24.537 IST: NAT64 (stats): Received stats update for IDB(FastEthernet0/3/4)
Sep 16 18:26:24.537 IST: NAT64 (stats): Updating pkts_translated_v6v4 from 7771538 to 7894088
 (is_delta(TRUE) value(122550))
Sep 16 18:26:24.537 IST: NAT64 (stats): Received global stats update
Sep 16 18:26:24.537 IST: NAT64 (stats): Updating pkts_translated_v4v6 from 1718650332 to
1720138437 (is_delta(TRUE) value(1488105))
Sep 16 18:26:24.537 IST: NAT64 (stats): Updating pkts_translated_v6v4 from 1604459283 to
1604581833 (is_delta(TRUE) value(122550))
```

The following is sample output from the **debug nat64 memory** command. The output fields are self-explanatory.

```
Router# debug nat64 memory

NAT64 memory debugging is on
Sep 16 18:28:03.713 IST: NAT64 (memory): Allocated 0x7FFA7DA2F750
Sep 16 18:28:03.713 IST: NAT64 (memory): Allocated 0x7FFA9EC00D30
Sep 16 18:28:03.713 IST: NAT64 (memory): Allocated 0x7FFA9D1532C8
```

Related Commands

Command	Description
nat64 enable	Enables stateless NAT64 on an interface.

debug ncia circuit

To display circuit-related information between the native client interface architecture (NCIA) server and client, use the **debugnciacircuit**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ncia circuit [**error**| **event**| **flow-control**| **state**]

no debug ncia circuit [**error**| **event**| **flow-control**| **state**]

Syntax Description

error	(Optional) Displays the error situation for each circuit.
event	(Optional) Displays the packets received and sent for each circuit.
flow-control	(Optional) Displays the flow control information for each circuit.
state	(Optional) Displays the state changes for each circuit.

Command Modes

Privileged EXEC

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing Systems Network Architecture (SNA) applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

You cannot enable debugging output for a particular client or particular circuit.

Caution Do not enable the **debugnciacircuit** command during normal operation because this command generates a substantial amount of output messages and could slow down the router.

Examples

The following is sample output from the**debugnciacircuiterror** command. In this example, the possible errors are displayed. The first error message indicates that the router is out of memory. The second message indicates that the router has an invalid circuit control block. The third message indicates that the router is out of memory. The remaining messages identify errors related to the finite state machine.

```
Router# debug ncia circuit error
NCIA: ncia_circuit_create memory allocation fail
NCIA: ncia_send_ndlc: invalid circuit control block
NCIA: send_ndlc: fail to get buffer for ndlc primitive xxx
NCIA: ncia circuit fsm: Invalid input
NCIA: ncia circuit fsm: Illegal state
NCIA: ncia circuit fsm: Illegal input
NCIA: ncia circuit fsm: Unexpected input
NCIA: ncia circuit fsm: Unknown error rtn code
```

The following is sample output from the **debugnciacircuitevent**command. In this example, a session startup sequence is displayed.

```
Router# debug ncia circuit event
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_START_DL, Len: 24, tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 0, lfs 16, ws 1
NCIA: create circuit: saddr 4000.1060.1000, ssap 4, daddr 4000.3000.0003, dsap 8 sid:
          8B09A8
NCIA: send NDLC_DL_STARTED to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_DL_STARTED, Len: 2,4 tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 8B09A8, lfs 16, ws 1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8B09A8, FC 0x81
NCIA: send NDLC_XID_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 18, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_CONTACT_STN to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_CONTACT_STN, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_STN_CONTACTED, Len: 12, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_INFO_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_INFO_FRAME, Len: 30, sid: 8A91E8, FC 0xC1
```

The following table describes the significant fields shown in the display.

Table 1: debug ncia circuit event Field Descriptions

Field	Description
IN	Incoming message from client.
OUT	Outgoing message to client.
Ver_Id	NDLC version ID.
MsgType	NDLC message type.
Len	NDLC message length.
tmac	Target MAC.
tsap	Target SAP.
csap	Client SAP.
oid	Origin ID.
tid	Target ID.
lfs	Largest frame size flag.
ws	Window size.
saddr	Source MAC address.
ssap	Source SAP.
daddr	Destination MAC address.

Field	Description
dsap	Destination SAP.
sid	Session ID.
FC	Flow control flag.

In the following messages, an NDLC_START_DL messages is received from a client to start a data-link session:

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_START_DL, Len: 24, tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 0, lfs 16, ws 1
NCIA: create circuit: saddr 4000.1060.1000, ssap 4, daddr 4000.3000.0003, dsap 8 sid:
          8B09A8
```

The next two messages indicate that an NDLC_DL_STARTED message is sent to a client. The server informs the client that a data-the link session is started.

```
NCIA: send NDLC_DL_STARTED to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_DL_STARTED, Len: 2,4 tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 8B09A8, lfs 16, ws 1
```

In the following two messages, an NDLC_XID_FRAME message is received from a client, and the client starts an XID exchange:

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8B09A8, FC 0x81
NCIA: send NDLC_XID_FRAME to client 10.2.20.3 for ckt: 8B09A8
```

In the following two messages, an NDLC_XID_FRAME message is sent from a client, and an DLC_XID_FRAME message is received from a client:

```
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 18, sid: 8B09A8, FC 0xC1
```

The next two messages show that an NDLC_CONTACT_STN message is sent to a client:

```
NCIA: send NDLC_CONTACT_STN to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_CONTACT_STN, Len: 12, sid: 8A91E8, FC 0xC1
```

In the following message, an NDLC_STN_CONTACTED message is received from a client. The client informs the server that the station has been contacted.

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_STN_CONTACTED, Len: 12, sid: 8B09A8, FC 0xC1
```

In the last two messages, an NDLC_INFO_FRAME is sent to a client, and the server sends data to the client:

```
NCIA: send NDLC_INFO_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_INFO_FRAME, Len: 30, sid: 8A91E8, FC 0xC1
```

The following is sample output from the **debugnciacircuitflow-control** command. In this example, the flow control in a session startup sequence is displayed:

```
Router# debug ncia circuit flow-control
NCIA: no flow control in NDLC_DL_STARTED frame
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0x81, IW 1 GP 2 CW 2, Client IW 1 GP 0 CW 1
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 2 CW 2, Client IW 1 GP 2 CW 2
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0xC1, IW 1 GP 5 CW 3, Client IW 1 GP 2 CW 2
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 5 CW 3, Client IW 1 GP 5 CW 3
```

```
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0xC1, IW 1 GP 9 CW 4, Client IW 1 GP 5 CW 3
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 8 CW 4, Client IW 1 GP 9 CW 4
NCIA: reduce ClientGrantPacket by 1 (Granted: 8)
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
```

The following table describes the significant fields shown in the display.

Table 2: debug ncia circuit flow-control Field Descriptions

Field	Description
IW	Initial window size.
GP	Granted packet number.
CW	Current window size.

The following is sample output from the **debugnciacircuitstate**command. In this example, a session startup sequence is displayed:

```
Router# debug ncia circuit state
NCIA: pre-server fsm: event CONN_OPENED
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - STDL state: CLSOED
NCIA: ncia server fsm action 32
NCIA: circuit state: CLOSED -> START_DL_RCVD
NCIA: server event: DLU - TestStn.Rsp state: START_DL_RCVD
NCIA: ncia server fsm action 17
NCIA: circuit state: START_DL_RCVD -> DL_STARTED_SND
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - XID state: DL_STARTED_SND
NCIA: ncia server fsm action 33
NCIA: circuit state: DL_STARTED_SND -> DL_STARTED_SND
NCIA: server event: DLU - ReqOpnStn.Req state: DL_STARTED_SND
NCIA: ncia server fsm action 33
NCIA: circuit state: DL_STARTED_SND -> OPENED
NCIA: server event: DLU - Id.Rsp state: OPENED
NCIA: ncia server fsm action 11
NCIA: circuit state: OPENED -> OPENED
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - XID state: OPENED
NCIA: ncia server fsm action 33
NCIA: circuit state: OPENED -> OPENED
NCIA: server event: DLU - Connect.Req state: OPENED
NCIA: ncia server fsm action 6
NCIA: circuit state: OPENED -> CONNECT_PENDING
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - CONR state: CONNECT_PENDING
NCIA: ncia server fsm action 33 --> CLS_CONNECT_CNF sets NciaClsBusy
NCIA: circuit state: CONNECT_PENDING -> CONNECTED
NCIA: server event: DLU - Flow.Req (START) state: CONNECTED
NCIA: ncia server fsm action 25 --> unset NciaClsBusy
NCIA: circuit state: CONNECTED -> CONNECTED
NCIA: server event: DLU - Data.Rsp state: CONNECTED
NCIA: ncia server fsm action 8
NCIA: circuit state: CONNECTED -> CONNECTED
```

The following table describes the significant fields shown in the display.

Table 3: debug ncia circuit state Field Descriptions

Field	Description
WAN	Event from WAN (client).
DLU	Event from upstream module--dependent logical unit (DLU).
ADMIN	Administrative event.
TIMER	Timer event.

Related Commands

Command	Description
debug dmsp fax-to-doc	Enables debugging of DLSw+.
debug ncia client	Displays debug information for all NCIA client processing that occurs in the router.
debug ncia server	Displays debug information for the NCIA server and its upstream software modules.

debug ncia client

To display debug information for all native client interface architecture (NCIA) client processing that occurs in the router, use the **debug ncia client**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ncia client [*ip-address*| **error** [*ip-address*]| **event** [*ip-address*]| **message** [*ip-address*]]

no debug ncia client [*ip-address*| **error** [*ip-address*]| **event** [*ip-address*]| **message** [*ip-address*]]

Syntax Description

ip-address	(Optional) The remote client IP address.
error	(Optional) Triggers the recording of messages only when errors occur. The current state and event of an NCIA client are normally included in the message. If you do not specify an IP address, the error messages are logged for all active clients.
event	(Optional) Triggers the recording of messages that describe the current state and event--and sometimes the action that just completed--for the NCIA client. If you do not specify an IP address, the messages are logged for all active clients.
message	(Optional) Triggers the recording of messages that contain up to the first 32 bytes of data in a TCP packet sent to or received from an NCIA client. If you do not specify an IP address, the messages are logged for all active clients.

Command Modes

Privileged EXEC

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing Systems Network Architecture (SNA) applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

Use the **debug ncia client error** command to see only certain error conditions that occur.

Use the **debug ncia client event** command to determine the sequences of activities that occur while an NCIA client is in different processing states.

Use the **debug ncia client message** command to see only the first 32 bytes of data in a TCP packet sent to or received from an NCIA client.

The **debug ncia client**command can be used in conjunction with the **debug ncia server** and **debug ncia circuit** commands to get a complete picture of NCIA activity.

Examples

The following is sample output from the **debug ncia client** command. Following the example is a description of each sample output message.

```
Router# debug ncia client
NCIA: Passive open 10.2.20.123(1088) -> 1973
NCIA: index for client hash queue is 27
NCIA: number of element in client hash queue 27 is 1
NCIA: event PASSIVE_OPEN, state NCIA_CLOSED for client 10.2.20.123
NCIA: Rcvd msg type NDLC_CAP_XCHG in tcp packet for client 10.2.20.123
NCIA: First 17 byte of data rcvd: 811200110000000000000400050104080C
NCIA: Sent msg type NDLC_CAP_XCHG in tcp packet to client 10.2.20.123
NCIA: First 17 byte of data sent: 811200111000000010000400050104080C
NCIA: event CAP_CMD_RCVD, state NCIA_CAP_WAIT, for client 10.2.20.123, cap xchg cmd sent
NCIA: Rcvd msg type NDLC_CAP_XCHG in tcp packet for client 10.2.20.123
NCIA: First 17 byte of data rcvd: 811200111000000010000000050104080C
NCIA: event CAP_RSP_RCVD, state NCIA_CAP_NEG for client 10.2.20.123
NCIA: Rcvd msg type NDLC_PEER_TEST_REQ in tcp packet for client 10.2.20.123
NCIA: First 4 byte of data rcvd: 811D0004
NCIA: event KEEPALIVE_RCVD, state NCIA_OPENED for client 10.2.20.123
NCIA: Sent msg type NDLC_PEER_TEST_RSP in tcp packet to client 10.2.20.123
NCIA: First 4 byte of data sent: 811E0004IA
NCIA: event TIME_OUT, state NCIA_OPENED, for client 10.2.20.123, keepalive_count = 0
NCIA: Sent msg type NDLC_PEER_TEST_REQ, in tcp packet to client 10.2.20.123
NCIA: First 4 byte of data sent: 811D0004
NCIA: Rcvd msg type NDLC_PEER_TEST_RSP in tcp packet for client 10.2.20.123
NCIA: First 4 byte of data rcvd: 811E0004
NCIA: event KEEPALIVE_RSP_RCVD, state NCIA_OPENED for client 10.2.20.123
NCIA: Error, event PASIVE_OPEN, state NCIA_OPENED, for client 10.2.20.123, should not have
 occurred.
NCIA: Error, active_open for pre_client_fsm while client 10.2.20.123 is active or not
configured, registered.
```

Messages in lines 1 through 12 show the events that occur when a client connects to the router (the NCIA server). These messages show a passive_open process.

Messages in lines 13 to 17 show the events that occur when a TIME_OUT event is detected by a client PC workstation. The workstation sends an NDLC_PEER_TEST_REQ message to the NCIA server, and the router responds with an NDLC_PEER_TEST_RSP message.

Messages in lines 18 to 23 show the events that occur when a TIME_OUT event is detected by the router (the NCIA server). The router sends an NDLC_PEER_TEST_REQ message to the client PC workstation, and the PC responds with an NDLC_PEER_TEST_RSP message.

When you use the **debug ncia client message** command, the messages shown on lines 6, 8, 11, 14, 17, 20, and 22 are output in addition to other messages not shown in this example.

When you use the **debug ncia client error** command, the messages shown on lines 24 and 25 are output in addition to other messages not shown in this example.

Related Commands

Command	Description
debug ncia circuit	Displays debug information for all NCIA client processing that occurs in the router.
debug ncia server	Displays debug information for the NCIA server and its upstream software modules.

debug ncia server

To display debug information for the native client interface architecture (NCIA) server and its upstream software modules, use the **debug ncia server**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ncia server

no debug ncia server

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing Systems Network Architecture (SNA) applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

The **debug ncia server** command displays all Cisco Link Services (CLS) messages between the NCIA server and its upstream modules, such as data-link switching (DLSw) and downstream physical units (DSPUs). Use this command when a problem exists between the NCIA server and other software modules within the router.

You cannot enable debugging output for a particular client or particular circuit.

Examples

The following is sample output from the **debug ncia server** command. In this example, a session startup sequence is displayed. Following the example is a description of each group of sample output messages.

```
Router# debug ncia server
NCIA: send CLS_TEST_STN_IND to DLU
NCIA: Receive TestStn.Rsp
NCIA: send CLS_ID_STN_IND to DLU
NCIA: Receive ReqOpnStn.Req
NCIA: send CLS_REQ_OPNSTN_CNF to DLU
NCIA: Receive Id.Rsp
NCIA: send CLS_ID_IND to DLU
NCIA: Receive Connect.Req
NCIA: send CLS_CONNECT_CNF to DLU
NCIA: Receive Flow.Req
NCIA: Receive Data.Req
NCIA: send CLS_DATA_IND to DLU
NCIA: send CLS_DISC_IND to DLU
NCIA: Receive Disconnect.Rsp
```

In the following messages, the client is sending a test message to the host and the test message is received by the host:

```
NCIA: send CLS_TEST_STN_IND to DLU
NCIA: Receive TestStn.Rsp
```

In the next message, the server is sending an exchange identification (XID) message to the host:

```
NCIA: send CLS_ID_STN_IND to DLU
```

In the next two messages, the host opens the station and the server responds:

```
NCIA: Receive ReqOpnStn.Req
NCIA: send CLS_REQ_OPNSTN_CNF to DLU
```

In the following two messages, the client is performing an XID exchange with the host:

```
NCIA: Receive Id.Rsp
NCIA: send CLS_ID_IND to DLU
```

In the next group of messages, the host attempts to establish a session with the client:

```
NCIA: Receive Connect.Req
NCIA: send CLS_CONNECT_CNF to DLU
NCIA: Receive Flow.Req
```

In the next two messages, the host sends data to the client:

```
NCIA: Receive Data.Req
NCIA: send CLS_DATA_IND to DLU
```

In the last two messages, the client closes the session:

```
NCIA: send CLS_DISC_IND to DLU
NCIA: Receive Disconnect.Rsp
```

Related Commands

Command	Description
debug dmsp fax-to-doc	Enables debugging of DLSw+.
debig mcoa circuit	Displays circuit-related information between the NCIA server and client.
debug ncia client	Displays debug information for all NCIA client processing that occurs in the router.

debug netbios error

To display information about Network Basic Input/Output System (NetBIOS) protocol errors, use the **debug netbios error**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug netbios error

no debug netbios error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

For complete information on the NetBIOS process, use the **debug netbios packet** command along with the **debug netbios error** command.

Examples

The following is sample output from the **debug netbios error**command. This example shows that an illegal packet has been received on the asynchronous interface.

```
Router# debug netbios error
Async1 nbf Bad packet
```

Related Commands

Command	Description
debug netbios-name-cache	Displays name caching activities on a router.
debug netbios packet	Displays general information about NetBIOS packets.

debug netbios packet

To display general information about Network Basic Input/Output System (NetBIOS) packets, use the **debug netbios packet**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug netbios packet

no debug netbios packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

For complete information on the NetBIOS process, use the **debug netbios error** command along with the **debug netbios packet** command.

Examples

The following is sample output from the **debug netbios packet**and **debug netbios error** commands. This example shows the Logical Link Control (LLC) header for an asynchronous interface followed by the NetBIOS information. For additional information on the NetBIOS fields, refer to *IBM LAN Technical Reference IEEE 802.2* .

```
Router# debug netbios packet
Async1 (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_NAME_QUERY
 Resp_correlator= 0x6F 0x0
 Src name=CS-NT-1
Async1 (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_GROUP_QUERY
 Resp_correlator= 0x6F 0x0
 Src name=COMMSERVER-WG
Async1 (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_NAME_QUERY
 Resp_correlator= 0x6F 0x0
 Src name=CS-NT-1
Ethernet0 (i) U-format UI C_R=0x0
(i) NETBIOS_DATAGRAM
 Length= 0x2C 0x0
 Dest name=COMMSERVER-WG
 Src name=CS-NT-3
```

Related Commands

Command	Description
debug netbios error	Displays information about NetBIOS protocol errors.
debug netbios-name-cache	Displays name caching activities on a router.

debug netbios-name-cache

To display name caching activities on a router, use the **debugnetbios-name-cache**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug netbios-name-cache

no debug netbios-name-cache

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Examine the display to diagnose problems in Network Basic Input/Output System (NetBIOS) name caching.

Examples

The following is sample output from the **debugnetbios-name-cache**command:

```
Router# debug netbios-name-cache
NETBIOS: L checking name ORINDA, vrn=0
NetBIOS name cache table corrupted at offset 13
NetBIOS name cache table corrupted at later offset, at location 13
NETBIOS: U chk name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
NETBIOS: U upd name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
NETBIOS: U add name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
NETBIOS: U no memory to add cache entry. name=ORINDA,addr=1000.4444.5555
NETBIOS: Invalid structure detected in netbios_name_cache_ager
NETBIOS: flushed name=ORINDA, addr=1000.4444.5555
NETBIOS: expired name=ORINDA, addr=1000.4444.5555
NETBIOS: removing entry. name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0
NETBIOS: Tossing ADD_NAME/STATUS/NAME/ADD_GROUP frame
NETBIOS: Lookup Failed -- not in cache
NETBIOS: Lookup Worked, but split horizon failed
NETBIOS: Could not find RIF entry
NETBIOS: Cannot duplicate packet in netbios_name_cache_proxy
```

Note The sample display is a composite output. Debugging output that you actually see would not necessarily occur in this sequence.

The following table describes the significant fields shown in the display.

Table 4: debug netbios-name-cache Field Descriptions

Field	Description
NETBIOS	NetBIOS name caching debugging output.
L, U	L means lookup; U means update.
addr=1000.4444.5555	MAC address of machine being looked up in NetBIOS name cache.

Field	Description
idb=TR1	Indicates that the name of machine was learned from Token Ring interface number 1; idb is into interface data block.
vrn=0	Packet comes from virtual ring number 0. This packet actually comes from a real Token Ring interface, because virtual ring number 0 is not valid.
type=1	Indicates the way that the router learned about the specified machine. The possible values are as follows: • 1--Learned from traffic • 2--Learned from a remote peer • 4--Statically entered via the configuration of the router

With the first line of output, the router declares that it has examined the NetBIOS name cache table for the machine name ORINDA and that the packet that prompted the lookup came from virtual ring 0. In this case, this packet comes from a real interface--virtual ring number 0 is not valid.

```
NETBIOS: L checking name ORINDA, vrn=0
```

The following two lines indicate that an invalid NetBIOS entry exists and that the corrupted memory was detected. The invalid memory will be removed from the table; no action is needed.

```
NetBIOS name cache table corrupted at offset 13
NetBIOS name cache table corrupted at later offset, at location 13
```

The following line indicates that the router attempted to check the NetBIOS cache table for the name ORINDA with MAC address 1000.4444.5555. This name was obtained from Token Ring interface 1. The type field indicates that the name was learned from traffic.

```
NETBIOS: U chk name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
```

The following line indicates that the NetBIOS name ORINDA is in the name cache table and was updated to the current value:

```
NETBIOS: U upd name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
```

The following line indicates that the NetBIOS name ORINDA is not in the table and must be added to the table:

```
NETBIOS: U add name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
```

The following line indicates that there was insufficient cache buffer space when the router tried to add this name:

```
NETBIOS: U no memory to add cache entry. name=ORINDA,addr=1000.4444.5555
```

The following line indicates that the NetBIOS ager detects an invalid memory in the cache. The router clears the entry; no action is needed.

```
NETBIOS: Invalid structure detected in netbios_name_cache_ager
```

The following line indicates that the entry for ORINDA was flushed from the cache table:

```
NETBIOS: flushed name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the entry for ORINDA timed out and was flushed from the cache table:

```
NETBIOS: expired name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the router removed the ORINDA entry from its cache table:

```
NETBIOS: removing entry. name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0
```

The following line indicates that the router discarded a NetBIOS packet of type ADD_NAME, STATUS, NAME_QUERY, or ADD_GROUP. These packets are discarded when multiple copies of one of these packet types are detected during a certain period of time.

```
NETBIOS: Tossing ADD_NAME/STATUS/NAME/ADD_GROUP frame
```

The following line indicates that the system could not find a NetBIOS name in the cache:

```
NETBIOS: Lookup Failed -- not in cache
```

The following line indicates that the system found the destination NetBIOS name in the cache, but located on the same ring from which the packet came. The router will drop this packet because the packet should not leave this ring.

```
NETBIOS: Lookup Worked, but split horizon failed
```

The following line indicates that the system found the NetBIOS name in the cache, but the router could not find the corresponding RIF. The packet will be sent as a broadcast frame.

```
NETBIOS: Could not find RIF entry
```

The following line indicates that no buffer was available to create a NetBIOS name cache proxy. A proxy will not be created for the packet, which will be forwarded as a broadcast frame.

```
NETBIOS: Cannot duplicate packet in netbios_name_cache_proxy
```

Related Commands

Command	Description
debug netbios error	Displays information about NetBIOS protocol errors.
debug netbios packet	Displays general information about NetBIOS packets.

debug netconf

To enable debugging of network configuration protocol (NETCONF) sessions, use the **debug netconf** command in privileged EXEC mode. To turn off NETCONF debugging, use the **no** form of this command.

debug netconf {**all**| **error**}

no debug netconf {**all**| **error**}

Syntax Description

all	Enables debugging of NETCONF sessions, including NETCONF errors.
error	Enables debugging of NETCONF errors.

Command Default

NETCONF debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRA	This command was introduced.
12.4(9)T	This command was integrated into Cisco IOS Release 12.4(9)T.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

Usage Guidelines

The **debug netconf**command issues debug information only when an operational error has happened. In most situations, the NETCONF notifications sent between the NETCONF Network Manager and the client are sufficient to diagnose most NETCONF problems.

To view Extensible Markup Language (XML) parsing errors when using NETCONF over SSHv2, you must also configure the **debug cns xml all** command.

Examples

The following example shows how to enable debugging of all NETCONF sessions:

```
Router# debug netconf

00:14:03: NETCONF-ERROR: could not find user1
00:14:03: NETCONF-ERROR: could not find tftp://samplelocation/samplefile
00:14:03: NETCONF: locking 1 by session 646B7038
00:14:03: NETCONF: locking 2 by session 646B7038
00:14:03: NETCONF: locking 1 by session 646B7038
00:14:03: NETCONF-ERROR: invalid session unlock attempt
00:14:03: NETCONF: locking 1 by session 646B7038
00:14:03: NETCONF-ERROR: lock already active
00:14:13: NETCONF-ERROR: lock time 1 expired closing session 646B7038
```

The following table describes the significant fields shown in the display.

Table 5: debug netconf Field Descriptions

Field	Description
NETCONF-ERROR: could not find user1	NETCONF could not find the specified username.
NETCONF-ERROR: could not find tftp://samplelocation/samplefile	NETCONF could not find the specified file path.
NETCONF: locking 1 by session 646B7038	This user is locking NETCONF.
NETCONF-ERROR: invalid session unlock attempt	Another user is trying to unlock NETCONF without first acquiring the lock.
NETCONF-ERROR: lock already active	Another user is trying to lock NETCONF while it is currently locked.
NETCONF-ERROR: lock time 1 expired closing session 646B7038	A locked NETCONF session has been idle longer than the time configured by the **netconf lock-time** command. The locked NETCONF session is closed.

Related Commands

Command	Description
clear netconf	Clears NETCONF statistics counters, NETCONF sessions, and frees associated resources and locks.
debug cns xml	Turns on debugging messages related to the CNS XML parser.
netconf lock-time	Specifies the maximum time a NETCONF configuration lock is in place without an intermediate operation.
netconf max-sessions	Specifies the maximum number of concurrent NETCONF sessions allowed.

Command	Description
netconf ssh	Enables NETCONF over SSHv2.
show netconf	Displays NETCONF statistics counters and session information.

debug nextport vsmgr detail

To turn on debugging for NextPort voice services, use the **debug nextport vsmgr detail**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug nextport vsmgr detail

no debug nextport vsmgr detail

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)XB	This command was introduced.
12.3(4)T	This command was integrated into Cisco IOS Release 12.3(4)T.
12.3(14)T	T.38 fax relay call statistics were made available to Call Detail Records (CDRs) through Vendor-Specific Attributes (VSAs) and added to the call log.

Usage Guidelines

This command debugs digital signal processor (DSP) message exchanges between applications and the DSP.

Examples

The following examples turn on debugging for NextPort voice services:

Examples

```
Router# debug nextport vsmgr detail
NextPort Voice Service Manager:
  NP Voice Service Manager Detail debugging is on
.
.
.
May  7 21:09:49.135 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:49.195 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:49.291 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:09:51.191 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:51.331 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:09:51.803 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:09:51.803 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:09:51.803 UTC: VOICE_TRANSMIT_STATS(1/2): num_voice_packets 4 num_sig_packets 0
num_cn_packets 1 transmit_duration 8FC end_point_detection 0
May  7 21:09:51.803 UTC: VOICE_RECEIVE_STATS(1/2): num_voice_packets 4 num_sig_packets 0
num_cn_packets 1 receive_duration 8FC  voice_receive_duration 0 num_pos_packets 0
```

```
num_bph_packets 0 num_late_packets 0 num_early_packets 0
May  7 21:09:51.803 UTC: VOICE_PLAYOUT_DELAY_STATS(1/2): curr_playout_delay 0
min_playout_delay 0 max_playout_delay 0 clock offset 0
May  7 21:09:51.803 UTC: VOICE_PLAYOUT_ERROR(1/2): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x0 endpt_det_error 0x0
May  7 21:09:53.231 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:09:53.231 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:09:53.231 UTC: VOICE_TRANSMIT_STATS(1/2): num_voice_packets 1E num_sig_packets 0
 num_cn_packets 1 transmit_duration E92 end_point_detection 0
May  7 21:09:53.231 UTC: VOICE_RECEIVE_STATS(1/2): num_voice_packets 4 num_sig_packets 0
num_cn_packets 1 receive_duration E92  voice_receive_duration 0 num_pos_packets 0
num_bph_packets 0 num_late_packets 0 num_early_packets 0
May  7 21:09:53.231 UTC: VOICE_PLAYOUT_DELAY_STATS(1/2): curr_playout_delay 5A
min_playout_delay 5A max_playout_delay 5A clock offset 19778906
May  7 21:09:53.231 UTC: VOICE_PLAYOUT_ERROR(1/2): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x0 endpt_det_error 0x0
May  7 21:09:56.055 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:09:56.055 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:09:56.055 UTC: VOICE_TRANSMIT_STATS(1/2): num_voice_packets 23 num_sig_packets 0
 num_cn_packets 2 transmit_duration 19A0 end_point_detection BB8
May  7 21:09:56.055 UTC: VOICE_RECEIVE_STATS(1/2): num_voice_packets 8A num_sig_packets 0
num_cn_packets 1 receive_duration 19A0  voice_receive_duration 0 num_pos_packets 0
num_bph_packets 0 num_late_packets 0 num_early_packets 1
May  7 21:09:56.055 UTC: VOICE_PLAYOUT_DELAY_STATS(1/2): curr_playout_delay 3C
min_playout_delay 3C max_playout_delay 64 clock offset 197788E4
May  7 21:09:56.055 UTC: VOICE_PLAYOUT_ERROR(1/2): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x1 endpt_det_error 0x0
May  7 21:09:56.855 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:57.907 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:09:57.907 UTC: FAX_RELAY_LINK_INFO_RSP_NTF: slot 1 port 2 timestamp 68137565
fr-entered (20ms)
May  7 21:09:57.907 UTC: chan_id [3/1:D] np_vsmgr_fax_relay_link_info_response:
May  7 21:10:15.047 UTC: np_fax_relay_t30_decode : Tx Direction
May  7 21:10:15.067 UTC: FARELAY_INIT_HS_MOD : 0xC
May  7 21:10:51.579 UTC: FAX_RELAY_DATA_PUMP_STATS(1/2) - valid:0x3FFC1F55 state_code:0x0
level:0x18 phase_jitter:0x5 freq_offset:0x0 eqm:0x7FFE jit_depth:0x230 jit_buf_ov:0x0
tx_paks:0x626 rx_pkts:0x5A inv_pkts:0x0 oos_pkts:0x0 hs_mod:0x8 init_hs_mod:0xC tx_pgs:0x1
 rx_pgs:0x0 ecm:0x1 nsf_country:0x0 nsf_manuf_len:0x20
nsf_manuf:0031B8EE80C48511DD0D0000DDDD0000DDDD000000000000000022ED00B0A400 encap:0x1
pkt_loss_con:0x0
May  7 21:10:52.463 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:10:52.463 UTC: vsm(1/2): np_vsmgr_voice_state_change - NULL DSP Interface Handle
```

Examples

```
Router# debug nextport vsmgr detail
NextPort Voice Service Manager:
  NP Voice Service Manager Detail debugging is on
.
.
Router#
May  7 21:09:51.179 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:51.263 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:09:51.303 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:51.443 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:09:51.467 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:09:51.467 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:09:51.467 UTC: VOICE_TRANSMIT_STATS(1/2): num_voice_packets 0 num_sig_packets 0
num_cn_packets 0 transmit_duration 0 end_point_detection 0
May  7 21:09:51.467 UTC: VOICE_RECEIVE_STATS(1/2): num_voice_packets 0 num_sig_packets 0
num_cn_packets 0 receive_duration 0  voice_receive_duration 0 num_pos_packets 0
num_bph_packets 0 num_late_packets 0 num_early_packets 0
May  7 21:09:51.467 UTC: VOICE_PLAYOUT_DELAY_STATS(1/2): curr_playout_delay 0
min_playout_delay 0 max_playout_delay 0 clock offset 0
May  7 21:09:51.467 UTC: VOICE_PLAYOUT_ERROR(1/2): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x0 endpt_det_error 0x0
May  7 21:09:53.787 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:09:53.787 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:09:53.787 UTC: VOICE_TRANSMIT_STATS(1/2): num_voice_packets 19 num_sig_packets 0
 num_cn_packets 1 transmit_duration 910 end_point_detection 0
May  7 21:09:53.787 UTC: VOICE_RECEIVE_STATS(1/2): num_voice_packets 1F num_sig_packets 0
num_cn_packets 2 receive_duration 910  voice_receive_duration 0 num_pos_packets 0
num_bph_packets 0 num_late_packets 0 num_early_packets 0
May  7 21:09:53.787 UTC: VOICE_PLAYOUT_DELAY_STATS(1/2): curr_playout_delay 5A
min_playout_delay 5A max_playout_delay 5A clock offset 68877C4
May  7 21:09:53.787 UTC: VOICE_PLAYOUT_ERROR(1/2): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x0 endpt_det_error 0x0
May  7 21:09:56.571 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:09:56.571 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:09:56.571 UTC: VOICE_TRANSMIT_STATS(1/2): num_voice_packets A5 num_sig_packets 0
 num_cn_packets 1 transmit_duration 13F6 end_point_detection 0
May  7 21:09:56.571 UTC: VOICE_RECEIVE_STATS(1/2): num_voice_packets 30 num_sig_packets 0
num_cn_packets 2 receive_duration 13F6  voice_receive_duration 7D0 num_pos_packets 0
num_bph_packets 0 num_late_packets 0 num_early_packets 0
May  7 21:09:56.571 UTC: VOICE_PLAYOUT_DELAY_STATS(1/2): curr_playout_delay 64
min_playout_delay 5A max_playout_delay 64 clock offset 68877D4
May  7 21:09:56.571 UTC: VOICE_PLAYOUT_ERROR(1/2): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x0 endpt_det_error 0x0
May  7 21:09:56.807 UTC: VOICE_DET_STATUS_CHANGE_NTF(1/2): detector mask: 1 timestamp
791687D5
May  7 21:09:56.855 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:09:57.911 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:09:57.911 UTC: FAX_RELAY_LINK_INFO_RSP_NTF: slot 1 port 2 timestamp 65325022
fr-entered (20ms)
May  7 21:09:57.911 UTC: chan_id [3/1:D (6)] np_vsmgr_fax_relay_link_info_response:
May  7 21:10:15.043 UTC: np_fax_relay_t30_decode : Rx Direction
May  7 21:10:15.107 UTC: FARELAY_INIT_HS_MOD : 0x8
May  7 21:10:51.376 UTC: FAX_RELAY_DET_STATUS_CHANGE: slot: 1 port: 2 detector mask 0x2
May  7 21:10:51.404 UTC: FAX_RELAY_DATA_PUMP_STATS(1/2) - valid:0x3FFC1F55 state_code:0x1
level:0x18 phase_jitter:0x0 freq_offset:0x0 eqm:0x7FFE jit_depth:0x39E jit_buf_ov:0x0
tx_paks:0x5A rx_pkts:0x626 inv_pkts:0x0 oos_pkts:0x0 hs_mod:0x8 init_hs_mod:0x8 tx_pgs:0x0
 rx_pgs:0x1 ecm:0x1 nsf_country:0x0 nsf_manuf_len:0x20
nsf_manuf:0031B8EE80C48511DD0D0000DDDD0000DDDD000000000000000022ED00B0A400 encap:0x1
pkt_loss_con:0x0
May  7 21:10:52.288 UTC: FAX_RELAY_LINK_INFO_RSP_NTF: slot 1 port 2 timestamp 65760060
fr-end
May  7 21:10:52.304 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:10:52.388 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:10:52.416 UTC: np_vsmgr_dispatch_voice_rsp(1/2): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:10:52.416 UTC: request_id = 0x05, request_type = 0x30
May  7 21:10:52.416 UTC: VOICE_LEVELS_STATS(1/2): tx_power FF7E tx_mean FF7F rx_power FDBD
 rx_mean FB48  bnl FD81 erl FD acom 1EA tx_act 1 rx_act 0
May  7 21:10:52.440 UTC: vsm(1/2): np_vsmgr_voice_state_change() - state IDLE
May  7 21:10:52.440 UTC: vsm(1/2): np_vsmgr_voice_state_change - NULL DSP Interface Handle
```

Related Commands

Command	Description
debug dspapi detail	Displays details of the DSP API message events with debugging enabled.
voicecap entry	Creates a voicecap on NextPort platforms.
voicecap configure	Applies a voicecap on NextPort platforms.

debug nhrp

To enable Next Hop Resolution Protocol (NHRP) debugging, use the **debug nhrp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug nhrp [**attribute**| **cache** | **condition**{**interface tunnel** *number*| **Virtual-Access** *number*| **peer**{**nbma**{*nbma-address*| *nbma-name*}| **tunnel**{*ipv4-address*| *ipv6-address/mask*}}| **vrf** *vrf-name*}| **detail**| **error**| **extension**| **group**| **packet**| **rate**| **routing**]

no debug nhrp[**attribute**| **cache** | **condition**{**interface tunnel** *number*| **Virtual-Access** *number*| **peer**{**nbma**{*nbma-address*| *nbma-name*}| **tunnel**{*ipv4-address*| *ipv6-address/mask*}}| **vrf** *vrf-name*}| **detail**| **error**| **extension**| **group**| **packet**| **rate**| **routing**]

Syntax Description

attribute	(Optional) Enables NHRP attribute debugging operations.
cache	(Optional) Enables NHRP cache debugging operations.
condition	(Optional) Enables NHRP conditional debugging operations.
interface tunnel *number*	Enables debugging operations for the tunnel interface.
Virtual-Access *number*	Enables debugging operations for the virtual access interface.
nbma	Enables debugging operations for the non-broadcast multiple access (NBMA) network.
nbma-address	Enables debugging operations based on the IPv4 address of the NBMA network.
nbma-name	NBMA network name.
tunnel {*IPv4-address* | *IPv6-address/mask*}	Enables debugging operations for IPv4 or IPv6 addresses of the tunnel in the NBMA network.
vrf *vrf-name*	Enables debugging operations for the tunnel interface.
detail	(Optional) Displays detailed logs of NHRP debugs.
error	(Optional) Enables NHRP error debugging operations.
extension	(Optional) Enables NHRP extension processing debugging operations.

group	(Optional) Enables NHRP group debugging operations.
packet	(Optional) Enables NHRP activity debugging.
rate	(Optional) Enables NHRP rate limiting.
routing	(Optional) Enables NHRP routing debugging operations.

Command Default

NHRP debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(20)T	This command was introduced.
15.3(2)T	This command was modified. The **detail** keyword was added and the command output was enhanced to display more NHRP debugging information. The **Virtual-Access** *number* keyword-argument pair was added.

Usage Guidelines

Use the **debug nhrp detail** command to view the NHRP attribute logs.

The **Virtual-Access** *number* keyword-argument pair is visible only if the virtual access interface is available on the device.

Examples

The following example shows NHRP debugging output for IPv6:

```
Router# debug nhrp
Aug  9 13:13:41.486: NHRP: Attempting to send packet via DEST
   - 2001:0db8:3c4d:0015:0000:0000:1a2f:3d2c/32
Aug  9 13:13:41.486: NHRP: Encapsulation succeeded.
Aug  9 13:13:41.486: NHRP: Tunnel NBMA addr 11.11.11.99
Aug  9 13:13:41.486: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 105
Aug  9 13:13:41.486: src: 2001:0db8:3c4d:0015:0000:0000:1a2f:3d2c/32,
        dst: 2001:0db8:3c4d:0015:0000:0000:1a2f:3d2c/32
Aug  9 13:13:41.486: NHRP: 105 bytes out Tunnel0
Aug  9 13:13:41.486: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 125
```

The following example shows NHRP debugging output for IPv4:

```
Router# debug nhrp
Aug  9 13:13:41.486: NHRP: Attempting to send packet via DEST 10.1.1.99
Aug  9 13:13:41.486: NHRP: Encapsulation succeeded.  Tunnel IP addr 10.11.11.99
Aug  9 13:13:41.486: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 105
Aug  9 13:13:41.486:       src: 10.1.1.11, dst: 10.1.1.99
Aug  9 13:13:41.486: NHRP: 105 bytes out Tunnel0
Aug  9 13:13:41.486: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 125
Aug  9 13:13:41.486: NHRP: netid_in = 0, to_us = 1
```

The following example shows NHRP debugging output for the **detail** keyword:

```
Device# debug nhrp detail
NHRP detail debugging is on

*Aug  3 06:28:38.077: NHRP: if_up: Tunnel0 proto 'NHRP_IPv4'
*Aug  3 06:28:38.077: NHRP: Registration with Tunnels Decap Module succeeded
*Aug  3 06:28:38.077: NHRP: Adding all static maps to cache
*Aug  3 06:28:38.077: NHRP: Adding Tunnel Endpoints (VPN: 10.0.0.254, NBMA: 172.16.1.4)
*Aug  3 06:28:38.077: NHRP: Successfully attached NHRP subblock for Tunnel Endpoints (VPN:
 10.0.0.254, NBMA: 172.16.1.4)
*Aug  3 06:28:38.077: NHRP: if_up: Tunnel0 proto 'NHRP_IPv6'
*Aug  3 06:28:38.077: NHRP: Registration with Tunnels Decap Module succeeded
*Aug  3 06:28:38.077: NHRP: Adding all static maps to cache
*Aug  3 06:28:38.077: NHRP: Adding Tunnel Endpoints (VPN: 2001::2, NBMA: 172.16.1.4)
*Aug  3 06:28:38.078: NHRP: Successfully attached NHRP subblock for Tunnel Endpoints (VPN:
 2001::2, NBMA: 172.16.1.4)

*Aug  3 06:28:38.078: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
*Aug  3 06:28:38.079: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
*Aug  3 06:28:38.716: %SYS-5-CONFIG_I: Configured from console by console
*Aug  3 06:28:39.030: NHRP: Sending one-time request for nhs 2001::2
*Aug  3 06:28:39.030: NHRP-ATTR:  Requester Ext Len: Total ext_len  with NHRP attribute VPE
 64

*Aug  3 06:28:39.030: NHRP: Attempting to send packet through interface Tunnel0 via dst
2001::2
*Aug  3 06:28:39.030: NHRP: Encapsulation succeeded.  Sending NHRP Control Packet  NBMA
Address: 172.16.1.4
*Aug  3 06:28:39.030: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 160
*Aug  3 06:28:39.030:       src: 2001::3, dst: 2001::2
*Aug  3 06:28:39.031: NHRP: 188 bytes out Tunnel0
*Aug  3 06:28:39.032: NHRP-ATTR: ext_type: 32771, ext_len : 32

*Aug  3 06:28:39.032: NHRP-ATTR: ext_type: 32772, ext_len : 0

*Aug  3 06:28:39.032: NHRP-ATTR: ext_type: 32773, ext_len : 0

*Aug  3 06:28:39.032: NHRP-ATTR: ext_type: 32775, ext_len : 8

*Aug  3 06:28:39.032: NHRP-ATTR: ext_type: 9, ext_len : 32

*Aug  3 06:28:39.032: NHRP-ATTR: ext_type: 32768, ext_len : 0

*Aug  3 06:28:39.032: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 224
*Aug  3 06:28:39.032: NHRP: netid_in = 0, to_us = 1
*Aug  3 06:28:39.032: NHRP: NHS-UP: 2001::2
*Aug  3 06:28:39.032: NHRP: Adding Tunnel Endpoints (VPN: FE80::A8BB:CCFF:FE01:F500, NBMA:
 172.16.1.4)
*Aug  3 06:28:39.032: NHRP: Successfully attached NHRP subblock for Tunnel Endpoints (VPN:
 FE80::A8BB:CCFF:FE01:F500, NBMA: 172.16.1.4)
*Aug  3 06:28:39.032: NHRP: Caching Additional Address: FE80::A8BB:CCFF:FE01:F500, cache:
0x0x2A98CBCE28, hold_time: 300
*Aug  3 06:28:39.060: NHRP: Sending one-time request for nhs 10.0.0.254
*Aug  3 06:28:39.060: NHRP-ATTR:  Requester Ext Len: Total ext_len  with NHRP attribute VPE
 52

*Aug  3 06:28:39.060: NHRP: Attempting to send packet through interface Tunnel0 via dst
10.0.0.254
*Aug  3 06:28:39.060: NHRP: Encapsulation succeeded.  Sending NHRP Control Packet  NBMA
Address: 172.16.1.4
*Aug  3 06:28:39.060: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 104
*Aug  3 06:28:39.060:       src: 10.0.0.2, dst: 10.0.0.254
*Aug  3 06:28:39.060: NHRP: 132 bytes out Tunnel0
*Aug  3 06:28:39.061: NHRP-ATTR: ext_type: 32771, ext_len : 20

*Aug  3 06:28:39.061: NHRP-ATTR: ext_type: 32772, ext_len : 0

*Aug  3 06:28:39.061: NHRP-ATTR: ext_type: 32773, ext_len : 0
```

```
*Aug  3 06:28:39.061: NHRP-ATTR: ext_type: 32775, ext_len : 8

*Aug  3 06:28:39.061: NHRP-ATTR: ext_type: 9, ext_len : 20

*Aug  3 06:28:39.061: NHRP-ATTR: ext_type: 32768, ext_len : 0

*Aug  3 06:28:39.061: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 124
*Aug  3 06:28:39.061: NHRP: netid_in = 0, to_us = 1
*Aug  3 06:28:39.061: NHRP: NHS-UP: 10.0.0.254
*Aug  3 06:28:39.080: NHRP-ATTR:  Requester Ext Len: Total ext_len  with NHRP attribute VPE
 52

*Aug  3 06:28:39.080: NHRP: Attempting to send packet through interface Tunnel0 via dst
10.0.0.254
*Aug  3 06:28:39.080: NHRP: Encapsulation succeeded.  Sending NHRP Control Packet  NBMA
Address: 172.16.1.4
*Aug  3 06:28:39.080: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 104
*Aug  3 06:28:39.080:       src: 10.0.0.2, dst: 10.0.0.254
*Aug  3 06:28:39.080: NHRP: 132 bytes out Tunnel0
*Aug  3 06:28:39.080: NHRP-ATTR:  Requester Ext Len: Total ext_len  with NHRP attribute VPE
 64

*Aug  3 06:28:39.080: NHRP: Attempting to send packet through interface Tunnel0 via dst
2001::2
*Aug  3 06:28:39.081: NHRP: Encapsulation succeeded.  Sending NHRP Control Packet  NBMA
Address: 172.16.1.4
*Aug  3 06:28:39.081: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 160
*Aug  3 06:28:39.081:       src: 2001::3, dst: 2001::2
*Aug  3 06:28:39.081: NHRP: 188 bytes out Tunnel0
*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32771, ext_len : 20

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32772, ext_len : 0

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32773, ext_len : 0

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32775, ext_len : 8

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 9, ext_len : 20

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32768, ext_len : 0

*Aug  3 06:28:39.081: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 124
*Aug  3 06:28:39.081: NHRP: netid_in = 0, to_us = 1
*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32771, ext_len : 32

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32772, ext_len : 0

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32773, ext_len : 0

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32775, ext_len : 8

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 9, ext_len : 32

*Aug  3 06:28:39.081: NHRP-ATTR: ext_type: 32768, ext_len : 0

*Aug  3 06:28:39.081: NHRP: Receive Registration Reply via Tunnel0

vrf 0, packet size: 224
*Aug  3 06:28:39.082: NHRP: netid_in = 0, to_us = 1
*Aug  3 06:28:39.082: NHRP: Adding Tunnel Endpoints (VPN: FE80::A8BB:CCFF:FE01:F500, NBMA:
 172.16.1.4)
*Aug  3 06:28:39.082: NHRP: NHRP subblock already exists for Tunnel Endpoints (VPN:
FE80::A8BB:CCFF:FE01:F500, NBMA: 172.16.1.4)
*Aug  3 06:28:39.082: NHRP: Cache already has a subblock node attached for Tunnel Endpoints
 (VPN: FE80::A8BB:CCFF:FE01:F500, NBMA: 172.16.1.4)
*Aug  3 06:28:39.082: NHRP: Caching Additional Address: FE80::A8BB:CCFF:FE01:F500, cache:
0x0x2A98CBCE28, hold_time: 300
*Aug  3 06:28:40.080: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state
 to up
*Aug  3 06:28:40.081: NHRP: if_up: Tunnel0 proto 'NHRP_IPv4'
*Aug  3 06:28:40.081: NHRP: Registration with Tunnels Decap Module succeeded
*Aug  3 06:28:40.081: NHRP: Adding all static maps to cache
*Aug  3 06:28:40.081: NHRP: Adding Tunnel Endpoints (VPN: 10.0.0.254, NBMA: 172.16.1.4)
```

```
*Aug  3 06:28:40.081: NHRP: NHRP subblock already exists for Tunnel Endpoints (VPN:
10.0.0.254, NBMA: 172.16.1.4)
*Aug  3 06:28:40.081: NHRP: Cache already has a subblock node attached for Tunnel Endpoints
 (VPN: 10.0.0.254, NBMA: 172.16.1.4)
*Aug  3 06:28:40.081: NHRP-ATTR:  Requester Ext Len: Total ext_len  with NHRP attribute VPE
 52

*Aug  3 06:28:40.081: NHRP: Attempting to send packet through interface Tunnel0 via dst
10.0.0.254
*Aug  3 06:28:40.081: NHRP: Encapsulation succeeded.  Sending NHRP Control Packet  NBMA
Address: 172.16.1.4
*Aug  3 06:28:40.081: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 104
*Aug  3 06:28:40.081:       src: 10.0.0.2, dst: 10.0.0.254
*Aug  3 06:28:40.081: NHRP: 132 bytes out Tunnel0
*Aug  3 06:28:40.081: NHRP: if_up: Tunnel0 proto 'NHRP_IPv6'
*Aug  3 06:28:40.081: NHRP: Registration with Tunnels Decap Module succeeded
*Aug  3 06:28:40.081: NHRP: Adding all static maps to cache
*Aug  3 06:28:40.081: NHRP: Adding Tunnel Endpoints (VPN: 2001::2, NBMA: 172.16.1.4)
*Aug  3 06:28:40.081: NHRP: NHRP subblock already exists for Tunnel Endpoints (VPN: 2001::2,
 NBMA: 172.16.1.4)
*Aug  3 06:28:40.081: NHRP: Cache already has a subblock node attached for Tunnel Endpoints
 (VPN: 2001::2, NBMA: 172.16.1.4)
*Aug  3 06:28:40.081: NHRP-ATTR:  Requester Ext Len: Total ext_len  with NHRP attribute VPE
 64

*Aug  3 06:28:40.081: NHRP: Attempting to send packet through interface Tunnel0 via dst
2001::2
*Aug  3 06:28:40.081: NHRP: Encapsulation succeeded.  Sending NHRP Control Packet  NBMA
Address: 172.16.1.4
*Aug  3 06:28:40.081: NHRP: Send Registration Request via Tunnel0 vrf 0, packet size: 160
*Aug  3 06:28:40.081:       src: 2001::3, dst: 2001::2
*Aug  3 06:28:40.081: NHRP: 188 bytes out Tunnel0
*Aug  3 06:28:40.081: %LINK-3-UPDOWN: Interface Tunnel0, changed state to up
*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32771, ext_len : 20

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32772, ext_len : 0

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32773, ext_len : 0

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32775, ext_len : 8

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 9, ext_len : 20

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32768, ext_len : 0

*Aug  3 06:28:40.084: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 124
*Aug  3 06:28:40.084: NHRP: netid_in = 0, to_us = 1
*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32771, ext_len : 32

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32772, ext_len : 0

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32773, ext_len : 0

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32775, ext_len : 8

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 9, ext_len : 32

*Aug  3 06:28:40.084: NHRP-ATTR: ext_type: 32768, ext_len : 0

*Aug  3 06:28:40.084: NHRP: Receive Registration Reply via Tunnel0 vrf 0, packet size: 224
*Aug  3 06:28:40.084: NHRP: netid_in = 0, to_us = 1
*Aug  3 06:28:40.084: NHRP: Adding Tunnel Endpoints (VPN: FE80::A8BB:CCFF:FE01:F500, NBMA:
 172.16.1.4)
*Aug  3 06:28:40.084: NHRP: NHRP subblock already exists for Tunnel Endpoints (VPN:
FE80::A8BB:CCFF:FE01:F500, NBMA: 172.16.1.4)
*Aug  3 06:28:40.084: NHRP:
Cache already has a subblock node attached for Tunnel Endpoints (VPN:
FE80::A8BB:CCFF:FE01:F500, NBMA: 172.16.1.4)
*Aug  3 06:28:40.084: NHRP: Caching Additional Address: FE80::A8BB:CCFF:FE01:F500, cache:
0x0x2A98CBCE28, hold_time: 300

*Aug  3 06:28:41.521: %DUAL-5-NBRCHANGE: EIGRP-IPv6 2: Neighbor FE80::A8BB:CCFF:FE01:F500
(Tunnel0) is up: new adjacency
```

```
*Aug  3 06:28:41.531: NHRP: Attempting to check and send Traffic Indication to NBMA: UNKNOWN
*Aug  3 06:28:41.531: NHRP: IPv6 NHRP Shortcut Enabled: Attempting switching
*Aug  3 06:28:41.570: NHRP: Attempting to check and send Traffic Indication to NBMA: UNKNOWN
*Aug  3 06:28:41.570: NHRP: IPv6 NHRP Shortcut Enabled: Attempting switching
*Aug  3 06:28:41.590: NHRP: Attempting to check and send Traffic Indication to NBMA: UNKNOWN
*Aug  3 06:28:41.590: NHRP: IPv6 NHRP Shortcut Enabled: Attempting switching
*Aug  3 06:28:41.610: NHRP: Attempting to check and send Traffic Indication to NBMA: UNKNOWN
*Aug  3 06:28:41.610: NHRP: IPv6 NHRP Shortcut Enabled: Attempting switching
*Aug  3 06:28:42.731: %DUAL-5-NBRCHANGE: EIGRP-IPv4 1: Neighbor 10.0.0.254 (Tunnel0) is up:
 new adjacency

*Aug  3 06:28:43.140: NHRP: Attempting to check and send Traffic Indication to NBMA: UNKNOWN
*Aug  3 06:28:43.140: NHRP: IPv6 NHRP Shortcut Enabled: Attempting switching
```

Related Commands

Command	Description
debug dmvpn	Displays DMVPN session debugging information.
debug nhrp error	Displays NHRP error-level debugging information.

debug nhrp condition

To enable Next Hop Resolution Protocol (NHRP) conditional debugging, use the **debug nhrp condition**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug nhrp condition [**interface tunnel** *number*| **peer** {**nbma** {*ip-address*| *FQDN-string*}| **tunnel** {*ip-address*| *ipv6-address*}}| **vrf** *vrf-name*]

no debug nhrp condition [**interface tunnel** *number*| **peer** {**nbma** {*ip-address*| *FQDN-string*}| **tunnel** {*ip-address*| *ipv6-address*}}| **vrf** *vrf-name*]

Syntax Description

tunnel	(Optional) Specifies a tunnel.
interface	(Optional) Displays NHRP information based on a specific interface.
tunnel *number*	(Optional) Specifies the tunnel address for the NHRP peer.
peer	(Optional) Specifies an NHRP peer.
nbma	(Optional) Specifies mapping nonbroadcast multiple access (NBMA).
ip-address	(Optional) The IPv4 address for the NHRP peer.
FQDN-string	(Optional) Next hop server (NHS) fully qualified domain name (FQDN) string.
ipv6-address	(Optional) The IPv6 address for the NHRP peer. **Note** Cisco IOS XE Release 2.5 does not support the ipv6-address argument.
vrf *vrf-name*	(Optional) Specifies debugging information for sessions related to the specified virtual routing and forwarding (VRF) configuration.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(9)T	This command was introduced.
12.4(20)T	This command was modified. The *ipv6-address* argument was added.

Release	Modification
Cisco IOS XE Release 2.5	This command was integrated into Cisco IOS XE Release 2.5 and implemented on the Cisco ASR 1000 Series Aggregation Services Routers.
15.1(2)T	This command was modified. The *FQDN-string* argument was added.

Examples

The following example shows how to enable conditional NHRP debugging for a specified NBMA address:

```
Router# debug nhrp condition peer tunnel 192.0.2.1
```

The following example shows how to enable conditional NHRP debugging for a specified FQDN string:

```
Router# debug nhrp condition peer examplehub.example1.com
```

Related Commands

Command	Description
debug dmvpn	Displays DMVPN session debugging information.
debug nhrp error	Displays NHRP error level debugging information.

debug nhrp error

To display Next Hop Resolution Protocol (NHRP) error-level debugging information, use the **debug nhrp error**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug nhrp {**ipv4**| **ipv6**} **error**

no debug nhrp {**ipv4**| **ipv6**} **error**

Syntax Description

ipv4	Specifies the IPv6 overlay network.
ipv6	Specifies the IPv6 overlay network. **Note** Cisco IOS XE Release 2.5 does not support the ipv6 keyword.

Command Default

NHRP error-level debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(9)T	This command was introduced.
12.4(20)T	The **ipv4** and **ipv6** keywords were added.
Cisco IOS XE Release 2.5	This command was modified. It was integrated into Cisco IOS XE Release 2.5.

Examples

The following example shows how to enable error level debugging for IPv4 NHRP:

```
Router# debug nhrp ipv4 error
NHRP errors debugging is on
```

Related Commands

Command	Description
debug dmvpn	Displays DMVPN session debugging information.
debug nhrp condition	Enables NHRP conditional debugging.

debug nhrp extension

To display the extensions portion of a NHRP packet, use the **debug nhrp extension**privileged EXEC command. The **no** form of this command disables debugging output.

NHRP:**debug nhrp extension commanddebug nhrp extension commanddebug nhrp extension**

no debug nhrp extension

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
10.0	This command was introduced.
12.2(13)T	This command is no longer supported in Cisco IOS Mainline or Technology-based (T) releases. It may continue to appear in Cisco IOS 12.2S-family releases.

Examples

The following is sample output from the **debug nhrp extension**command:

```
Router# debug nhrp extension
NHRP extension processing debugging is on
Router#
Forward Transit NHS Record Extension(4):
 (C-1) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 7200
       addr_len: 20(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client NBMA: 47.0091810000000002ba08e101.525354555354.01
       client protocol: 135.206.58.54
Reverse Transit NHS Record Extension(5):
Responder Address Extension(3):
 (C) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 7200
       addr_len: 20(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client NBMA: 47.0091810000000002ba08e101.525354555355.01
       client protocol: 135.206.58.55
Forward Transit NHS Record Extension(4):
 (C-1) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 7200
       addr_len: 20(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client NBMA: 47.0091810000000002ba08e101.525354555354.01
       client protocol: 135.206.58.54
Reverse Transit NHS Record Extension(5):
Responder Address Extension(3):
Forward Transit NHS Record Extension(4):
Reverse Transit NHS Record Extension(5):
```

debug nhrp options

To display information about NHRP option processing, use the **debugnhrpoptions**privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp options

no debug nhrp options

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
10.0	This command was introduced.
12.2(13)T	This command is no longer supported in Cisco IOS Mainline or Technology-based (T) releases. It may continue to appear in Cisco IOS 12.2S-family releases.

Usage Guidelines

Use this command to show you whether there are problems or error situations with NHRP option processing (for example, unknown options).

Examples

The following is sample output from the **debugnhrpoptions**command:

```
Router#
debug nhrp options
NHRP-OPT: MASK 4
NHRP-OPT-MASK: FFFFFFFF
NHRP-OPT: NETID 4
NHRP-OPT: RESPONDER 4
NHRP-OPT: RECORD 0
NHRP-OPT: RRECORD 0
```

The following table describes the significant fields shown in the display.

Table 6: debug nhrp options Field Descriptions

Field	Descriptions
NHRP-OPT	NHRP options debugging output.
MASK 4	Number of bytes of information in the destination prefix option.
NHRP-OPT-MASK	Contents of the destination prefix option.
NETID	Number of bytes of information in the subnetwork identifier option.

Field	Descriptions
RESPONDER	Number of bytes of information in the responder address option.
RECORD	Forward record option.
RRECORD	Reverse record option.

Related Commands

Command	Description
debug nhrp	Displays information about NHRP activity.
debug nhrp packet	Displays a dump of NHRP packets.

debug nhrp packet

To display a dump of NHRP packets, use the **debug nhrp packet**privileged EXEC command. The **no** form of this command disables debugging output.

NHRP:**debug nhrp packet commanddebug nhrp packet commanddebug nhrp packet**

no NHRP:**debug nhrp packet commanddebug nhrp packet commanddebug nhrp packet**

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
10.0	This command was introduced.
12.2(13)T	This command is no longer supported in Cisco IOS Mainline or Technology-based (T) releases. It may continue to appear in Cisco IOS 12.2S-family releases.

Examples

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

```
Router# debug nhrp packet
NHRP activity debugging is on
Router#
NHRP: Send Purge Request via ATM3/0.1, packet size: 72
 src: 135.206.58.55, dst: 135.206.58.56
  (F) afn: NSAP(3), type: IP(800), hop: 255, ver: 1
      shtl: 20(NSAP), sstl: 0(NSAP)
  (M) flags: "reply required", reqid: 2
      src NBMA: 47.0091810000000002ba08e101.525354555355.01
      src protocol: 135.206.58.55, dst protocol: 135.206.58.56
  (C-1) code: no error(0)
        prefix: 0, mtu: 9180, hd_time: 0
        addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
        client protocol: 135.206.58.130
NHRP: Receive Purge Reply via ATM3/0.1, packet size: 72
  (F) afn: NSAP(3), type: IP(800), hop: 254, ver: 1
      shtl: 20(NSAP), sstl: 0(NSAP)
  (M) flags: "reply required", reqid: 2
      src NBMA: 47.0091810000000002ba08e101.525354555355.01
      src protocol: 135.206.58.55, dst protocol: 135.206.58.56
  (C-1) code: no error(0)
        prefix: 0, mtu: 9180, hd_time: 0
        addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
        client protocol: 135.206.58.130
```

debug nhrp rate

To display information about NHRP traffic rate limits, use the **debugnhrprate**privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp rate

no debug nhrp rate

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
10.0	This command was introduced.
12.2(13)T	This command is no longer supported in Cisco IOS Mainline or Technology-based (T) releases. It may continue to appear in Cisco IOS 12.2S-family releases.

Usage Guidelines

Use this command to verify that the traffic is consistent with the setting of the NHRP commands (such as **ipnhrpuse** and**ipmax-send** commands).

Examples

The following is sample output from the **debugnhrprate**command:

```
Router#
debug nhrp rate
NHRP-RATE: Sending initial request
NHRP-RATE: Retransmitting request (retrans ivl 2)
NHRP-RATE: Retransmitting request (retrans ivl 4)
NHRP-RATE: Ethernet1: Used 3
```

The following table describes the significant fields shown in the display.

Table 7: debug nhrp rate Field Descriptions

Field	Descriptions
NHRP-RATE	NHRP rate debugging output.
Sending initial request	First time an attempt was made to send an NHRP packet to a particular destination.
Retransmitting request	Indicates that the NHRP packet was re-sent, and shows the time interval (in seconds) to wait before the NHRP packet is re-sent again.

Field	Descriptions
Ethernet1:	Interface over which the NHRP packet was sent.
Used 3	Number of packets sent out of the default maximum five (in this case, three were sent).

Related Commands

Command	Description
debug nhrp	Displays information about NHRP activity.
debug nhrp options	Displays information about NHRP option processing

debug ntp

To display debugging messages for Network Time Protocol (NTP) features, use the **debug ntp**command in prvileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ntp {**adjust**| **all**| **authentication**| **core**| **events**| **loopfilter**| **packet**| **params**| **refclock**| **select**| **sync**| **validity**}

no debug ntp {**adjust**| **all**| **authentication**| **core**| **events**| **loopfilter**| **packet**| **params**| **refclock**| **select**| **sync**| **validity**}

Syntax Description

adjust	Displays debugging information on NTP clock adjustments.
all	Displays all debugging information on NTP.
authentication	Displays debugging information on NTP authentication.
core	Displays debugging information on NTP core messages.
events	Displays debugging information on NTP events.
loopfilter	Displays debugging information on NTP loop filters.
packet	Displays debugging information on NTP packets.
params	Displays debugging information on NTP clock parameters.
refclock	Displays debugging information on NTP reference clocks.
select	Displays debugging information on NTP clock selection.
sync	Displays debugging information on NTP clock synchronization.
validity	Displays debugging information on NTP peer clock validity.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.1	This command was introduced in a release prior to Cisco IOS Release 12.1.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.4(20)T	Support for IPv6 and NTP version 4 was added. The **all** and **core** keywords were added. The **authentication**, **loopfilter**, **params**, **select**, **sync** and **validity** keywords were removed. The **packets** keyword was modified as **packet**.
Cisco IOS XE Release 3.5S	This command was integrated into Cisco IOS XE Release 3.5S.
Cisco IOS Release 15.2(1)S	This command was integrated into Cisco IOS Release 15.2(1)S.

Usage Guidelines

Starting from Cisco IOS Release 12.4(20)T, NTP version 4 is supported. In NTP version 4 the debugging options available are **adjust**, **all**, **core**, **events**, **packet**, and **refclock**. In NTP version 3 the debugging options available were **events**, **authentication**, **loopfilter**, **packets**, **params**, **select**, **sync** and **validity**.

Examples

The following example shows how to enable all debugging options for NTP:

```
Router# debug ntp all
NTP events debugging is on
NTP core messages debugging is on
NTP clock adjustments debugging is on
NTP reference clocks debugging is on
NTP packets debugging is on
```

Related Commands

Command	Description
ntp refclock	Configures an external clock source for use with NTP services.

debug oam

To display operation and maintenance (OAM) events, use the **debugoam** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug oam

no debug oam

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debugoam** command:

```
Router# debug oam
4/0(O): VCD:0x0 DM:0x300 *OAM Cell* Length:0x39
0000 0300 0070 007A 0018 0100 0000 05FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
FFFF FFFF FFFF FFFF FF6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A00 0000
```

The following table describes the significant fields shown in the display.

Table 8: debug oam Field Descriptions

Field	Description
0000	Virtual circuit designator (VCD) Special OAM indicator.
0300	Descriptor MODE bits for the ATM Interface Processor (AIP).
0	GFC (4 bits).
07	Virtual path identifier (VPI) (8 bits).
0007	Virtual channel identifier (VCI)(16 bits).
A	Payload type field (PTI) (4 bits).
00	Header Error Correction (8 bits).
1	OAM Fault mangement cell (4 bits).
8	OAM LOOPBACK indicator (4 bits).
01	Loopback indicator value, always 1 (8 bits).
00000005	Loopback unique ID, sequence number (32 bits).

Field	Description
FF6A	Fs and 6A required in the remaining cell, per UNI3.0.

debug object-group event

To enable debug messages for object-group events, use the **debug object-group event** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug object-group event

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.2(1)S	This command was introduced in Cisco IOS Release 15.2(1)S.
Cisco IOS XE Release 3.5	This command was introduced in Cisco IOS XE Release 3.5.

Usage Guidelines

When an object group is created to identify traffic coming from a specific user or endpoint, object-group identity mode is entered where a security group can be specified for the object group with a security group tag (SGT) ID. The SGT ID is used by a Security Group Access (SGA) Zone-Based Policy firewall (ZBPF) to apply an enforcement policy by filtering on this SGT ID. The **debug object-group event** command is used to view messages for object-group events while configuring the class map part of the SGA ZBPF.

Note A policy map must also be configured for the SGA ZBPF.

Examples

The following is sample output from the **debug object-group event** command:

```
Router# debug object-group event
Router# configure terminal
Router(config)# object-group security objsgt1
Router(config-security-group)# GLO INFO conf_objectgroup_cmd type(3) name(objsgt1)
Router(config-security-group)# security-group tag 120
Router(config-security-group)#
*Nov 21 16:23:02.041: INFO og_security_create_fn
*Nov 21 16:23:02.041: og_security_sgt_copy_fn:1633: object_group 'objsgt1' sgt name '' id
120
*Nov 21 16:23:02.041: og_classes_update:1373: walking class-maps in object_group 'objsgt1'
Router(config-security-group)#exit
Router(config)#
Router(config)# object-group security objsgt2
Router(config-security-group)# GLO INFO conf_objectgroup_cmd type(3) name(objsgt2)gr
Router(config-security-group)# group-object objsgt1
Router(config-security-group)#
*Nov 21 16:23:44.891: INFO og_security_create_fn
*Nov 21 16:23:44.891: og_classes_update:1373: walking class-maps in object_group 'objsgt2'
Router(config-security-group)#exit
```

Related Commands

Command	Description
group-object	Specifies a nested reference to a type of user group.
match group-object security	Matches traffic from a user in the security group.
object-group security	Creates an object group to identify traffic coming from a specific user or endpoint.
security-group	Specifies the membership of security group for an object group.
show object-group	Displays the content of all user groups.

debug oer api

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer api** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display Optimized Edge Routing (OER) application interface debugging information, use the **debug oer api**command in privileged EXEC mode. To stop the display of OER application interface debugging information, use the **no** form of this command.

debug oer api [detail]

no debug oer api

Syntax Description

detail	(Optional) Displays detailed application interface debugging information.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(15)T	This command was introduced.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer api**command is used to display messages about any configured OER application interface providers or host devices. The OER application interface defines the mode of communication and messaging between applications and the network for the purpose of optimizing the traffic associated with the applications. A provider is defined as an entity outside the network in which the router configured as an OER master controller exists, for example, an ISP, or a branch office of the same company. The provider has one or more host devices running one or more applications that use the OER application interface to communicate with an OER master controller. A provider must be registered with an OER master controller before an application on a host device can interface with OER. Use the **api provider** command to register the provider, and use the **host-address**command to configure a host device. After registration, a host device in the provider network can initiate a session with an OER master controller. The application interface provides an automated method for networks to be aware of applications and provides application-aware performance routing.

Caution When the **detail** keyword is entered, the amount of detailed output to be displayed can utilize a considerable amount of system resources. Use the **detail**keyword with caution in a production network.

Examples

The following example enables the display of OER application interface debugging messages and the output shows that an OER policy failed due to a prefix that is not found:

```
Router# debug oer api
OER api debugging is on
*May 26 01:04:07.278: OER API: Data set id received 5, data set len 9, host ip 10.3.3.3,
session id 1, requies2
*May 26 01:04:07.278: OER API: Received get current policy, session id 1 request id 22
*May 26 01:04:07.278: OER API: Recvd Appl with Prot 256 DSCP 0 SrcPrefix 0.0.0.0/0
SrcMask 0.0.0.0
*May 26 01:04:07.278: OER API:  DstPrefix 10.2.0.0/24 DstMask 255.255.255.0 Sport_min 0
Sport_max 0 Dport_mi0
*May 26 01:04:07.278: OER API: get prefix policy failed - prefix not found
*May 26 01:04:07.278: OER API: Get curr policy cmd received. rc 0
*May 26 01:04:07.278: OER API: Received send status response, status 0, session id 1,
request id 22, sequence0
*May 26 01:04:07.278: OER API: rc for data set 0
```

The table below describes the significant fields shown in the display. The content of the debugging messages depends on the commands that are subsequently entered at the router prompt.

Table 9: debug oer api Field Descriptions

Field	Description
OER api debugging is on	Shows that application interface debugging is enabled.
OER API	Displays an OER application interface message.

Related Commands

Command	Description
api provider	Registers an application interface provider with an OER master controller and enters OER master controller application interface provider configuration mode.
host-address	Configures information about a host device used by an application interface provider to communicate with an OER master controller.
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.
show oer api provider	Displays information about application interface providers registered with OER.

debug oer api client

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer api** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

Note Effective with Cisco IOS Release 12.4(15)T, the **debug oer api client** command is replaced by the **debug oer api** command. See the **debug oer api** command for more information.

To display Optimized Edge Routing (OER) application interface client debugging information for master controller and border router communication, use the **debug oer api client** command in privileged EXEC mode. To stop the display of OER application interface debugging information, use the **no** form of this command.

debug oer api client [detail]

no debug oer api client [detail]

Syntax Description

detail	(Optional) Displays detailed information.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(6)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.4(15)T	The **debug oer api client** command is replaced by the **debug oer api** command.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer api client** command can be entered on a master controller. This command is used to display messages about a configured OER application interface client. When the **detail** keyword is entered, the amount of detailed output to be displayed can utilize a considerable amount of system resources. Use the **detail**keyword with caution in a production network.

Cisco IOS Release 12.4(15)T

In Cisco IOS Release 12.4(15)T and later releases, the **debug oer api client** command is replaced by the **debug oer api** command. The **debug oer api client** command is currently supported for backwards compatibility, but support may be removed in a future Cisco IOS software release.

Examples

The following example enables the display of OER application interface client debugging messages:

```
Router# debug oer api client
API Client debugging enabled
```

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer border

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer border** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display general OER border router debugging information, use the **debug oer border**command in privileged EXEC mode. To stop the display of OER debugging information, use the **no** form of this command.

debug oer border

no debug oer border

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer border** command is entered on a border router. This command is used to display debugging information about the OER border process, controlled routes and monitored prefixes.

Examples

The following example displays general OER debugging information:

```
Router# debug oer border
*May  4 22:32:33.695: OER BR: Process Message, msg 4, ptr 33272128, value  140
*May  4 22:32:34.455: OER BR: Timer event, 0
```

The table below describes the significant fields shown in the display.

Table 10: debug oer border Field Descriptions

Field	Description
OER BR:	Indicates debugging information for OER Border process.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer border active-probe

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer border active-probe** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display debugging information for active probes configured on the local border router, use the **debug oer border active-probe**command in privileged EXEC mode. To stop the display of debug event information, use the **no** form of this command.

debug oer border active-probe

no debug oer border active-probe

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer border active-probe** command is entered on a master controller. This command is used to display the status and results of active probes that are configured on the local border router.

Examples

The following example enables the display of active-probe debug information on a border router:

```
Router# debug oer border active-probe

*May  4 23:47:45.633: OER BR ACTIVE PROBE: Attempting to retrieve Probe
Statistics.
      probeType = echo, probeTarget = 10.1.5.1, probeTargetPort = 0
      probeSource = Default, probeSourcePort = 0, probeNextHop = Default
      probeIfIndex = 13
*May  4 23:47:45.633: OER BR ACTIVE PROBE: Completed retrieving Probe
```

```
Statistics.
      probeType = echo, probeTarget = 10.1.5.1, probeTargetPort = 0
      probeSource = Default, probeSourcePort = 0, probeNextHop = 10.30.30.2
      probeIfIndex = 13, SAA index = 15
*May  4 23:47:45.633: OER BR ACTIVE PROBE: Completions 11, Sum of rtt 172,
Max rtt 36, Min rtt 12
*May  4 23:47:45.693: OER BR ACTIVE PROBE: Attempting to retrieve Probe
Statistics.
      probeType = echo, probeTarget = 10.1.4.1, probeTargetPort = 0
      probeSource = Default, probeSourcePort = 0, probeNextHop = Default
      probeIfIndex = 13
*May  4 23:47:45.693: OER BR ACTIVE PROBE: Completed retrieving Probe
Statistics.
      probeType = echo, probeTarget = 10.1.4.1, probeTargetPort = 0
      probeSource = Default, probeSourcePort = 0, probeNextHop = 10.30.30.2
      probeIfIndex = 13, SAA index = 14
```

The table below describes the significant fields shown in the display.

Table 11: debug oer border active-probe Field Descriptions

Field	Description
OER BR ACTIVE PROBE:	Indicates debugging information for OER active probes on a border router.
Statistics	The heading for OER active probe statistics.
probeType	The active probe type. The active probe types that can be displayed are ICMP, TCP, and UDP.
probeTarget	The target IP address of the active probe.
probeTargetPort	The target port of the active probe.
probeSource	The source IP address of the active probe. Default is displayed for a locally generated active probe.
probeSourcePort	The source port of the active probe.
probeNextHop	The next hop for the active probe.
probeIfIndex	The active probe source interface index.
SAA index	The IP SLAs collection index number.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer border learn

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer border learn** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display debugging information about learned prefixes on the local border router, use the **debug oer border learn**command in privileged EXEC mode. To stop the display of debug event information, use the **no** form of this command.

debug oer border learn [**top** *number*]

no debug oer border learn [**top** *number*]

Syntax Description

top *number*	(Optional) Displays debugging information about the top delay or top throughput prefixes. The number of top delay or throughput prefixes can be specified. The range of prefixes that can be specified is a number from 1 to 65535.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer border learn** command is entered on a border router. This command is used to display debugging information about prefixes learned on the local border router.

Examples

The following example enables the display of active-probe debug information on a border router:

```
Router# debug oer border learn

*May  4 22:51:31.971: OER BR LEARN: Reporting prefix 1: 10.1.5.0, throughput 201
*May  4 22:51:31.971: OER BR LEARN: Reporting 1 throughput learned prefixes
*May  4 22:51:31.971: OER BR LEARN: State change, new STOPPED, old STARTED, reason Stop
Learn
```

The table below describes the significant fields shown in the display.

Table 12: debug oer border learn Field Descriptions

Field	Description
OER BR LEARN:	Indicates debugging information for the OER border router learning process.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer border routes

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer border routes** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display debugging information for OER-controlled or monitored routes on the local border router, use the **debug oer border routes** command in privileged EXEC mode. To stop the display of debug event information, use the **no** form of this command.

debug oer border routes {**bgp**| **eigrp** [**detail**]| **piro** [**detail**]| **static**}

no debug oer border routes {**bgp**| **eigrp**| **static**| **piro**}

Syntax Description

bgp	Displays debugging information for BGP routes.
eigrp	Displays debugging information for EIGRP routes.
detail	(Optional) Displays detailed debugging information. This keyword applies only to EIGRP or PIRO routes.
static	Displays debugging information for static routes.
piro	Displays debugging information for Protocol Independent Route Optimization (PIRO) routes.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.4(24)T	This command was modified. The **piro** keyword was added to support the Protocol Independent Route Optimization (PIRO) feature.
15.0(1)M	This command was modified. The **eigrp** keyword was added to support EIGRP route control.

Release	Modification
12.2(33)SRE	This command was modified. The **eigrp** keyword was added to support EIGRP route control and the **piro** keyword was added to support the PIRO feature.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer border routes** command is entered on a border router. This command is used to display the debugging information about OER-controlled or monitored routes on the local border router.

In Cisco IOS Release 12.4(24)T, 12.2(33)SRE, and later releases, PIRO introduced the ability for OER to search for a parent route--an exact matching route, or a less specific route--in any IP Routing Information Base (RIB). If a parent route for the traffic class exists in the RIB, policy-based routing is used to control the prefix.

In Cisco IOS Release 15.0(1)M, 12.2(33)SRE, and later releases, EIGRP route control introduced the ability for OER to search for a parent route--an exact matching route, or a less specific route--in the EIGRP routing table. If a parent route for the traffic class exists in the EIGRP routing table, temporary EIGRP routes are injected and identified by adding a configurable extended community tag value.

Examples

The following example enables the display of active-probe debug information on a border router:

```
Router# debug oer border routes
 bgp
*May  4 22:35:53.239: OER BGP: Control exact prefix 10.1.5.0/24
*May  4 22:35:53.239: OER BGP: Walking the BGP table for 10.1.5.0/24
*May  4 22:35:53.239: OER BGP: Path for 10.1.5.0/24 is now under OER control
*May  4 22:35:53.239: OER BGP: Setting prefix 10.1.5.0/24 as OER net#
```

The table below describes the significant fields shown in the display.

Table 13: debug oer border routes Field Descriptions

Field	Description
OER BGP:	Indicates debugging information for OER-controlled BGP routes.
OER STATIC:	Indicates debugging information for OER-controlled Static routes. (Not displayed in the example output.)

The following example enables the display of detailed debugging information for PIRO routes and shows that the parent route for the prefix 10.1.1.0 is found in the RIB and a route map is created to control the application. Note that detailed border PBR debugging is also active. This example requires Cisco IOS Release 12.4(24)T, 12.2(33)SRE, or a later release.

```
Router# debug oer border routes piro detail
Feb 21 00:20:44.431: PIRO: Now calling ip_get_route
Feb 21 00:20:44.431: PFR PIRO: Parent lookup found parent 10.1.1.0, mask 255.255.255.0,
nexthop 10.1.1.0 for network 10.1.1.0/24
...
```

```
Feb 21 00:22:46.771: PFR PIRO: Parent lookup found parent 10.1.1.0, mask 255.255.255.0,
nexthop 10.1.1.0 for network 10.1.1.0/24
Feb 21 00:22:46.771: PFR PIRO: Control Route, 10.1.1.0/24, NH 0.0.0.0, IF Ethernet4/2
Feb 21 00:22:46.771: PIRO: Now calling ip_get_route
Feb 21 00:22:46.771: PIRO: Now calling ip_get_route
Feb 21 00:22:46.771: PFR PIRO: Parent lookup found parent 10.1.1.0, mask 255.255.255.0,
nexthop 10.1.1.0 for network 10.1.1.0/24
Feb 21 00:22:46.771: OER BR PBR(det): control app: 10.1.1.0/24, nh 0.0.0.0, if
Ethernet4/2,ip prot 256, dst opr 0, src opr 0, 0 0 0 0, src net 0.0.0.0/0, dscp 0/0
Feb 21 00:22:46.771: OER BR PBR(det): Create rmap 6468E488
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) T 10.1.1.0/24 EVENT Track
start
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) N 10.1.1.0/24 Adding track
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) N 10.1.1.0/24 QP Schedule
query
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) T 10.1.1.0/24 EVENT Query
found route
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) N 10.1.1.0/24 Adding route
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) R 10.1.1.0/24  d=0 p=0 ->
Updating
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) R 10.1.1.0/24  d=110 p=1 ->
Et4/2 40.40.40.2 40 Notifying
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: Adding to client notification queue
Feb 21 00:22:46.775: PfR-RIB RIB_RWATCH: (default:ipv4:base) W 10.1.1.0/24 c=0x15 Client
notified reachable
Feb 21 00:22:46.779: PFR PIRO: Route update rwinfo 680C8E14, network 10.1.1.0, mask_len 24
event Route Up
Feb 21 00:22:46.779: OER BR PBR(det): PIRO Path change notify for prefix:10.1.1.0,
masklen:24, reason:1
```

The table below describes the significant fields shown in the display.

Table 14: debug oer border routes Field Descriptions

Field	Description
PFR PIRO	Indicates debugging information for Performance Routing-controlled PIRO activities.
OER BR PBR	Indicates debugging information about policy-based routing activities on the border router.
PfR-RIB RIB_RWATCH	Indicates debugging information about RIB activities.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer border traceroute reporting

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer border traceroute reporting** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display debugging information for traceroute probes on the local border router, use the **debug oer border traceroute reporting** command in privileged EXEC mode. To stop the display of debug event information, use the **no** form of this command.

debug oer border traceroute reporting [detail]

no debug oer border traceroute reporting [detail]

Syntax Description

detail	(Optional) Displays detailed traceroute debug information.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(14)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer border traceroute reporting**command is entered on a border router. This command is used to display the debugging information about traceroute probes sourced on the local border router.

Examples

The following example enables the display of active-probe debug information on a border router:

```
Router# debug oer border traceroute reporting
```

```
May 19 03:46:23.807: OER BR TRACE(det): Received start message: msg1 458776,
msg2 1677787648, if index 19, host addr 100.1.2.1, flags 1, max ttl 30,
protocol 17, probe delay 0
May 19 03:46:26.811: OER BR TRACE(det): Result msg1 458776,
msg2 1677787648 num hops 30 sent May 19 03:47:20.919: OER BR TRACE(det):
Received start message: msg1 524312, msg2 1677787648, if index 2,
host addr 100.1.2.1, flags 1, max ttl 30, protocol 17, probe delay 0
May 19 03:47:23.923: OER BR TRACE(det): Result msg1 524312,
msg2 1677787648 num hops 3 sent
```

The table below describes the significant fields shown in the display.

Table 15: debug oer border traceroute reporting Field Descriptions

Field	Description
OER BR TRACE:	Indicates border router debugging information for traceroute probes.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer cc

Note Effective with Cisco IOS Release 15.0(1)SY, the **debug oer cc** command is hidden. Although this command is still available in Cisco IOS software, the CLI interactive Help does not display it if you attempt to view it by entering a question mark at the command line. This command will be completely removed in a future release.

To display OER communication control debugging information for master controller and border router communication, use the **debug oer cc** command in privileged EXEC mode. To stop the display of OER debugging information, use the **no** form of this command.

debug oer cc [detail]

no debug oer cc [detail]

Syntax Description

detail	(Optional) Displays detailed information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
15.0(1)SY	This command was modified. This command was hidden.

Usage Guidelines

The **debug oer cc** command can be entered on a master controller on a border router. This command is used to display messages exchanged between the master controller and the border router. These messages include control commands, configuration commands, and monitoring information. Enabling this command will cause very detailed output to be displayed and can utilize a considerable amount of system resources. This command should be enabled with caution in a production network.

Examples

The following example enables the display of OER communication control debugging messages:

```
Router# debug oer cc
*May  4 23:03:22.527: OER CC: ipflow prefix reset received: 10.1.5.0/24
```

The table below describes the significant fields shown in the display.

Table 16: debug oer cc Field Descriptions

Field	Description
OER CC:	Indicates debugging information for OER communication messages.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master border

To display debugging information for OER border router events on an OER master controller, use the **debug oer master border**command in privileged EXEC mode. To stop border router event debugging, use the **no** form of this command.

debug oer master border [*ip-address*]

no debug oer master border

Syntax Description

ip-address	(Optional) Specifies the IP address of a border router.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master border** command is entered on a master controller. The output displays information related to the events or updates from one or more border routers.

Examples

The following example shows the status of 2 border routers. Both routers are up and operating normally.

```
Router# debug oer master border
OER Master Border Router debugging is on
Router#
1d05h: OER MC BR 10.4.9.7: BR I/F update, status UP, line 1 index 1, tx bw 10000
0, rx bw 100000, time, tx ld 0, rx ld 0, rx rate 0 rx bytes 3496553, tx rate 0,
tx bytes 5016033
1d05h: OER MC BR 10.4.9.7: BR I/F update, status UP, line 1 index 2, tx bw 10000
0, rx bw 100000, time, tx ld 0, rx ld 0, rx rate 0 rx bytes 710149, tx rate 0, t
x bytes 1028907
1d05h: OER MC BR 10.4.9.6: BR I/F update, status UP, line 1 index 2, tx bw 10000
0, rx bw 100000, time, tx ld 0, rx ld 0, rx rate 0 rx bytes 743298, tx rate 0, t
x bytes 1027912
1d05h: OER MC BR 10.4.9.6: BR I/F update, status UP, line 1 index 1, tx bw 10000
0, rx bw 100000, time, tx ld 0, rx ld 0, rx rate 0 rx bytes 3491383, tx rate 0,
tx bytes 5013993
```

The table below describes the significant fields shown in the display.

Table 17: debug oer master border Field Descriptions

Field	Description
OER MC BR ip-address:	Indicates debugging information for a border router process. The ip-address identifies the border router.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master collector

To display data collection debugging information for OER monitored prefixes, use the **debug oer master collector**command in privileged EXEC mode. To disable the display of this debugging information, use the **no** form of this command.

debug oer master collector {**active-probes** [**detail [trace]**]| **netflow**}

no debug oer master collector {**active-probes** [**detail [trace]**]| **netflow**}

Syntax Description

active-probes	Displays aggregate active probe results for a given prefix on all border routers that are executing the active probe.
detail	(Optional) Displays the active probe results from each target for a given prefix on all border routers that are executing the active probe.
trace	(Optional) Displays aggregate active probe results and historical statistics for a given prefix on all border routers that are executing the active probe.
netflow	Displays information about the passive (NetFlow) measurements received by the master controller for prefixes monitored from the border router.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master collector** command is entered on a master controller. The output displays data collection information for monitored prefixes.

Examples

Examples

The following example displays aggregate active probe results for the 10.1.0.0/16 prefix on all border routers that are configured to execute this active probe:

```
Router# debug oer master collector active-probes

*May  4 22:34:58.221: OER MC APC: Probe Statistics Gathered for prefix 10.1.0.0/16 on all
  exits,notifying the PDP
*May  4 22:34:58.221: OER MC APC: Summary Exit Data (pfx 10.1.0.0/16, bdr 10.2.2.2, if 13,

  nxtHop Default):savg delay 13, lavg delay 14, sinits 25, scompletes 25
*May  4 22:34:58.221: OER MC APC: Summary Prefix Data: (pfx 10.1.0.0/16) sloss 0, lloss 0,

  sunreach 25, lunreach 25, savg raw delay 15, lavg raw delay 15, sinits 6561,
  scompletes 6536, linits 6561, lcompletes 6536
*May  4 22:34:58.221: OER MC APC: Active OOP check done
```

The table below describes the significant fields shown in the display.

Table 18: debug oer master collector active-probes Field Descriptions

Field	Description
OER MC APC:	Indicates debugging information for active probes from the r OER master collector.

Examples

The following example displays aggregate active probe results from each target for the 10.1.0.0/16 prefix on all border routers that are configured to execute this active probe:

```
Router# debug oer master collector active-probes detail

*May  4 22:36:21.945: OER MC APC: Rtrv Probe Stats: BR 10.2.2.2, Type echo,
  Tgt 10.1.1.1,TgtPt 0, Src Default, SrcPt 0, NxtHp Default, Ndx 13
*May  4 22:36:22.001: OER MC APC: Remote stats received: BR 10.2.2.2, Type
  echo, Tgt 10.15.1, TgtPt 0, Src Default, SrcPt 0, NxtHp Default, Ndx 13
*May  4 22:36:22.313: OER MC APC: Perf data point (pfx 10.1.0.0/16, bdr
  10.2.2.2, if 13, xtHop Default):  avg delay 20, loss 0, unreach 0,
  initiations 2, completions 2, delay sum40, ldelay max 20, ldelay min 12
*May  4 22:36:22.313: OER MC APC: Perf data point (pfx 10.1.0.0/16, bdr
  10.2.2.2, if 13, xtHop Default):  avg delay 20, loss 0, unreach 0,
  initiations 2, completions 2, delay sum40, ldelay max 20, ldelay min 12
*May  4 22:36:22.313: OER MC APC: Probe Statistics Gathered for prefix
  10.1.0.0/16 on al exits, notifying the PDP
*May  4 22:36:22.313: OER MC APC: Active OOP check done
```

The table below describes the significant fields shown in the display.

Table 19: debug oer master collector active-probes detail Field Descriptions

Field	Description
OER MC APC:	Indicates debugging information for active probes from the r OER master collector.

Examples

The following example displays aggregate active probe results and historical statistics from each target for the 10.1.0.0/16 prefix on all border routers that are configured to execute this active probe:

```
Router# debug oer master collector active-probes detail trace

*May  4 22:40:33.845: OER MC APC: Rtrv Probe Stats: BR 10.2.2.2, Type echo,
  Tgt 10.1.5.1, TgtPt 0, Src Default, SrcPt 0, NxtHp Default, Ndx 13
*May  4 22:40:33.885: OER MC APC: Remote stats received: BR 10.2.2.2, Type
  echo, Tgt 10.1.5.1, TgtPt 0, Src Default, SrcPt 0, NxtHp Default, Ndx 13
*May  4 22:40:34.197: OER MC APC: Remote stats received: BR 10.2.2.2, Type
  echo, Tgt 10.1.2.1, TgtPt 0, Src Default, SrcPt 0, NxtHp Default, Ndx 13
*May  4 22:40:34.197: OER MC APC: Updating Probe (Type echo Tgt 10.1.2.1
  TgtPt 0) Total Completes 1306, Total Attempts 1318
*May  4 22:40:34.197: OER MC APC: All stats gathered for pfx 10.1.0.0/16
  Accumulating Stats
*May  4 22:40:34.197: OER MC APC: Updating Curr Exit Ref (pfx 10.1.0.0/16,
  bdr 10.2.2.2, if 13, nxtHop Default) savg delay 17, lavg delay 14, savg loss
  0, lavg loss 0, savg unreach 0, lavg unreach 0
*May  4 22:40:34.197: OER MC APC: Probe Statistics Gathered for prefix
  10.1.0.0/16 on all exits, notifying the PDP
*May  4 22:40:34.197: OER MC APC: Active OOP check done
```

The table below describes the significant fields shown in the display.

Table 20: debug oer master collector active-probes detail trace Field Descriptions

Field	Description
OER MC APC:	Indicates debugging information for active probes from the r OER master collector.

Examples

The following example displays passive monitoring results for the 10.1.5.0/24 prefix:

```
Router# debug oer master collector netflow

*May  4 22:31:45.739: OER MC NFC: Rcvd  egress update from BR 10.1.1.2
  prefix 10.1.5.0/24  Interval 75688  delay_sum 0  samples 0  bytes 20362  pkts 505
  flows 359  pktloss 1 unreach 0
*May  4 22:31:45.739: OER MC NFC: Updating exit_ref; BR 10.1.1.2 i/f Et1/0,
  s_avg_delay 655, l_avg_delay 655, s_avg_pkt_loss 328, l_avg_pkt_loss 328,
  s_avg_flow_unreach 513, l_avg_flow_unreach 513
*May  4 22:32:07.007: OER MC NFC: Rcvd ingress update from BR 10.1.1.3
  prefix 10.1.5.0/24  Interval 75172  delay_sum 42328  samples 77  bytes 22040
  pkts 551  flows 310  pktloss 0 unreach 0
```

The table below describes the significant fields shown in the display.

Table 21: debug oer master collector netflow Field Descriptions

Field	Description
OER MC NFC:	Indicates debugging information for the OER master collector from passive monitoring (NetFlow).

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master cost-minimization

To display debugging information for cost-based optimization policies, use the **debug oer master cost-minimization** command in privileged EXEC mode. To disable the display of this debugging information, use the **no** form of this command.

debug oer master cost-minimization [detail]

no debug oer master cost-minimization [detail]

Syntax Description

detail	(Optional) Displays detailed information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master cost-minimization** command is entered on a master controller. The output displays debugging information for cost-minimization policies.

Examples

The following example displays detailed cost optimization policy debug information:

```
Router# debug oer master cost-minimization detail
OER Master cost-minimization Detail debugging is on
*May 14 00:38:48.839: OER MC COST: Momentary target utilization for exit 10.1.1.2 i/f
Ethernet1/0 nickname ISP1 is 7500 kbps, time_left 52889 secs, cumulative 16 kb, rollup
period 84000 secs, rollup target 6000 kbps, bw_capacity 10000 kbps
*May 14 00:38:48.839: OER MC COST: Cost OOP check for border 10.1.1.2, current util: 0
target util: 7500 kbps
*May 14 00:39:00.199: OER MC COST: ISP1 calc separate rollup ended at 55 ingress Kbps
*May 14 00:39:00.199: OER MC COST: ISP1 calc separate rollup ended at 55 egress bytes
*May 14 00:39:00.199: OER MC COST: Target utilization for nickname ISP1 set to 6000,
rollups elapsed 4, rollups left 24
*May 14 00:39:00.271: OER MC COST: Momentary target utilization for exit 10.1.1.2 i/f
Ethernet1/0 nickname ISP1 is 7500 kbps, time_left 52878 secs, cumulative 0 kb, rollup
period 84000 secs, rollup target 6000 kbps, bw_capacity 10000 kbps
*May 14 00:39:00.271: OER MC COST: Cost OOP check for border 10.1.1.2, current util: 0
target util: 7500 kbps
```

The table below describes the significant fields shown in the display.

Table 22: debug oer master cost-minimization detail Field Descriptions

Field	Description
OER MC COST:	Indicates debugging information for cost-based optimization on the master controller.

Related Commands

Command	Description
cost-minimization	Configures cost-based optimization policies on a master controller.
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.
show oer master cost-minimization	Displays the status of cost-based optimization policies.

debug oer master exit

To display debug event information for OER managed exits, use the **debug oer master exit**command in privileged EXEC mode. To stop the display of debug event information, use the **no** form of this command.

debug oer master exit [detail]

no debug oer master exit [detail]

Syntax Description

detail	Displays detailed OER managed exit information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master exit** command is entered on a master controller. This command is used to display debugging information for master controller exit selection processes.

Examples

The following example shows output form the **debug oer master exit** command, entered with the **detail**keyword:

```
Router# debug oer master exit
 detail
*May  4 11:26:51.539: OER MC EXIT: 10.1.1.1, intf Fa4/0 INPOLICY
*May  4 11:26:52.195: OER MC EXIT: 10.2.2.3, intf Se2/0 INPOLICY
*May  4 11:26:55.515: OER MC EXIT: 10.1.1.2, intf Se5/0 INPOLICY
*May  4 11:29:14.987: OER MC EXIT: 7 kbps should be moved from 10.1.1.1, intf Fa4/0
*May  4 11:29:35.467: OER MC EXIT: 10.1.1.1, intf Fa4/0 in holddown state so skip OOP check

*May  4 11:29:35.831: OER MC EXIT: 10.2.2.3, intf Se2/0 in holddown state so skip OOP check

*May  4 11:29:39.455: OER MC EXIT: 10.1.1.2, intf Se5/0 in holddown state so skip OOP check
```

The table below describes the significant fields shown in the display.

Table 23: debug oer master exit detail Field Descriptions

Field	Description
OER MC EXIT:	Indicates OER master controller exit event.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master learn

To display debug information for OER master controller learning events, use the **debug oer master learn** command in privileged EXEC mode. To stop the display of debug information, use the **no** form of this command.

debug oer master learn

no debug oer master learn

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master learn** command is entered on a master controller. This command is used to display debugging information for master controller learning events.

Examples

The following example shows output from the **debug oer master learn** command. The output an shows OER Top Talker debug events. The master controller is enabling prefix learning for new border router process:

```
Router# debug oer master learn
06:13:43: OER MC LEARN: Enable type 3, state 0
06:13:43: OER MC LEARN: OER TTC: State change, new RETRY, old DISABLED, reason TT start
06:13:43: OER MC LEARN: OER TTC: State change, new RETRY, old DISABLED, reason TT start
request
06:13:43: OER MC LEARN: OER TTC: State change, new RETRY, old DISABLED, reason T
T start request
06:14:13: OER MC LEARN: TTC Retry timer expired
06:14:13: OER MC LEARN: OER TTC: State change, new STARTED, old RETRY, reason At
 least one BR started
06:14:13: %OER_MC-5-NOTICE: Prefix Learning STARTED
06:14:13: OER MC LEARN: MC received BR TT status as enabled
06:14:13: OER MC LEARN: MC received BR TT status as enabled
06:19:14: OER MC LEARN: OER TTC: State change, new WRITING DATA, old STARTED, reason
Updating DB
06:19:14: OER MC LEARN: OER TTC: State change, new SLEEP, old WRITING DATA, reason
Sleep state
```

The table below describes the significant fields shown in the display.

Table 24: debug oer master learn Field Descriptions

Field	Description
OER MC LEARN:	Indicates OER master controller learning events.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master prefix

To display debug events related to prefix processing on an OER master controller, use the **debug oer master prefix**command in privileged EXEC mode. To disable the display of debug information, use the **no** form of this command.

debug oer master prefix [*prefix*| **appl**] **[detail]**

no debug oer master prefix [*prefix*| **appl**] **[detail]**

Syntax Description

prefix	(Optional) Specifies a single prefix or prefix range. The prefix address and mask are entered with this argument.
appl	(Optional) Displays information about prefixes used by applications monitored and controlled by an OER master controller.
detail	(Optional) Displays detailed OER prefix processing information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master prefix** command is entered on a master controller. This command displays debugging information related to prefix monitoring and processing.

Examples

The following example shows the master controller searching for the target of an active probe after the target has become unreachable.

```
Router# debug oer master prefix

OER Master Prefix debugging is on
06:01:28: OER MC PFX 10.4.9.0/24: APC last target deleted for prefix, no targets
 left assigned and running
06:01:38: OER MC PFX 10.4.9.0/24: APC Attempting to probe all exits
06:02:59: OER MC PFX 10.4.9.0/24: APC last target deleted for prefix, no targets
 left assigned and running
06:03:08: OER MC PFX 10.4.9.0/24: APC Attempting to probe all exits
```

```
06:04:29: OER MC PFX 10.4.9.0/24: APC last target deleted for prefix, no targets
 left assigned and running
06:04:39: OER MC PFX 10.4.9.0/24: APC Attempting to probe all exits
06:05:59: OER MC PFX 10.4.9.0/24: APC last target deleted for prefix, no targets
 left assigned and running
06:06:09: OER MC PFX 10.4.9.0/24: APC Attempting to probe all exits
```

The table below describes the significant fields shown in the display.

Table 25: debug oer master prefix Field Descriptions

Field	Description
OER MC PFX ip-address:	Indicates debugging information for OER monitored prefixes. The ip-address identifies the prefix.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master prefix-list

To display debug events related to prefix-list processing on an OER master controller, use the **debug oer master prefix-list**command in privileged EXEC mode. To disable the display of debug information, use the **no** form of this command.

debug oer master prefix-list *list-name* **[detail]**

no debug oer master prefix-list *list-name*

Syntax Description

list-name	Specifies a single prefix or prefix range. The prefix address and mask are entered with this argument.
detail	(Optional) Displays detailed OER prefix-list processing information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master prefix-list**command is entered on a master controller. This command displays debugging information related to prefix-list processing.

Examples

The following example shows output from the **debug oer master prefix-list**command.

```
Router# debug oer master prefix-list

23:02:16.283: OER MC PFX 10.1.5.0/24: Check PASS REL loss: loss 0, policy 10%, notify TRUE
23:02:16.283: OER MC PFX 10.1.5.0/24: Passive REL loss in-policy
23:02:16.283: OER MC PFX 10.1.5.0/24: Check PASS REL delay: delay 124, policy 50%, notify
TRUE
23:02:16.283: OER MC PFX 10.1.5.0/24: Passive REL delay in policy
23:02:16.283: OER MC PFX 10.1.5.0/24: Prefix not OOP
23:02:16.283: OER MC PFX 10.1.5.0/24: Check PASS REL unreachable: unreachable 0, policy
50%, notify TRUE
23:02:16.283: OER MC PFX 10.1.5.0/24: Passive REL unreachable in-policy
23:02:16.283: OER MC PFX 10.1.5.0/24: Check PASS REL loss: loss 0, policy 10%, notify TRUE
23:02:16.283: OER MC PFX 10.1.5.0/24: Passive REL loss in policy
```

The table below describes the significant fields shown in the display.

Table 26: debug oer master prefix-list Field Descriptions

Field	Description
OER MC PFX ip-address:	Indicates debugging information for OER monitored prefixes. The ip-address identifies the prefix.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master process

To display debug information about the OER master controller process, use the **debug oer master process** command in privileged EXEC mode. To stop displaying debug information, use the **no** form of this command.

debug oer master process

no debug oer master process

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

The **debug oer master process** command is entered on a master controller.

Examples

The following sample debug output for a master controller process:

```
Router# debug oer master process
01:12:00: OER MC PROCESS: Main msg type 15, ptr 0, value 0
```

The table below describes the significant fields shown in the display.

Table 27: debug oer master process Field Descriptions

Field	Description
OER MC PROCESS:	Indicates a master controller master process debugging message.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug oer master traceroute reporting

To display debug information about traceroute probes, use the **debug oer master traceroute reporting** command in privileged EXEC mode. To stop displaying debug information, use the **no** form of this command.

debug oer master traceroute reporting [detail]

no debug oer master traceroute reporting [detail]

Syntax Description

detail	(Optional) Displays detailed information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.

Usage Guidelines

The **debug oer master traceroute reporting** command is entered on a master controller. This command is used to display traceroute events on a master controller.

Examples

The following sample debug output for a master controller process:

```
Router# debug oer master traceroute reporting detail
*May 12 18:55:14.239: OER MC TRACE: sent start message msg1 327704, msg2 167838976,
if index 2, host add 10.1.5.2, flags 1, max ttl 30, protocol 17
*May 12 18:55:16.003: OER MC TRACE: sent start message msg1 393240, msg2 167838976,
if index 2, host add 10.1.5.2, flags 1, max ttl 30, protocol 17
master#
*May 12 18:55:17.303: OER MC TRACE: Received result: msg_id1 327704, prefix 10.1.5.0/24,
hops 4, flags 1
*May 12 18:55:19.059: OER MC TRACE: Received result: msg_id1 393240, prefix 10.1.5.0/24,
hops 4, flags 1
```

The table below describes the significant fields shown in the display.

Table 28: debug oer master traceroute reporting detail Field Descriptions

Field	Description
OER MC PROCESS:	Indicates master controller debugging information for traceroute probes.

Related Commands

Command	Description
oer	Enables an OER process and configures a router as an OER border router or as an OER master controller.

debug ospfv3

To display debugging information for Open Shortest Path First version 3 (OSPF) for IPv4 and IPv6, use the **debug ospfv3**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ospfv3 [*process-id*] [*address-family*] [**adj| ipsec| database-timer| flood| hello| lsa-generation| retransmission**]

no debug ospfv3 [*process-id*] [*address-family*] [**adj| ipsec| database-timer| flood| hello| lsa-generation| retransmission**]

Syntax Description

process-id	(Optional) Internal identification. The number used here is the number assigned administratively when enabling the OSPFv3 routing process and can be a value from 1 through 65535.
address-family	(Optional) Enter **ipv6** for the IPv6 address family or **ipv4** for the IPv4 address family.
adj	(Optional) Displays adjacency information.
ipsec	(Optional) Displays the interaction between OSPFv3 and IPSec, including creation and removal of policy definitions.
database-timer	(Optional) Displays database-timer information.
flood	(Optional) Displays flooding information.
hello	(Optional) Displays hello packet information.
l2api	(Optional) Enables layer 2 and layer 3 application program interface (API) debugging.
lsa-generation	(Optional) Displays link-state advertisement (LSA) generation information for all LSA types.
retransmission	(Optional) Displays retransmission information.

Command Default

Debugging of OSPFv3 is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1(3)S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was integrated into Cisco IOS XE Release 3.4S.
15.2(1)T	This command was integrated into Cisco IOS Release 15.2(1)T.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays adjacency information for OSPFv3:

```
Device# debug ospfv3 adj
```

debug ospfv3 database-timer rate-limit

To display debugging information about the current wait-time used for shortest path first (SPF) scheduling, use the **debug ospfv3 database-timer rate-limit**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ospfv3 [*process-id*] [*address-family*] **database-timer rate-limit** [*acl-number*]

no debug ospfv3 [*process-id*] [*address-family*] **database-timer rate-limit**

Syntax Description

process-id	(Optional) Internal identification. The number used here is the number assigned administratively when enabling the OSPFv3 routing process and can be a value from 1 through 65535.
address-family	(Optional) Enter **ipv6** for the IPv6 address family or **ipv4** for the IPv4 address family.
acl-number	(Optional) Access list number.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(3)S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was integrated into Cisco IOS XE Release 3.4S.
15.2(1)T	This command was integrated into Cisco IOS Release 15.2(1)T.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example shows how to turn on debugging for SPF scheduling in OSPFv3 process 1:

```
Device# debug ospfv3 1 database-timer rate-limit
```

debug ospfv3 events

To display information on Open Shortest Path First version 3 (OSPFv3)-related events, such as designated router selection and shortest path first (SPF) calculation, use the **debug ospfv3 events** command in privileged EXEC com mand. To disable debugging output, use the **no** form of this command.

debug ospfv3 [*process-id*] [*address-family*] **events**

no debug ipv6 ospfv3 [*process-id*] [*address-family*] **events**

Syntax Description

process-id	(Optional) Internal identification. The number used here is the number assigned administratively when enabling the OSPFv3 routing process and can be a value from 1 through 65535.
address-family	(Optional) Enter **ipv6** for the IPv6 address family or **ipv4** for the IPv4 address family.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1(3)S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was integrated into Cisco IOS XE Release 3.4S.
15.2(1)T	This command was integrated into Cisco IOS Release 15.2(1)T.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays information on OSPFv3-related events:

```
Device#
debug ospfv3 events
```

debug ospfv3 lsdb

To display database modifications for Open Shortest Path First version 3 (OSPFv3), use the **debug ospfv3 lsdb** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ospfv3 [*process-id*] [*address-family*] **lsdb**

no debug ospfv3 [*process-id*] [*address-family*] **lsdb**

Syntax Description

process-id	(Optional) Internal identification. The number used here is the number assigned administratively when enabling the OSPFv3 routing process and can be a value from 1 through 65535.
address-family	(Optional) Enter **ipv6** for the IPv6 address family or **ipv4** for the IPv4 address family.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1(3)S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was integrated into Cisco IOS XE Release 3.4S.
15.2(1)T	This command was integrated into Cisco IOS Release 15.2(1)T.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays database modification information for OSPFv3:

```
Device# debug ospfv3 lsdb
```

debug ospfv3 packet

To display information about each Open Shortest Path First version 3 (OSPFv3) packet received, use the **debug ospfv3 packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ospfv3 [*process-id*] [*address-family*] **packet**

no debug ospfv3 [*process-id*] [*address-family*] **packet**

Syntax Description

process-id	(Optional) Internal identification. The number used here is the number assigned administratively when enabling the OSPFv3 routing process and can be a value from 1 through 65535.
address-family	(Optional) Enter **ipv6** for the IPv6 address family or **ipv4** for the IPv4 address family.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1(3)S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was integrated into Cisco IOS XE Release 3.4S.
15.2(1)T	This command was integrated into Cisco IOS Release 15.2(1)T.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

Consult Cisco technical support before using this command.

Examples

The following example displays information about each OSPFv3 packet received:

```
Router# debug ospfv3 packet
```

debug ospfv3 spf statistic

To display statistical information while running the shortest path first (SPF) algorithm, use the **debug ospfv3 spf statistic**command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

debug ospfv3 [*address-family*] **spf statistic**

no debug ospfv3 [*address-family*] **spf statistic**

Syntax Description

address-family	(Optional) Enter **ipv6** for the IPv6 address family or **ipv4** for the IPv4 address family.

Command Modes

Privileged EXEC

Command History

Release	Modification
15.1(3)S	This command was introduced.
Cisco IOS XE Release 3.4S	This command was integrated into Cisco IOS XE Release 3.4S.
15.2(1)T	This command was integrated into Cisco IOS Release 15.2(1)T.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Usage Guidelines

The **debug ospfv3 spf statistic** command displays the SPF calculation times in milliseconds, the node count, and a time stamp. Consult Cisco technical support before using this command.

Examples

The following example displays statistical information while running the SPF algorithm:

```
Router# debug ospfv3 spf statistics
```

Related Commands

Command	Description
debug ospfv3	Displays debugging information for the OSPFv3 feature.
debug ospfv3 events	Displays information on OSPFv3-related events.
debug ospfv3 packet	Displays information about each OSPFv3 packet received.

debug otv

To enable debugging of Overlay Transport Virtualization (OTV) and Intermediate System-to-Intermediate System (IS-IS) activities, use the **debug otv** command in privileged EXEC mode. To disable the debugging, use the **no** form of this command.

debug otv [**adjacency**| **all**| **arp-nd**| **database**| **error**| **evc**| **event**| **ha**| **igp**| **isis**| **l2rib**| **l2rtgvpn**| **misc**| **multicast**| **overlay**| **packet**| **pim**| **state**| **tunnel**| **ui**]

no debug otv [**adjacency**| **all**| **arp-nd**| **database**| **error**| **evc**| **event**| **ha**| **igp**| **isis**| **l2rib**| **l2rtgvpn**| **misc**| **multicast**| **overlay**| **packet**| **pim**| **state**| **tunnel**| **ui**]

Syntax Description

adjacency	Enables logging of adjacency-related events.
all	Enables logging of all debugging messages.
arp-nd	Enables logging of OTV database-related operations.
database	Enables logging of the Address Routing Protocol (ARP) suppression feature.
error	Enables logging of error debug messages.
evc	Enables logging of Ethernet Virtual Connections (EVC) interactions.
event	Enables logging of the event dispatcher.
ha	Enables logging of high availability (HA) events.
igp	Enables logging of OTV IS-IS events.
isis	Enables logging of IS-IS information.
l2rib	Enables logging of Layer 2 Routing Information Base (L2RIB) interactions.
l2rtgvpn	Enables logging of Layer 2 routing VPN manager.
misc	Enables logging of miscellaneous OTV debug messages.
multicast	Enables logging of multicast-related events.
overlay	Enables logging of overlay interface events.
packet	Enables logging of OTV packet forwarding activities.
pim	Enables logging of Protocol Independent Multicast (PIM) messages.
state	Enables logging of OTV state change events.

tunnel	Enables logging of tunnel interactions.
ui	Enables logging of OTV user interface (UI) events.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Examples

The following example shows how to enable logging of OTV activities:

```
Router# debug otv all

OTV APP all debugging is on
Router#
*Oct 27 13:53:45.155: OTV-APP-PKT-RX: Received packet on Overlay1 L3 dest 224.1.1.2 source
 209.165.201.20, L2 dest 0100.0cdf.dfdf source 0023.33cc.ebbc, linktype 25
*Oct 27 13:53:46.241: OTV-APP-PKT-RX: Received packet on Overlay1 L3 dest 224.1.1.2 source
 209.165.201.20, L2 dest 0100.0cdf.dfdf source 0015.17b9.c479, linktype 25
*Oct 27 13:53:46.824: OTV-APP-PKT-RX: Received packet on Overlay1 L3 dest 224.1.1.2 source
 209.165.201.20, L2 dest 0100.0cdf.dfdf source 0023.33cc.ebbc, linktype 25
*Oct 27 13:53:49.166: OTV-APP-PKT-RX: Received packet on Overlay1 L3 dest 224.1.1.2 source
 209.165.201.20, L2 dest 0100.0cdf.dfdf source 0015.17b9.c479, linktype 25
*Oct 27 13:53:50.055: OTV-APP-PKT-RX: Received packet on Overlay1 L3 dest 224.1.1.2 source
 209.165.201.20, L2 dest 0100.0cdf.dfdf source 0023.33cc.ebbc, linktype 25
*Oct 27 13:53:50.085: OTV-APP-PKT-TX: Overlay 1 process switching packet to 224.1.1.2
```

Related Commands

Command	Description
interface overlay	Creates an OTV overlay interface.
show otv	Displays OTV information.

debug otv isis

To enable debugging of Overlay Transport Virtualization (OTV) Intermediate System-to-Intermediate System (IS-IS) activities, use the **debug otv isis** command in privileged EXEC mode. To disable the debugging, use the **no** form of this command.

debug otv isis [**overlay** *interface*][**site**]{**adj-packets** *interface-type interface-number*| **aed**| **authentication information**| **checksum-errors**| **common event**| **local-updates**| **nsf**[**cisco**| **detail**| **ietf**]| **protocol-errors**| **rib**[**redistribution**][**mac**| **multicast[mapping]**]| **snp-packets**| **update-packets**| **vlan-database**}

no debug otv isis [**overlay** *interface*][**site**]{**adj-packets** *interface-type interface-number*| **aed**| **authentication information**| **checksum-errors**| **common event**| **local-updates**| **nsf**[**cisco**| **detail**| **ietf**]| **protocol-errors**| **rib**[**redistribution**][**mac**| **multicast[mapping]**]| **snp-packets**| **update-packets**| **vlan-database**}

Syntax Description

overlay *overlay-interface*	(Optional) Enables debugging of the specified overlay interface. The range is from 0 to 512.
site	(Optional) Enables logging of the IS-IS Layer 2 site process.
adj-packets	Enables logging of adjacency packets.
interface-type	Type of interface.
interface-number	Port, connector, or interface card number. The numbers are assigned at the factory at the time of installation or when added to a system; they can be displayed with the **show interfaces** command.
aed	Enables logging of authoritative edge device (AED) information.
authentication information	Enables logging of packet authentication information.
checksum-errors	Enables logging of link-state packet (LSP) checksum errors.
common event	Enables logging of common IS-IS events.
local-updates	Enables logging of local update packets.
nsf	Enables logging of IS-IS nonstop forwarding (NSF) information.
cisco	(Optional) Enables logging of only Cisco NSF information.
detail	(Optional) Enables logging of detailed NSF information.
ietf	(Optional) Enables logging of only IETF NSF information.
protocol-errors	Enables logging of LSP protocol errors.
rib	Enables logging of local Routing Information Base (RIB) events.

redistribution	(Optional) Enables logging of redistribution RIB events.
mac	(Optional) Enables logging of Layer 2 MAC RIB events.
multicast	(Optional) Enables logging of Layer 2 multicast RIB events.
mapping	(Optional) Enables logging of Layer 2 multicast mapping RIB events.
snp-packets	Enables logging of complete sequence number protocol data units (PDUs) (CSNP)/partial sequence number PDUs (PSNPs).
update-packets	Enables logging of update packets.
vlan-database	Enables logging of information about the VLAN database.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Examples

The following is sample output from the **debug otv isis aed** command:

```
Router# debug otv isis aed
```

```
*Nov 11 22:16:21.309: ISIS-AEDInfo (Overlay1): Neighbor AABB.CC00.0300 not found in osn
list
*Nov 11 22:16:21.309: ISIS-AEDInfo (Overlay1): Neighbor AABB.CC00.0300 not found in osn
list
*Nov 11 22:16:21.309: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0300 in
site 0000.0000.0013
*Nov 11 22:16:21.309: ISIS-AEDInfo (Overlay1): Local AED enabled for isis
*Nov 11 22:16:21.309: ISIS-AEDInfo (Overlay1): adding neighbor AABB.CC00.0100 to osn list
*Nov 11 22:16:21.309: ISIS-AEDInfo (Overlay1): Adding site neighbor AABB.CC00.0100 to osn
*Nov 11 22:16:22.309: ISIS-AEDInfo (Overlay1): Neighbor AABB.CC00.0300 not found in osn
list
*Nov 11 22:16:43.182: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0300 in
site 0000.0000.0013
*Nov 11 22:16:43.182: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0300 in
site 0000.0000.0013
*Nov 11 22:16:43.182: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0300 in
site 0000.0000.0013
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): adding neighbor AABB.CC00.0200 to osn list
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): Adding site neighbor AABB.CC00.0200 to osn
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0200 in
site 0000.0000.0000
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): Removing overlay/all neighbor AABB.CC00.0200
 from osn
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): removed neighbor AABB.CC00.0200 from osn
list
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0300 in
site 0000.0000.0013
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): Found overlay neighbor AABB.CC00.0300 in
site 0000.0000.0013
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): adding neighbor AABB.CC00.0200 to osn list
```

```
*Nov 11 22:16:45.327: ISIS-AEDInfo (Overlay1): Adding overlay neighbor AABB.CC00.0200 to
osn
*Nov 11 22:16:48.144: ISIS-AEDInfo (Overlay1): Neighbor AABB.CC00.0200 already in osn list
*Nov 11 22:16:48.144: ISIS-AEDInfo (Overlay1): Adding site neighbor AABB.CC00.0200 to osn
```

Related Commands

Command	Description
show otv isis	Displays the IS-IS status and configuration.

debug packet

To display per-packet debugging output, use the **debugpacket**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug packet [**interface** *number* [**vcd** *vcd-number*]| **vc** *vpi*/*vci*| *vc-name*]

no debug packet [**interface** *number* [**vcd** *vcd-number*]| **vc** *vpi*/*vci*| *vc-name*]

Syntax Description

interface *number*	(Optional) interface or subinterface number.
vcd *vcd-number*	(Optional) Number of the virtual circuit designator (VCD).
vc *vpi* / *vci*	(Optional) Virtual path identifier (VPI) and virtual channel identifier (VCI) numbers of the VC.
vc-name	(Optional) Name of the PVC or SVC.

Command Default

Debugging for packets is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
9.21	This command was introduced.
12.2(13)T	Support for Apollo Domain and Banyan VINES was removed.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

The **debugpacket** command displays all process-level packets for both outbound and inbound packets. This command is useful for determining whether packets are being received and sent correctly. The output reports information online when a packet is received or a transmission is attempted.

For sent packets, the information is displayed only after the protocol data unit (PDU) is entirely encapsulated and a next hop VC is found. If information is not displayed, the address translation probably failed during encapsulation. When a next hop VC is found, the packet is displayed exactly as it will be presented on the wire. Having a display indicates that the packets are properly encapsulated for transmission.

For received packets, information is displayed for all incoming frames. The display can show whether the sending station properly encapsulates the frames. Because all incoming frames are displayed, this information is useful when performing back-to-back testing and corrupted frames cannot be dropped by an intermediary switch.

The **debugpacket** command also displays the initial bytes of the actual PDU in hexadecimal. This information can be decoded only by qualified support or engineering personnel.

Caution Because the **debugpacket** command generates a substantial amount of output for every packet processed, use it only when traffic on the network is low so other activity on the system is not adversely affected.

Examples

The following is sample output from the **debugpacket** command:

```
Router# debug packet
2/0.5(I): VCD:0x9 VCI:0x23 Type:0x0 SAP:AAAA CTL:03 OUI:000000 TYPE:0800 Length0x70
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFF 0105
```

The following table describes the significant fields shown in the display.

Table 29: debug packet Field Descriptions

Field	Description
2/0.5	Indicates the subinterface that generated this packet.
(I)	Indicates a receive packet. (O) indicates an output packet.
VCD: 0x*n*	Indicates the virtual circuit associated with this packet, where *n* is some value.
DM: 0x*nnnn*	Indicates the descriptor mode bits on output only, where *nnnn* is a hexadecimal value.
TYPE:*n*	Displays the encapsulation type for this packet.
Length:*n*	Displays the total length of the packet including the headers.

The following two lines of output are the binary data, which are the contents of the protocol data unit (PDU) before encapsulation:

```
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFF 0105
```

The following is sample output from the **debugpacket** command:

```
Router# debug packet
Ethernet0: Unknown ARPA, src 0000.0c00.6fa4, dst ffff.ffff.ffff, type 0x0a0
data 00000c00f23a00000c00ab45, len 60
Serial3: Unknown HDLC, size 64, type 0xaaaa, flags 0x0F00
Serial2: Unknown PPP, size 128
Serial7: Unknown FRAME-RELAY, size 174, type 0x5865, DLCI 7a
Serial0: compressed TCP/IP packet dropped
```

The following table describes the significant fields shown in the display.

Table 30: debug packet Field Descriptions

Field	Description
Ethernet0	Name of the Ethernet interface that received the packet.
Unknown	Network could not classify this packet. Examples include packets with unknown link types.
ARPA	Packet uses ARPA-style encapsulation. Possible encapsulation styles vary depending on the media command mode (MCM) and encapsulation style. **Ethernet (MCM)** --*EncapsulationStyle:* • ARP • ETHERTALK • ISO1 • ISO3 • LLC2 • NOVELL-ETHER • SNAP
	FDDI (MCM) --*Encapsulation Style:* • ISO1 • ISO3 • LLC2 • SNAP
	Frame Relay --*EncapsulationStyle:* • BRIDGE • FRAME-RELAY

Field	Description
ARPA (continued)	**Serial (MCM)** --*EncapsulationStyle:* • BFEX25 • BRIDGE • DDN-X25 • DDNX25-DCE • ETHERTALK • FRAME-RELAY • HDLC • HDH • LAPB • LAPBDCE • MULTI-LAPB • PPP • SDLC-PRIMARY • SDLC-SECONDARY • SLIP • SMDS • STUN • X25 • X25-DCE
	Token Ring (MCM) --*EncapsulationStyle:* • 3COM-TR • ISO1 • ISO3 • MAC • LLC2 • NOVELL-TR • SNAP • VINES-TR
src 0000.0c00.6fa4	MAC address of the node generating the packet.

Field	Description
dst.ffff.ffff.ffff	MAC address of the destination node for the packet.
type 0x0a0	Packet type.
data...	First 12 bytes of the datagram following the MAC header.
len 60	Length of the message (in bytes) that the interface received from the wire.
size 64	Length of the message (in bytes) that the interface received from the wire. Equivalent to the len field.
flags 0x0F00	HDLC or PP flags field.
DLCI 7a	The DLCI number on Frame Relay.
compressed TCP/IP packet dropped	TCP header compression is enabled on an interface and the packet is not HDLC or X25.

debug packet-capture

To enable packet capture debugs, use the **debug packet-capture** command in privileged EXEC mode. To disable debugging packet capture, use the **no** form of this command.

debug packet-capture

no debug packet-capture

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(20)T	This command was introduced.
12.2(33)SRE	This command was integrated into Cisco IOS 12.2(33)SRE.

Examples

The following example shows output from a successful request when using the **debug packet-capture** command:

```
Router# debug packet-capture
Buffer Capture Infrastructure debugging is on
```

Related Commands

Command	Description
show monitor capture	Displays the contents of a capture buffer or a capture point.

debug pad

To display debugging messages for all packet assembler/disassembler (PAD) connections, use the **debug pad** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug pad

no debug pad

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0	This command was introduced in a release prior to Cisco IOS Release 12.0.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

Use the **debug pad** command to gather information to forward to the Cisco Technical Assistance Center (TAC) to assist in troubleshooting a problem that involves PAD connections.

The following example shows output of the **debug pad** and **debug x25 event** commands for an incoming PAD call destined for a terminal line. The incoming PAD call is rejected by the terminal line because the selected network closed user group (CUG) has not been subscribed to by the caller:

```
Router# debug pad
Router# debug x25 event
Serial1/1:X.25 I R1 Call (16) 8 lci 8
  From (7):2001534 To (9):200261150
  Facilities:(2)
    Closed User Group (basic):99
  Call User Data (4):0x01000000 (pad)
pad_svc_announce:destination matched 1
PAD:incoming call to 200261150 on line 130 CUD length 4
!PAD130:Incoming Call packet, Closed User Group (CUG) service protection, selected network
 CUG not subscribed
PAD:CUG service protection Cause:11 Diag:65
Serial1/1:X.25 O R1 Clear (5) 8 lci 8
  Cause 0, Diag 65 (DTE originated/Facility code not allowed)
Serial1/1:X.25 I R1 Clear Confirm (3) 8 lci 8
```

The following example shows the output of the **debug pad** command for an outgoing PAD call initiated from a terminal line with a subscribed CUG that bars outgoing access:

```
!PAD130:Outgoing Call packet, Closed User Group - CUG service validation, selected CUG !bars
 outgoing access
PAD130:Closing connection to .  In 0/0, out 0/0
```

debug piafs events

To check the debugging messages for Personal Handyphone Internet Access Forum Standard (PIAFS) calls, use the **debugpiafsevents** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug piafs events

no debug piafs events

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(8)T	This command was introduced on Cisco 803, Cisco 804, and Cisco 813 routers.

Usage Guidelines

The **debugpiafsevents**command provides debugging information for the PIAFS calls on the router, including the inband negotiation process.

Examples

The **debugpiafsevents** command was configured to provide the following information for PIAFS calls:

```
Router# debug piafs events
02:16:39:PIAFS events debugging is on
02:16:167516180371:PIAFS: RX <- CDAPI :cdapi_route_call Request
02:16:167517398148:PIAFS: RX <- CDAPI :CDAPI_MSG_CONNECT_IND
02:16:171798691839:PIAFS: TX -> CDAPI :CDAPI_MSG_SUBTYPE_ALERT_REQ
02:16:167503724545:PIAFS: TX -> CDAPI :CDAPI_MSG_CONNECT_RESP
02:16:167503765504:PIAFS: TX -> CDAPI :CDAPI_MSG_CONN_ACTIVE_REQ
02:16:167503724544:PIAFS: RX <- CDAPI :CDAPI_MSG_CONN_ACTIVE_IND
02:16:171798691839:PIAFS:Network allotted Channel :B1
02:16:167503765504:PIAFS:Enabling QMC in PIAFS mode for B1
02:16:171798691839:PIAFS:piafs_driver_enable_settings()
02:16:167503765504:PIAFS:The speed is :64
02:16:167503724544:PIAFS:Starting 64 kbps PIAFS Incoming
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:13 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:Updating conf resp num
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:1 RSN:1 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:14 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:2 RSN:2 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:15 RSN:1 CRSN:1 SISN:
255]
```

```
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:3 RSN:3 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:16 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:4 RSN:4 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:17 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:5 RSN:5 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:18 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:6 RSN:6 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:19 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:7 RSN:7 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:Piafs layer up & Main FSM set to DATA
02:16:39:PIAFS:Compression v42bis enabled
02:16:39:PIAFS:V42BIS:v42bis_init()
02:16:39:PIAFS:V42BIS:v42bis_init()
02:16:39:PIAFS:V42BIS:Negotiated Values for P1, P2 are - 4096 , 250
02:16:39:PIAFS:Incoming call invoking ISDN_CALL_CONNECT
02:16:39:%LINK-3-UPDOWN:Interface BRI0:1, changed state to up
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
```

```
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS:  Data Protocol:Version 1
02:16:39:PIAFS:  Control Protocol:Version 1
02:16:39:PIAFS:  RTF value:9
02:16:39:PIAFS:  Compression:V.42bis
02:16:39:PIAFS:  Frame Length:80
02:16:39:PIAFS:  Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:piafs_setmap() tx_map FFFFFFFF
02:16:39:PIAFS:piafs_setmap() rx_map 0
02:16:41:PIAFS:PPP:Autoselect sample 7E
02:16:41:PIAFS:PPP:Autoselect sample 7EFF
02:16:41:PIAFS:PPP:Autoselect sample 7EFF7D
02:16:41:PIAFS:PPP:Autoselect sample 7EFF7D23
02:16:41:PIAFS:piafs_setmap() tx_map FFFFFFFF
02:16:41:PIAFS:piafs_setmap() rx_map 0
02:16:42:PIAFS:piafs_setmap() tx_map A0000
02:16:42:PIAFS:piafs_setmap() rx_map 0
```

The following table describes the significant fields shown in the display.

Table 31: debug piafs events Field Descriptions

Field	Description
RX <- CDAPI :cdapi_route_call Request	The call distributor application programming interface (CDAPI) in the router receives an ISDN call request from the switch.
RX <- CDAPI :CDAPI_MSG_CONNECT_IND	The CDAPI in the router receives a connection indicator message from the switch.
TX -> CDAPI :CDAPI_MSG_SUBTYPE_ALERT_REQ	The CDAPI in the router transmits an alert request to the switch.
TX -> CDAPI :CDAPI_MSG_CONNECT_RESP	The CDAPI in the router transmits a connect response message to the switch.
TX -> CDAPI :CDAPI_MSG_CONN_ACTIVE_REQ	The CDAPI in the router transmits a connection active request to the switch.

Field	Description
RX <-CDAPI:CDAPI_MSG_CONN_ACTIVE_IND	The CDAPI in the router receives a connection active indicator from the switch.
Enabling QMC in PIAFS mode for B1	QMC (global multichannel parameters) are being enabled in PIAFS mode for the B1 channel.
piafs_driver_enable_settings()	The PIAFS driver is enabling the settings.
Starting 64 kbps PIAFS Incoming	The speed of the transmission in kbps. In this case, the speed is 64 kbps.
RX <- NEGO_SYNC_REQUEST[GSN: RSN: CRSN: SISN:]	The router receives a PIAFS negotiation synchronization request frame from the peer PIAFS device. The frame contains the following: general sequence number (GSN), reception sequence number (RSN), confirmation response sequence number (CRSN), and synchronization initiation sequence number (SISN).
Updating conf resp num	The confirmation response number is being updated.
TX -> NEGO_SYNC_RECEPTION[GSN: RSN: CRSN: SISN:]	The router transmits a PIAFS negotiation synchronization reception message to the peer PIAFS device. The message includes the GSN, RSN, CRSN, and SISN.
RX <- CONTROL_REQUEST	The router receives a PIAFS control request frame that includes communication parameters.
Rx Parameters	The communication parameters are as follows.
Data Protocol	The version of the data protocol.
Control Protocol	The version of the control protocol.
RTF value	Round-trip frame value.
Compression	The compression standard.
Frame Length	The length of the frame, in bytes.
Frame Number	The number of packets per frame.
TX -> CONTROL_RECEPTION	The router transmits a PIAFS control reception frame.
ACKed all the Rx control parameters	The control reception frame acknowledges all the communication parameters that were received from the peer.

Field	Description
Piafs layer up & Main FSM set to DATA	The PIAFS protocol is active on the router. The router is ready to receive data from the peer device.
Compression v42bis enabled	The compression protocol v42bis is enabled.
V42BIS:v42bis_init()	The v42bis compression protocol has been initiated.
V42BIS:Negotiated Values for P1, P2 are - 4096 , 250	In this example, P1 is the total count of encoded words when v42bis compression is enabled. P2 is the maximum letter line length for the V42bis compression.
Incoming call invoking ISDN_CALL_CONNECT	An incoming ISDN call connection message is received.
PPP	The PPP layer on the router becomes active and starts to process the PPP frame from the peer PIAFS device.

debug platform 6rd

To enable debugging for all IPv6 rapid deployment related occurrences on the Cisco 7600 router, and report on errors that occur for IPv6 rapid deployment, use the **debug platform 6rd** command in the privileged EXEC configuration mode. To disable the debugging, use the **no** form of the command .

debug platform 6rd {**events**| **errors**}

no debug platform 6rd {**events**| **errors**}

Syntax Description

events	Displays the debugging output for all IPv6 rapid deployment related occurrences on the router such as the creation of adjacencies, or the setting of the tunnel end-point.
errors	Displays the debugging output for problems related to IPv6 rapid deployment tunnel of RP.

Command Default

None

Command Modes

Privileged EXEC

Command History

Release	Modification
15.3.(2)S	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Use the debug command only to troubleshoot specific problems, or during troubleshooting sessions with Cisco technical support staff.

Examples

The following shows sample output for events debugging:

```
CE1#debug platform 6rd events
6rd Events debugging is on
CE1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
CE1(config)#int tunn56
CE1(config-if)#sh
CE1(config-if)#^Z
CE1#
*Mar  1 00:14:39.825 IST: cwan_release_6rd_tunnel_endpt: Released tunnel endpt 55
*Mar  1 00:14:41.041 IST: %SYS-5-CONFIG_I: Configured from console by console
CE1#
CE1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
CE1(config)#int tunn56
CE1(config-if)#no sh
CE1(config-if)#^Z
CE1#
```

```
*Mar  1 00:14:59.013 IST: %SYS-5-CONFIG_I: Configured from console by console
*Mar  1 00:14:59.645 IST: cwan_6rd_tun_adj_attach_info: event for prefix [::]
*Mar  1 00:14:59.645 IST: cwan_adjacency_set_6rd_tunnel_endpoint: dest ip is
[0.0.0.0]
*Mar  1 00:14:59.645 IST: cwan_6rd_tun_adj_attach_info: final results:
Tunnel56 tunnel adj update, , ltl = 0x83  set, adj_handle [0x50201CD0]
*Mar  1 00:14:59.645 IST:  Tunnel end point ID[0] Active flag[1]
Source IP[100.0.56.1] Destination IP[0.0.0.0] Tunnel Vlan[1069]
Tunnel I/f no[102] Physical Vlan[1192]
Source MAC[0000.0000.0000] Dest MAC[0013.80b4.1c40]
*Mar  1 00:14:59.665 IST: cwan_6rd_tun_adj_attach_info: event for prefix
[2001:B000:6438::1]
*Mar  1 00:14:59.665 IST: cwan_adjacency_set_6rd_tunnel_endpoint: dest ip is
[100.100.56.1]
*Mar  1 00:14:59.665 IST: cwan_get_6rd_tunnel_endpt: Allocated tunne endpt 111
*Mar  1 00:14:59.665 IST: cwan_6rd_tun_adj_attach_info Cleared pending flag
                               tun_endpt->tunnel endpt 111
*Mar  1 00:14:59.665 IST: cwan_6rd_tun_adj_attach_info: final results: Tunnel56
tunnel adj update, GigabitEthernet3/4, ltl = 0x83  set, adj_handle [0x50201B10]
*Mar  1 00:14:59.665 IST:  Tunnel end point ID[111] Active flag[1]
Source IP[100.0.56.1] Destination IP[100.100.56.1] Tunnel Vlan[1069]
Tunnel I/f no[102] Physical Vlan[1310]
Source MAC[0013.80b4.1c40] Dest MAC[001c.b0ca.2240]
```

Examples

The following shows sample output for errors debugging:

```
CE1#debug platform 6rd errors
6rd Errors debugging is on
CE1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
CE1(config)#int tunn56
CE1(config-if)#sh
CE1(config-if)#^Z
CE1#
*Mar  1 09:49:17.963 IST: cwan_release_6rd_tunnel_endpt: tunnel endpt 0 out of
range(1,8000)
*Mar  1 09:49:18.707 IST: %SYS-5-CONFIG_I: Configured from console by console
CE1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
CE1(config)#int tunn56
CE1(config-if)#no sh
CE1(config-if)#^Z
CE1#
*Mar  1 09:49:45.603 IST: %SYS-5-CONFIG_I: Configured from console by console
```

debug platform hardware qfp active feature evtmon

To debug the event monitoring features in the Cisco QuantumFlow Processor (QFP), use the **debug platform hardware qfp feature evtmon** command in Privileged EXEC mode. To disable this form of debugging, use the **no** form of this command.

debug platform hardware qfp {**active**| **standby**} **feature evtmon** {**client debug-level**| **datapath protocol**}

no debug platform hardware qfp {**active**| **standby**} **feature evtmon** {**client debug-level**| **datapath protocol**}

Syntax Description

active	**Enables debug logging for the active processor.**
standby	**Enables debug logging for the standby processor.**
evtmon	Displays the event monitoring information pertaining to the processor.
client	Specifies the event monitoring QFP client information for one of the following debug-level options: • all • error • info • trace • warning
datapath	*Specifies the event monitoring datapath for one of the following protocols:* • *ip* --ipv4 protocol • *ipv6* --ipv6 protocol

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.2S	This command was introduced on the Cisco ASR 1000 Series Routers.

Examples

The following example shows how to debug the event monitoring datapath for an IPv4 protocol: :

```
Router# debug platform hardware qfp active feature evtmon datapath ip
The selected EVTMON Datapath debugging is on
```

debug platform hardware qfp active feature ipsec

To display debugging information for IPsec events and counters in the Cisco Quantum Flow Processor (QFP) client, use the **debug platform hardware qfp active feature ipsec** command in privileged EXEC mode. To disable the display of this debugging information, use the **no** form of this command.

debug platform hardware qfp active feature ipsecdebug platform hardware qfp active feature ipsec {**client** {**error**| **info**| **trace**| **warning**}| **counter read-only**| **datapath** {**cce**| **droptype** *drop-type-number*| **error**| **info**| **pktcorrupt** *maximum-number*| **trace**| **warning**}}

no debug platform hardware qfp active feature ipsec {**client** {**error**| **info**| **trace**| **warning**}| **counter read-only**| **datapath** {**cce**| **droptype** *drop-type-number*| **error**| **info**| **pktcorrupt** *maximum-number*| **trace**| **warning**}}

Syntax Description

client	Enables debugging of IPsec events in the QFP client.
error	Enables debugging of errors.
info	Enables debugging of information.
trace	Enables debugging of packet tracing.
warning	Enables debugging of warnings.
counter	Enables debugging of IPsec counter settings in the QFP client.
read-only	Sets the debugging level of IPsec counter settings to read-only.
datapath	Enables debugging of IPsec events in the QFP datapath.
cce	Enables debugging of the IPsec common classification engine (CCE) in IPsec events.
droptype *drop-type-number*	Enables debugging of packet drop types in IPsec events. The range is from 1 to 69.
pktcorrupt *maximum-number*	Enables debugging of corrupt packets in QFP datapath IPsec events. The range for the maximum number of corrupt packets that are dumped is from 1 to 255.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.7S	This command was introduced.

Usage Guidelines

If you enter the **no debug all** command, debugging of the IPsec platforms is disabled.

Examples

The following example shows how to enable debugging for the IPsec datapath in QFP:

```
Device# debug platform hardware qfp active feature ipsec datapath cce

CPP IPSEC DATAPATH debugging is on
```

debug platform hardware qfp active feature wccp

To enable debug logging for the Web Cache Communication Protocol (WCCP) client in the Cisco Quantum Flow Processor (QFP), use the **debug platform hardware qfp active feature wccp** command in privileged EXEC mode. To disable WCCP QFP debug logging, use the **no** form of this command.

debug platform hardware qfp active feature wccp {{**client**| **lib-client** {**all**| **error**| **info**| **trace**| **warning**}}| **datapath all**}

no debug platform hardware qfp active feature wccp {{**client**| **lib-client** {**all**| **error**| **info**| **trace**| **warning**}}| **datapath all**}

Syntax Description

client	Enables WCCP QFP client debug logging.
lib-client	Enables WCCP QFP client-library debug logging.
all	Enables all logs.
error	Enables error logs.
info	Enables info logs.
trace	Enables trace logs.
warning	Enables warning logs.
datapath all	Enables all WCCP QFP datapath debug logging.

Command Default

WCCP QFP debug logging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.1S	This command was introduced.

Usage Guidelines

When the **debug platform hardware qfp active feature wccp** command is configured, QFP client debugs are enabled and can be collected from the forwarding processor (FP) from the file cpp_cp_F0-0.log.

When the **debug platform hardware qfp active feature wccp lib-client all** command is configured,QFP lib-client debugs are enabled and can be collected from the FP from the file fman-fp_F0-0.log.

When the **debug platform hardware qfp active feature wccp datapath all**command is configured, QFP datapath debugs are enabled and can be collected from the FP from the file cpp_cp-F0-0.log.

Examples

The following is sample output from the **debug platform hardware qfp active feature wccp** command:

```
Router# debug platform hardware qfp active feature wccp
```

A WCCP service is configured:

```
06/17 10:48:15.980 [(null)]: (debug): cpp_wccp_service_add_handler: service_params::: type
 =0 id = 0priority = 240  is_closed = 0 assign = 0
06/17 10:48:15.980 [(null)]: (debug): cpp_wccp_dplane_init dplane cpp-init for all cpps
06/17 10:48:15.980 [(null)]: (debug): cpp_wccp_dplane_init_cpp Enter: cpp_info = 0x1027b970:
.
.
.
```

The sequence of messages repeats for each access control entry (ACE) of a merged access control list (ACL):

```
06/17 10:53:38.792 [(null)]: (debug): cpp_wccp_update_bind_obj_list:idx = 63 bind-info:no.lvl
 = 1 fobj = 80024000 bind-id = 0
06/17 10:53:38.792 [(null)]: (debug): cpp_wccp_update_bind_obj_list fobj:service-id = 0
type = 0 cache-id = 9action = 2 acl-log = 0
06/17 10:53:38.792 [(null)]: (debug): cpp_wccp_add_dplane_cache_desc service-index = 0,
cache_id = 9
06/17 10:53:38.792 [(null)]: (debug): cpp_wccp_get_dplane_cache_index service-index = 0,
cache_id = 9
06/17 10:53:38.792 [(null)]: (debug): cpp_wccp_create_dplane_cache_index Cache index = 0
exists for cache-id = 9,service-index  = 0
.
.
.
```

WCCP redirection is configured on an interface:

```
06/17 13:15:44.655 [(null)]: (debug): cpp_wccp_intf_attach_msg req = 0x13116848, msg-len =
 36
06/17 13:15:44.655 [(null)]: (debug): cpp_wccp_intf_attach_handler: type = 0 id = 0 ifh =
17dir = 0 vrfid = 0
06/17 13:15:44.655 [(null)]: (debug): cpp_wccp_get_service_index WCCP: service_id 0 vrfid
0service_desc_index 0
06/17 13:15:44.655 [(null)]: (debug): cpp_wccp_get_service_desc: service-id: 0  type = 0
index = 0
.
.
.
```

Debug messages appear for each ACE of the merged ACL for a service group:

```
06/17 13:15:44.670 [(null)]: (debug): cpp_wccp_translate_fobj_to_cce_result Entry
06/17 13:15:44.670 [(null)]: (debug): cpp_wccp_get_service_index WCCP: service_id 0 vrfid
0service_desc_index 0
06/17 13:15:44.670 [(null)]: (debug): cpp_wccp_get_service_desc: service-id: 0  type = 0
index = 0
06/17 13:15:44.670 [(null)]: (debug): cpp_wccp_get_dplane_cache_index service-index = 0,
cache_id = 9
.
.
.
```

Redirection is removed from an interface:

```
06/17 13:24:54.617 [(null)]: (debug): cpp_wccp_intf_detach_handler: type = 0 id = 0 ifh =
17dir = 0 vrfid = 0
06/17 13:24:54.617 [(null)]: (debug): cpp_wccp_get_service_index WCCP: service_id 0 vrfid
0service_desc_index 0
06/17 13:24:54.617 [(null)]: (debug): cpp_wccp_get_service_desc: service-id: 0  type = 0
index = 0
06/17 13:24:54.617 [(null)]: (debug): cpp_wccp_intf_detach_handler:hw_cg_node, ifh = 17 dir
```

```
 = 0vrfid = 0 service-index = 0 exists
.
.
```

A service group is unconfigured:

```
06/17 13:29:41.828 [(null)]: (debug): cpp_wccp_cache_delete_handler: cache-desc ip-addr =
5a140102 id-addr = 0cache-id = 9  cef_handle = 0x112d3b68 cef-obj-type = 10router-id =
42424242 ce_mac_addr  fwd-method = 0 hw-addr = 0x11188f78
06/17 13:29:41.828 [(null)]: (debug): cpp_wccp_remove_dplane_ip_hash_entry cache_id= 9:
06/17 13:29:41.828 [(null)]: (debug): cpp_wccp_remove_dplane_ip_hash_entry ip-hash-index =
 6934:
.
.
.
```

The following is sample output from the **debug platform hardware qfp active feature wccp lib-client all**command:

```
Router#
debug platform hardware qfp active feature wccp lib-client all
```

A WCCP service group is configured:

```
06/17 13:47:00.158 [buginf]: (debug): cpp_wccp_service_group_add_a: API call from PAL
service-type = 0 id = 0vrfid = 0, priority = 240 is_closed = 0 has_ports = 1 assign-method
 = 0
06/17 13:47:00.158 [buginf]: (debug): cpp_wccp_api_async_msg_send: data size = 28 for this
 3message
06/17 13:47:00.158 [buginf]: (debug): cpp_wccp_api_async_send_cb: SMC async send call-back
.
.
.
```

The set of debug messages repeats for each ACE of the merged ACL of the WCCP service group:

```
06/17 13:47:29.474 [buginf]: (debug):   Notification from CGM to WCCP, op:13, tid:0,async:
 0, ctx: (nil)
06/17 13:47:29.474 [buginf]: (debug): cpp_wccp_cgm_notif_handler:cgm BIND num_lvl = 1,
bind-id = 0 fobj = 80028000
06/17 13:47:29.474 [buginf]: (debug):   Notification from CGM to WCCP, op:2, tid:0,async:
1,ctx: 0x77
.
.
.
```

WCCP redirection is configured on an interface:

```
06/17 13:52:05.841 [buginf]: (debug):   Notification from CGM to WCCP, op:1, tid:0,async:
0,ctx: (nil)
06/17 13:52:05.841 [buginf]: (debug): cpp_wccp_attach_service_to_intf_a: API call from PAL
 service-type = 0 id = 0 vrfid = 0 if_h = 11 dir = 0
06/17 13:52:05.841 [buginf]: (debug): cpp_wccp_attach_service_to_intf_a:tid el= 0x11347470
 ifh = 17, dir = 0 id = 0 type = 0 vrfid = 0
.
.
.
```

WCCP is unconfigured on an interface:

```
06/17 13:54:30.544 [buginf]: (debug):   Notification from CGM to WCCP, op:1, tid:0,async:
0,ctx: (nil)
06/17 13:54:30.544 [buginf]: (debug): cpp_wccp_detach_service_from_intf_a: API call from
PALservice-type = 0 id = 0 vrfid = 0 if_h = 11 dir = 0
06/17 13:54:30.544 [buginf]: (debug): cpp_wccp_detach_service_from_intf_a:tid el=
0x11338890ifh = 17, dir = 0 id = 0 type = 0
06/17 13:54:30.544 [buginf]: (debug):   Notification from CGM to WCCP, op:2, tid:0,async:
1,ctx: 0x79
.
.
.
```

A WCCP service group is unconfigured:

```
06/17 13:56:14.492 [buginf]: (debug): cpp_wccp_cache_delete_a: API call from PAL cache-id=
 10
06/17 13:56:14.492 [buginf]: (debug): cpp_wccp_api_async_msg_send: data size = 2 for this
6 message
06/17 13:56:14.492 [buginf]: (debug): cpp_wccp_api_async_send_cb: SMC async send call-back
06/17 13:56:14.492 [buginf]: (debug):  cpp_wccp_api_async_msg_send successfully sent msg-type
 6 to server.
06/17 13:56:14.492 [buginf]: (debug):   Notification from CGM to WCCP, op:1, tid:0,async:
0,ctx: (nil)
06/17 13:56:14.492 [buginf]: (debug):   Notification from CGM to WCCP, op:14, tid:0,async:
 0, ctx: (nil)
06/17 13:56:14.493 [buginf]: (debug): cpp_wccp_cgm_notif_handler:cgm BIND num_lvl = 1,
bind-id = 0 fobj = 80028000
.
.
.
```

The debug messages repeat for each ACE of the merged ACL for the WCCP service group:

```
06/17 13:56:14.500 [buginf]: (debug):   Notification from CGM to WCCP, op:14, tid:0,async:
 0, ctx: (nil)
06/17 13:56:14.500 [buginf]: (debug): cpp_wccp_cgm_notif_handler:cgm BIND num_lvl = 1,
bind-id = 0 fobj = 80028000
06/17 13:56:14.501 [buginf]: (debug):   Notification from CGM to WCCP, op:2, tid:0,async:
1,ctx: 0x7a
.


.
.
```

The following is sample output from the **debug platform hardware qfp active feature wccp datapath all**command:

```
Router# debug platform hardware qfp active feature wccp datapath all
```

A packet is successfully redirected:

```
06/17 14:49:28.935 [(null)]: (debug):
QFP:00 Thread:090 TS:00003918904609765795
##########################################################################################
06/17 14:49:28.936 [(null)]: (debug):
QFP:00 Thread:090 TS:00003918904609777642 CCE IPV4 PKT
(src:3.3.3.2,dst:2.2.2.2,sprt:0000,dprt:0050,prot:06,tos:00,len:0014,ttl:3f) , intf:3f3
06/17 14:49:28.936 [(null)]: (debug):
QFP:00 Thread:090 TS:00003918904609814715
##########################################################################################
06/17 14:49:28.936 [(null)]: (debug):
QFP:00 Thread:090 TS:00003918904609825865 CCE IPV4 UIDB_INFO W0:00000004, W1:00084441,
tcam_region_index:0004, key_index:00, cmd:00084441
06/17 14:49:28.936 [(null)]: (debug):
.
.
.
```

Related Commands

Command	Description
clear ip wccp	Removes WCCP statistics (counts) maintained on the router for a particular service.
ip wccp	Enables support of the specified WCCP service for participation in a service group.

Command	Description
ip wccp check services all	Enables enable all WCCP services.
ip wccp outbound-acl-check	Enables execution of ACL applied on the actual outgoing interface of a packet before a decision is taken to redirect a packet.
ip wccp redirect	Enables packet redirection on an outbound or inbound interface using WCCP.

debug platform hardware qfp feature

To debug features in the Cisco QuantumFlow Processor (QFP), use the debug platform hardware qfp feature command in Privileged EXEC mode. To disable this form of debugging, use the **no** form of this command.

debug platform hardware qfp {**active**| **standby**} **feature alg** {**client debug-level**| **datapath protocol** **[detail]**}

no debug platform hardware qfp {**active**| **standby**} **feature alg** {**client debug-level**| **datapath netbios** **[detail]**}

Syntax Description

active	Enables debug logging for the active processor.
standby	Enables debug logging for the standby processor.
alg	Displays the Application Level Gateway (ALG) information of the processor.
client	Specifies the ALG QFP client information.
debug-level	One of the following debug level options: • **all** • **error** • **info** • **trace** • **warning** **Note** The debug level options are not supported in the following protocols: • dns • ftp • h323 • ldap • sip • skinny • rtsp • rcmd • tftp • netbios

datapath	Specifies the ALG datapath.
protocol	One of the following protocols: • **dns** • **ftp** • **h323** • **http** • **imap** • **ldap** • **netbios** • **pop3** • **rcmd** • **rtsp** • **sip** • **skinny** • **smtp** • **sunrpc** • **tftp**
detail	(Optional) Specifies the QFP datapath ALG in detail.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.2	This command was introduced.
Cisco IOS XE Release 3.1S	This command was modified. Support for the Network Basic Input Output System (NetBIOS) protocol. The following keywords were added: **netbios-dgm**,**netbios-ns**,**netbios-ssn**.
15.1(1)S	This command was integrated into Cisco IOS XE Release 15.1(1)S

Examples

The following example shows how to debug the ALG datapath for a dns protocol:

```
Router# debug platform hardware qfp  active feature alg datapath dns
CPP ALG datapath event debugging is on
```

Related Commands

Command	Description
show platform hardware qfp feature	Displays feature specific information in QFP.

debug platform hardware qfp feature otv client

To enable Overlay Transport Virtualization (OTV) debugging on the Quantum Flow Processor (QFP) client, use the **debug platform hardware qfp feature otv client** command in privileged EXEC mode. To disable logging of the debug messages, use the **no** form of this command.

debug platform hardware qfp{**active**| **standby**}**feature otv client** {**all**| **error**| **info**| **trace**| **warning**}

no debug platform hardware qfp{**active**| **standby**}**feature otv client** {**all**| **error**| **info**| **trace**| **warning**}

Syntax Description

active	Enables debug of the active instance of the processor.
standby	Enables debug of the standby instance of the processor.
all	Enables all debugging.
error	Enables error debugging.
info	Enables info debugging.
trace	Enables trace debugging.
warning	Enables warning debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Examples

The following is sample output from the **debug platform hardware qfp feature otv client** command:

```
Router# debug platform hardware qfp feature otv client all

The output of the debug is saved on the tracelog file for cpp_cp_F0-0.log(or cpp_cp_F1-0.log):
[cpp_otv_ea_decap_unprovision:844] Entering
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_decap_unprovision:865] received decap unprovision message, is_async==1
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_decap_unprovision_cmn:434] cpp_ifhandle=741
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_decap_dp_unprovision:192] ifhandle=741 clear output subblock
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_decap_dp_unprovision:230] disable Overlay EFP feature cpp_ifhandle=7741
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_decap_unprovision_cmn:474] OTV decap chain unprovision success, cpp__ifhandle=741
```

```
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_msg_send_cb:47] Entering cpp_otv_ea_msg_send_cb
11/02 17:12:39.383 [(null)]: (debug):
[cpp_otv_ea_msg_send:104] send reply back to API LIB, async=1
11/02 17:12:39.384 [(null)]: (debug): m
[cpp_otv_ea_decap_unprovision:888] cpp_otv_ea_decap_unprovision retval=Success
```

Related Commands

Command	Description
show platform hardware qfp feature otv client interface	Displays OTV feature-specific information for the specified overlay interface

debug platform link-dc

To display debugging messages for the link daughter card, use the **debugplatformlink-dc**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform link-dc {**dwdm| interface| interrupt| netclk| serdes| transceiver| wanphy**}

no debug platform link-dc {**dwdm| interface| interrupt| netclk| serdes| transceiver| wanphy**}

Syntax Description

dwdm	OTN G.709/DWDM driver debug information.
interface	Interface driver debug information.
interrupt	Interrupt debug information.
netclk	Network clocking debug information.
serdes	Physical layer (PHY) and SerDes debug information.
transceiver	Pluggable optics module information.
wanphy	WAN PHY driver debug information.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRD	This command was introduced. **Note** This command applies only to the Cisco 7600 Series Ethernet Services Plus (ES+) line card on the Cisco 7600 series router.
12.2(33)SRD1	This command added the dwdm and wanphy keywords.

Usage Guidelines

Use this command with the remote command command or the attach command in privileged EXEC mode.

Examples

The following examples show the output for both the debug platform link-dc tranceiver command and the debug platform link-dc interrupt command. Notice that the show platform hardware transceiver command shows the status for the port.

```
Router# remote command module 1 debug platform link-dc tranceiver
Link-DC transceiver debugging is on
Router# remote command module 1 debug platform link-dc interrupt
Link-DC interrupt debugging is on
Router# remote command module 1 show debug
x40g subsystem:
  Link-DC transceiver debugging is on
  Link-DC interrupt debugging is on
Router# remote command module 1 show platform hardware transceiver status 1
Show status info for port 1:
TenGigabitEthernet1/1:
        State: Enabled
        Environmental Information - raw values
                Temperature: 7616
                Tx voltage: 0 in units of 100uVolt
                Tx bias: 28722 uA
                Tx power: -2 dBm (5441 in units of 0.1 uW)
                Rx power: 0 dBm (7712 in units of 0.1 uW)
                (AUX1) Laser Temperature: 8704
                (AUX2) +3.3V Supply Voltage: 32928
        XFP TX is enabled.
        XFP TX is soft enabled.
        XFP is ready.
        XFP is not power down.
        XFP is not soft power down.
        XFP doesn't have interrupt(s).
        XFP is not LOS.
        XFP data is ready.
        XFP TX path is ready.
        XFP TX laser is not in fault condition.
        XFP TX path CDR is locked.
        XFP RX path is ready.
        XFP RX path CDR is locked.
        No active alarms
        No active warning
Router-dfc1#
*Aug 15 11:20:26.436 PDT: DFC1: TenGigabitEthernet1/1 XFP: show status
*Aug 15 11:20:26.436 PDT: DFC1: TenGigabitEthernet1/1 XFP: show environmental monitoring
*Aug 15 11:20:26.436 PDT: DFC1: pluggable optics read - addr: 50, offset: 60, len: 14,
dataptr: 2377A668
*Aug 15 11:20:26.448 PDT: DFC1: pluggable optics read - addr: 50, offset: 6E, len: 2,
dataptr: 21AA028E
*Aug 15 11:20:26.452 PDT: DFC1: pluggable optics read - addr: 50, offset: 50, len: 2,
dataptr: 2377A6A0
*Aug 15 11:20:26.456 PDT: DFC1: pluggable optics read - addr: 50, offset: 52, len: 2,
dataptr: 2377A6A2
```

Note The following console log is seen when both the debug platform link-dc tranceiver command and the debug platform link-dc interrupt command are entered (as in the preceding example), and there is a transceiver Rx loss of signal (LOS) event.

```
Router-dfc1#
*Aug 15 11:23:52.127 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x8000
*Aug 15 11:23:52.127 PDT: DFC1: x40g_link_xphy_isr: xphy intr intr_st 0x80000
*Aug 15 11:23:52.127 PDT: DFC1: x40g_link_xphy_isr: xphy intr port 1
*Aug 15 11:23:52.127 PDT: DFC1: x40g_xphy_link_status_callout: port 1 link status 0
*Aug 15 11:23:52.131 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x8000
*Aug 15 11:23:52.131 PDT: DFC1: x40g_link_xphy_isr: xphy intr intr_st 0x80000
*Aug 15 11:23:52.131 PDT: DFC1: x40g_link_xphy_isr: xphy intr port 1
*Aug 15 11:23:52.131 PDT: DFC1: x40g_xphy_link_status_callout: port 1 link status 1
```

```
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_dc_process: interrupt msg_id 6, msg_num 1
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x8000
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_xphy_isr: xphy intr intr_st 0x80000
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_xphy_isr: xphy intr port 1
*Aug 15 11:23:52.135 PDT: DFC1: x40g_xphy_link_status_callout: port 1 link status 0
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x4000
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_xcvr_isr: intr_st 0x2, start 0, end 4, type
2,port_offset 0x0
*Aug 15 11:23:52.135 PDT: DFC1: Link xcvr port 1: Rx LOS interrupt
*Aug 15 11:23:52.135 PDT: DFC1: x40g_link_dc_process: interrupt msg_id 2, msg_num 1
*Aug 15 11:23:52.135 PDT: DFC1: Port 2: transceiver Rx LOS event
*Aug 15 11:23:52.147 PDT: DFC1: x40g_link_dc_process: xcvr oir timer timeout
00:12:37: %LINEPROTO-DFC1-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/2, changed
 state to down
*Aug 15 11:24:46.576 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x4000
*Aug 15 11:24:46.576 PDT: DFC1: x40g_link_xcvr_isr: intr_st 0x2, start 0, end 4, type
2,port_offset 0x0
*Aug 15 11:24:46.576 PDT: DFC1: Link xcvr port 1: Rx LOS interrupt
*Aug 15 11:24:46.576 PDT: DFC1: x40g_link_dc_process: interrupt msg_id 2, msg_num 1
*Aug 15 11:24:46.576 PDT: DFC1: Port 2: transceiver Rx LOS recovered
*Aug 15 11:24:46.580 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x8000
*Aug 15 11:24:46.580 PDT: DFC1: x40g_link_xphy_isr: xphy intr intr_st 0x80000
*Aug 15 11:24:46.580 PDT: DFC1: x40g_link_xphy_isr: xphy intr port 1
*Aug 15 11:24:46.580 PDT: DFC1: x40g_xphy_link_status_callout: port 1 link status 0
*Aug 15 11:24:46.584 PDT: DFC1: x40g_link_dc_interrupt_handler:  intr_status 0x8000
*Aug 15 11:24:46.584 PDT: DFC1: x40g_link_xphy_isr: xphy intr intr_st 0x80000
*Aug 15 11:24:46.584 PDT: DFC1: x40g_link_xphy_isr: xphy intr port 1
*Aug 15 11:24:46.584 PDT: DFC1: x40g_xphy_link_status_callout: port 1 link status 1
*Aug 15 11:24:46.584 PDT: DFC1: x40g_link_dc_process: interrupt msg_id 6, msg_num 1
*Aug 15 11:24:46.600 PDT: DFC1: x40g_link_dc_process: xcvr oir timer timeout
00:13:31: %LINEPROTO-DFC1-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/2, changed
 state to up
```

The following example shows the output for the debug platform link-dc dwdm command.

```
Router-dfc1# debug platform link-dc dwdm
Link-DC OTN G.709/DWDM debugging is on
*Jan 28 12:10:38.784 PDT: DFC1: Port 1: OTN Alarm Query, return ptr 228E877C
los 1, oof 0, lof 0, mfas 1, lom 0
otuAis 0, otuIae 0-0, otuBdi 0, otuTim 0
oduAis 0, oduBdi 0, oduLck 0, oduOci 0, oduPtim 0
*Jan 28 12:10:38.864 PDT: DFC1: x40g_link_pemaquid_pm_tick_timer_event(1): pm_tick timer
timeout
*Jan 28 12:10:39.364 PDT: DFC1: x40g_link_pemaquid_pm_tick_timer_event(1): pm_tick timer
timeout
*Jan 28 12:10:39.840 PDT: DFC1: Port 1: OTN Alarm Query, return ptr 228E877C
los 1, oof 0, lof 0, mfas 1, lom 0
otuAis 0, otuIae 0-0, otuBdi 0, otuTim 0
oduAis 0, oduBdi 0, oduLck 0, oduOci 0, oduPtim 0
```

The following example shows the output for the debug platform link-dc wanphy command.

```
Router-dfc1# debug platform link-dc wanphy
Link-DC WAN PHY debugging is on
*Jan 28 11:59:16.184 PDT: DFC1: Port 1 WIS alarms:
     ser 0, plm_p_far 0, ais_p_far 0, lof 0, los 0
     rdi 0, ais_l 0, lcd_p 0, plm_p 0, ais_p 0, lop 0

*Jan 28 11:59:17.184 PDT: DFC1: Port 1 WIS alarms:
     ser 0, plm_p_far 0, ais_p_far 0, lof 0, los 0
     rdi 0, ais_l 0, lcd_p 0, plm_p 0, ais_p 0, lop 0
*Jan 28 11:59:17.184 PDT: DFC1: Port 1 WIS counters: b1 0, b2 0, b3 0, fe_b2 0, fe_b3 0
*Jan 28 11:59:17.184 PDT: DFC1: Port 1 WIS J1RX: 0x0000000000000089.0x302E302E302E3000
...
*Jan 28 11:59:22.288 PDT: DFC1: Port 1 WIS alarms:
     ser 0, plm_p_far 0, ais_p_far 0, lof 0, los 0
     rdi 0, ais_l 0, lcd_p 0, plm_p 0, ais_p 0, lop 0
*Jan 28 11:59:22.288 PDT: DFC1: Port 1 WIS counters: b1 0, b2 0, b3 0, fe_b2 0, fe_b3 0
*Jan 28 11:59:22.288 PDT: DFC1: Port 1 WIS J1RX: 0x0000000000000089.0x302E302E302E3000
```

Related Commands

Command	Description
show platform hardware transceiver	Displays transceiver information on a port.

debug platform software evtmon

To debug the event monitoring features in the Cisco QuantumFlow Processor (QFP), use the **debug platform software evtmon** command in Privileged EXEC mode. To disable this form of debugging, use the **no** form of this command.

debug platform software evtmon configuration

no debug platform software evtmon configuration

Syntax Description

configuration	**Enables configuration-related debugs.**

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.2S	This command was introduced on the Cisco ASR 1000 Series Routers.

Examples

The following example shows how to debug the event monitoring configurations:

```
Router# debug platform software evtmon configuration
evtmon configuration messages debugging is on
```

debug platform software l2fib

To enable Overlay Transport Virtualization (OTV) debugging on the Cisco IOS daemon (IOSd) for the Layer 2 Forwarding Information Base (L2FIB) object, use the **debug platform software l2fib** command in privileged EXEC mode. To disable logging of the debug messages, use the **no** form of this command.

debug platform software l2fib {**error**| **events**| **verbose**}

no debug platform software l2fib {**error**| **events**| **verbose**}

Syntax Description

error	Enables error debugging.
events	Enables event debugging.
verbose	Enables verbose debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Examples

The following is sample output from the **debug platform software l2fib** command:

```
Router# debug platform software l2fib events

*Nov  2 16:41:37.593: FMANRP-L2fib: Print download message TDL:
message@l2fib_mlist_cfg: {
  l2fib_mlist@l2fib_mlist: {
    mlist_id@obj_id: {
      index@U32:4006
    }
    nlist@l2fib_nhop_list: {
      num_nhop@U32:1
      entry_cfg[0]@l2fib_nhop_update: {
        nhop@l2fib_nhop: {
          nhop_key@l2fib_nhop_key: {
            nhdl@U32:16033
          }
          nhop_type@l2fib_nhop_type:L2FIB_NHOP_TYPE_EFP
          nhop_type.efp@efp_idx: {
            bindidx@U32:0
          }
        }
        upd_type@l2fib_nhop_upd_type:L2FIB_NHOP_UPD_TYPE_DEL
      }
    }
  }
  cfg_action@cfg_action:MCP_CFG_ACTION_MODIFY
}
```

Related Commands

Command	Description
show platform software l2fib fp	Displays the global bridge domain table for MAC and Layer 2 multicast on the FMAN on the FP.
show platform software l2fib rp	Displays the global bridge domain table for MAC and multicast on the FMAN on the RP.

debug platform software multicast

To display information about log events, packet information, and assert events, use the **debug platform software multicast**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast {**all events**| **assert events**}

no debug platform software multicast {**all events**| **assert events**}

Syntax Description

all	Displays all multicast hardware switching debugging information, including errors, events, and packets for the specified group.
assert	Specifies the assert events.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from thedebug platform software multicastcommand using the all keyword:

```
PE-3-sp#debug platform software multicast all
Global enable but not the periodic debugging is on
PE-3-sp#
*Oct 30 09:17:26.150 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
*Oct 30 09:17:26.770 EDT: SP: hal_timer_event: NRPF-AG
*Oct 30 09:17:27.151 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
*Oct 30 09:17:28.151 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
*Oct 30 09:17:28.395 EDT: SP: hal_timer_event: NRPF-AG
*Oct 30 09:17:29.152 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
*Oct 30 09:17:30.152 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
*Oct 30 09:17:30.248 EDT: SP: hal_timer_event: NRPF-AGun al
*Oct 30 09:17:31.153 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
```

The following example shows output from the debug platform software multicastcommand using the assert keyword:

```
PE-3-sp#debug platform software multicast assert
Assertion for Layer 2 multicast debugging is on
PE-3-sp#
PE-3-sp#debug platform software multicast ha l2-sso all
Debug for mcast SSO all debugging is on
PE-3-sp#debug platform software multicast ha l2-sso err
PE-3-sp#debug platform software multicast ha l2-sso error
Debug for mcast SSO error debugging is on
PE-3-sp#debug platform software multicast ha l2-sso eve
PE-3-sp#debug platform software multicast ha l2-sso event
Debug for mcast SSO events debugging is on
PE-3-sp#debug platform software multicast ha l2-sso pak
PE-3-sp#debug platform software multicast ha l2-sso pak
Debug for mcast SSO packets debugging is on
PE-3-sp#
```

Related Commands

Command	Description
debug platform software multicast	Displays the multicast debugging information.

debug platform software multicast cgmp

To display information about cgmp debugging events and packet information use the **debug platform software multicast cgmp**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast cgmp {**event events**| **pak events**}

debug platform software multicast cgmp {**event events**| **pak events**}

Syntax Description

event	Specifies the events for the selected group.
pak	Specifies the packet information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast cgmp** command using the event keyword:

```
PE-3-sp#debug platform software multicast cgmp event
Router Discovery (CGMP Protocol) event log debugging is on
```

The following example shows output from the **debug platform software multicast cgmp** command using the pak keyword:

```
PE-3-sp#debug platform software multicast cgmp pak
```

Router Discovery (CGMP Protocol) packet log debugging is on

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast igmp

To display information about igmp debugging events and packet information use the **debug platform software multicast igmp**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast igmp {**event events**| **pak events**}

no debug platform software multicast igmp {**event events**| **pak events**}

Syntax Description

event	Specifies the igmp events for the selected group.
pak	Specifies the igmp packet information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast igmp** command using the event keyword:

```
Router# debug platform software multicast igmp event
PE-3-sp#debug platform software multicast igmp event
IGMP snooping event log debugging is on
...
```

The following example shows output from the **debug platform software multicast igmp** command using the pak keyword:

```
PE-3-sp#debug platform software multicast igmp pak
PE-3-sp#debug platform software multicast igmp pak
IGMP snooping packet log debugging is on
PE-3-sp#
*Oct 30 09:26:22.143 EDT: SP: RELAYED PAK to index 0x0008440B, vlan 1035
*Oct 30 09:26:22.143 EDT: SP: Packet dump:
18000070:              0100 5E000016 00000E00        ..^.......
18000080: 02000800 45000028 00000000 400254BC  ....E..(....@.T<
18000090: 46000002 E0000016 2200CBF6 00000001  F...`...".Kv....
180000A0: 01000001 E8000104 28000002 00010203  ....h...(.......
180000B0: 04058C
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast ip cmfib

To display information about multicast ip cmfib errors, shortcut events, and export the hardware statistics command, use the **debug platform software multicast ip cmfib**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ip cmfib {**error**| **events**| **stats**}

no debug platform software multicast ip cmfib {**error**| **events**| **stats**}

Syntax Description

error	Specifies the mfib IPV4 error information.
event	Specifies the IPv4 shortcut event information.
stats	Specifies the IPV4 hardware statistic information for export.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast ip cmfib** command using the error keyword:

```
PE-3-sp#debug platform software multicast ip cmfib cmfib error
CMFIB-LC IPv6 error debugging enabled
```

The following example shows output from the **debug platform software multicast ip cmfib** command using the event keyword:

```
PE-3-sp#debug platform software multicast ip cmfib cmfib eve
CMFIB-LC IPv6 event debugging enabled
```

The following example shows output from the **debug platform software multicast ip cmfib** command using the stats keyword:

```
PE-3-sp#debug platform software multicast ip cmfib cmfib stats
```

```
CMFIB-LC IPv6 stats debugging enabled
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast ip cmfib error

To display information about source or group IP address and the mfib IPv4 pending entry , use the **debug platform software multicast ip cmfib error**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ip cmfib error {**A.B.C.D**| **pending**}

no debug platform software multicast ip cmfib error {**A.B.C.D**| **pending**}

Syntax Description

A.B.C.D	Specifies the source or group IP address information.
pending	Specifies the mfib IPv4 pending entry error information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast ip cmfib error** command:

```
PE-3-sp#debug platform software multicast ip cmfib error 232.0.1.4 ver
PE-3-sp#debug platform software multicast ip cmfib error 232.0.1.4 verbose
CMFIB-LC IPv4 verbose error debugging enabled for group 232.0.1.4
PE-3-sp#debug platform software multicast ip cmfib error pending ?
<cr>
PE-3-sp#debug platform software multicast ip cmfib error pending
CMFIB-LC IPv4 error pending debugging enabled
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast ip cmfib event

To display information about source or group IP address, mfib IPv4 ctrl entries events, mfib hw-api events, mfib IPv4 table events, mfib IPv4 pending entry events, and mfib IPv4 table events, use the **debug platform software multicast ip cmfib event**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ip cmfib event {**A**.**B**.**C**.**D**| **ctrl**| **hwapi**| **mdt**| **pending**| **table**}

no debug platform software multicast ip cmfib event {**A**.**B**.**C**.**D**| **ctrl**| **hwapi**| **mdt**| **pending**| **table**}

Syntax Description

A.B.C.D	Specifies the source or group IP address information.
pending	Specifies the mfib IPv4 pending entry information.
ctrl	Specifies the mfib IPv4 ctrl entry events.
hwapi	Specifies the mfib hardware API events.
mdt	Specifies the mfib IPv4 table events.
table	Specifies the mfib IPv4 table events.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast ip cmfib event** command:

PE-3-sp#debug platform software multicast ip cmfib event ctrl

CMFIB-LC IPv4 event control debugging enabled

PE-3-sp#debug platform software multicast ip cmfib event hwapi

CMFIB-LC IPv4 event hwapi debugging enabled

```
PE-3-sp#debug platform software multicast ip cmfib event mdt
CMFIB-LC IPv4 event mdt debugging enabled
PE-3-sp#debug platform software multicast ip cmfib event pending
CMFIB-LC IPv4 event pending debugging enabled
PE-3-sp#debug platform software multicast ip cmfib event table
CMFIB-LC IPv4 event table debugging enabled
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast ip hal

To display information about the the multicast hal error, event, timer and packet information, use the **debug platform software multicast ip hal**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ip hal {**error events**| **event events**| **pak**| **timer**}

no debug platform software multicast hal {**error events**| **event events**| **pak**| **timer**}

Syntax Description

event	Specifies the events for the selected group.
error	Specifies the debugging errors.
pak	Specifies the packet information.
timer	Specifies the timer information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast ip hal**command using the event keyword:

```
PE-3-sp#debug platform software multicast ip  hal eve
PE-3-sp#debug platform software multicast ip  hal event
Multicast HAL event log debugging is on
PE-3-sp#
*Oct 30 09:24:48.078 EDT: SP: hal_timer_event: NRPF-AG
*Oct 30 09:24:48.790 EDT: SP: hal_timer_event: S-CHECK
*Oct 30 09:24:49.754 EDT: SP: hal_timer_event: NRPF-AG
*Oct 30 09:24:51.530 EDT: SP: hal_timer_event: NRPF-AG
*Oct 30 09:24:53.298 EDT: SP: hal_timer_event: NRPF-AG
*Oct 30 09:24:55.154 EDT: SP: hal_timer_event: NRPF-AG
```

The following example shows output from the **debug platform software multicast ip hal**command using the error keyword:

```
PE-3-sp#debug platform software multicast ip  hal error
Multicast HAL error log debugging is on
```

The following example shows output from the **debug platform software multicast ip hal**command using the pak keyword:

```
PE-3-sp#debug platform software multicast ip  hal pak
PE-3-sp#debug platform software multicast ip  hal pak
Multicast HAL packet log debugging is on
```

The following example shows output from the **debug platform software multicast ip hal**command using the timer keyword:

```
PE-3-sp#debug platform software multicast ip  hal tim
PE-3-sp#debug platform software multicast ip  hal timer
Multicast HAL timer log debugging is on
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast ipv6

To display information about multicast IPv6 hardware switching, use the **debug platform software multicast ipv6**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ipv6 {**control**| **error group-address**| **event group-address**}

no debug platform software multicast ipv6 {**control**| **error group-address**| **event group-address**}

Syntax Description

control	Displays all multicast hardware switching debugging information, including errors, events, and packets.
error group-address	Displays error messages related to multicast hardware switching for the specified group-address.
event group-address	Displays the run-time sequence of events for multicast hardware switching.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast ipv6**command using the **control** keyword:

```
Router# debug platform software multicast ipv6 control
```

The following example shows output from the **debug platform software multicast ipv6**command using the **error** keyword:

```
Router# debug mls rp ip multicast error
```

The following example shows output from the **debug platform software multicast ipv6**command using the **event** keyword:

```
Router# debug mls rp ip multicast event
```

Related Commands

Command	Description
ipv6 multicast hardware-switching connected	Downloads the interface and mask entry for IPv6 multicast packet.
ipv6 multicast hardware-switching replication-mode ingress	Configures the ingress hardware replication mode for IPv6 multicast packets.

debug platform software multicast ipv6 cmfib

To display information about multicast ipv6 mfib errors, shortcut events, and hardware statistics export information, use the **debug platform software multicast ipv6 cmfib**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ipv6 cmfib {**error**| **event**| **stats**}

no debug platform software multicast ipv6 cmfib {**error**| **event**| **stats**}

Syntax Description

error	Specifies the multicast ipv6 mfib errors.
event	Specifies the mfib IPv4 pending entry information.
stats	Specifies the hardware statistics export information.

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keyword is required.

Examples

The following example shows output from the **debug platform software multicast ipv6 cmfib** command:

```
PE-3-sp#debug platform software multicast ipv6 cmfib error
CMFIB-LC IPv6 error debugging enabled
PE-3-sp#debug platform software multicast ipv6 cmfib event
CMFIB-LC IPv6 event debugging enabled
PE-3-sp#debug platform software multicast ipv6 cmfib stats
CMFIB-LC IPv6 stats debugging enabled
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast ipv6

To display information about multicast IPv6 hardware switching, use the **debug platform software multicast ipv6**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ipv6 {**control**| **error group-address**| **event group-address**}

no debug platform software multicast ipv6 {**control**| **error group-address**| **event group-address**}

Syntax Description

control	Displays all multicast hardware switching debugging information, including errors, events, and packets.
error group-address	Displays error messages related to multicast hardware switching for the specified group-address.
event group-address	Displays the run-time sequence of events for multicast hardware switching.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast ipv6**command using the **control** keyword:

```
Router# debug platform software multicast ipv6 control
```

The following example shows output from the **debug platform software multicast ipv6**command using the **error** keyword:

```
Router# debug mls rp ip multicast error
```

The following example shows output from the **debug platform software multicast ipv6**command using the **event** keyword:

```
Router# debug mls rp ip multicast event
```

Related Commands

Command	Description
ipv6 multicast hardware-switching connected	Downloads the interface and mask entry for IPv6 multicast packet.
ipv6 multicast hardware-switching replication-mode ingress	Configures the ingress hardware replication mode for IPv6 multicast packets.

debug platform software multicast ipv6 hal

To display information about multicast ipv6 hal errors and event information, use the **debug platform software multicast ipv6 hal**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast ipv6 hal {**error**| **event**}

no debug platform software multicast ipv6 hal {**error**| **event**}

Syntax Description

error	Specifies the multicast ipv6 mfib errors.
event	Specifies the mfib IPv4 pending entry information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keyword is required.

Examples

The following example shows output from the **debug platform software multicast ipv6 hal** command:

```
PE-3-sp#debug platform software multicast ipv6 hal error
CMFIB-LC IPv6 debugging enabled
PE-3-sp#debug platform software multicast ipv6 hal event
CMFIB-LC IPv6 IPv6 HAL error debugging enabled
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast lc

To display the layer 2 line card multicast events, use the **debug platform software multicast lc**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast lc

no debug platform software multicast lc

Syntax Description

lc	Specifies the line card for which the multicast events are to be displayed.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast lc** command:

```
PE-3-sp#debug platform software multicast lc
Debug from mls_mcast_lc library debugging is on
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast mld

To display information about the events and packet information for mld debugging, use the **debug platform software multicast mld**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast mld {**event events**| **pak events**}

debug platform software multicast mld {**event events**| **pak events**}

Syntax Description

event	Specifies the mld events for the selected group.
pak	Specifies the mld packet information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast mld** command using the event keyword:

```
PE-3-sp#debug platform software multicast igmp event
multicast snooping event log debugging is on
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast mrouter

To display the multicast router events and packet information, use the **debug platform software multicast mrouter**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast mrouter {**event events**| **pak events**}

no debug platform software multicast mrouter {**event events**| **pak events**}

Syntax Description

event	Specifies the mld events for the selected group.
pak	Specifies the mld packet information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast mrouter** command using the event keyword:

```
PE-3-sp#debug platform software multicast mrouter event
Router Discovery (MLD MROUTER Protocol) event log debugging is on
```

The following example shows output from the **debug platform software multicast mrouter** command using the pak keyword:

```
PE-3-sp#debug platform software multicast mrouter pak
```

Router Discovery (MLD MROUTER Protocol) packet log debugging is on

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast msc

To display information about multicast shortcut debugging, use the **debug platform software multicast msc**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast msc {**error events**| **event events**| **pak events**}

no debug platform software multicast msc {**error events**| **event events**| **pak events**}

Syntax Description

events	Specifies the events for the selected group.
error	Specifies the debugging errors.
pak	Specifies the packet information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast msc** command using the error keyword:

```
PE-3-sp#debug platform software multicast msc error
Multicast Shortcuts error log debugging is on
```

The following example shows output from the **debug platform software multicast msc** command using the event keyword:

```
PE-3-sp#debug platform software multicast msc eve
Multicast Shortcuts event log debugging is on
```

The following example shows output from the **debug platform software multicast msc** command using the pak keyword:

```
PE-3-sp#debug platform software multicast msc pak
Multicast Shortcuts packet log debugging is on
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast rgmp

To display information about multicast shortcut debugging, use the **debug platform software multicast rgmp**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast rgmp {**event events**| **pak events**}

no debug platform software multicast rgmp {**event events**| **pak events**}

Syntax Description

events	Specifies the events for the selected group.
pak	Specifies the packet information.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast rgmp** command using the event keyword:

```
PE-3-sp#debug platform software multicast rgmp event
RGMP event log debugging is on
```

The following example shows output from the **debug platform software multicast rgmp** command using the pak keyword:

```
PE-3-sp#debug platform software multicast rgmp pak
RGMP packet log debugging is on
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast rpdf

To display information about multicast bidirectional df debugging, use the **debug platform software multicast rpdf**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast rpdf {**error events**| **event events**}

no debug platform software multicast rpdf {**error events**| **event events**}

Syntax Description

events	Specifies the events for the selected group.
error	Specifies the debugging errors.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast rpdf** command using the error keyword:

PE-3-sp#debug platform software multicast rpdf error

Multicast Shortcuts error log debugging is on

The following example shows output from the **debug platform software multicast rpdf** command using the event keyword:

PE-3-sp#debug platform software multicast rpdf eve

Multicast Shortcuts event log debugging is on

The following example shows output from the **debug platform software multicast rpdf** command using the pak keyword:

PE-3-sp#debug platform software multicast rpdf pak

Multicast Shortcuts packet log debugging is on

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software multicast titan

To display information about multicast titan debugging, use the **debug platform software multicast titan**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug platform software multicast titan {**error events**| **event events**}

no debug platform software multicast titan {**error events**| **event events**}

Syntax Description

events	Specifies the events for the selected group.
error	Specifies the debugging errors.

Command Default

Debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRE	This command was introduced on Cisco 7600 series routers.

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug platform software multicast titan** command using the error keyword:

```
PE-3-sp#debug platform software multicast rpdf error
Multicast Bidir RP/DF error log debugging is on
```

The following example shows output from the **debug platform software multicast titan** command using the event keyword:

```
PE-3-sp#debug platform software multicast rpdf eve
PE-3-sp#debug platform software multicast rpdf event
Multicast Bidir RP/DF event log debugging is on
```

Related Commands

Command	Description
debug platform software multicast ha	Displays the high availability multicast shortcuts debugging errors and events.

debug platform software otv

To enable Overlay Transport Virtualization (OTV) debugging on the Cisco IOS daemon (IOSd) Shim layer for OTV-specific forwarding object, use the **debug platform software otv** command in privileged EXEC mode. To disable logging of the debug messages, use the **no** form of this command.

debug platform software otv {**error**| **event**| **packet**}

no debug platform software otv {**error**| **event**| **packet**}

Syntax Description

error	Enables error debugging.
event	Enables event debugging.
packet	Enables packet debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Examples

The following is sample output from the **debug platform software otv** command:

```
Router# debug platform software otv event
```

```
*Nov  2 16:49:44.282: FMANRP-OTV: Create decap oce, obj_hdl 448FAD1C, obj_id 8
*Nov  2 16:49:44.283: FMANRP-OTV: efp_id 10 on if_num 26, dpidx 16916300
*Nov  2 16:49:44.283: OTV OCE Message sent for
*Nov  2 16:49:44.284: FMANRP-OTV: Modify decap oce, obj_hdl 448FAD1C, obj_id 8
*Nov  2 16:49:44.284: FMANRP-OTV: efp_id 10 on if_num 26, dpidx 16916300
*Nov  2 16:49:44.284: OTV OCE Message sent for
*Nov  2 16:49:44.284: FMANRP-OTV: Create encap oce, obj_hdl 4D14CE2C obj_id 9
*Nov  2 16:49:44.284: FMANRP-OTV: efp_id 10 on if_num 26, dpidx 16916300
*Nov  2 16:49:44.285: FMANRP-OTV: Next obj hdl 4CB67890 type 1D, obj_id 8E25, type 11
```

Related Commands

Command	Description
show platform software otv fp	Displays the overlay configuration on an OTV edge device on the FMAN on the FP.

debug platform software wccp

To enable Web Cache Control Protocol (WCCP) platform debug messages, use the **debug platform software wccp** command in privileged EXEC mode. To disable WCCP platform debug messages, use the **no** form of this command.

debug platform software wccp {**configuration**| **counters**| **detail**| **messages**}

no debug platform software wccp {**configuration**| **counters**| **detail**| **messages**}

Syntax Description

configuration	Enables configuration related debugs.
counters	Enables statistics collection related debugs.
detail	Enables detailed debugs for all WCCP related configurations.
messages	Enables debugs related to type definition language (TDL) messages being exchanged.

Command Default

Debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.2	This command was introduced.
Cisco IOS XE Release 3.1S	This command was modified. The **counters** keyword was added.

Examples

The following is sample output from the **debug platform software wccp configuration** command:

```
Router# debug platform software wccp configuration
```

A WCCP service is configured:

```
*Jun 17 15:41:04.816: FMANRP-WCCP: Config Service Group (0, 0, 0)
                        acl = , propagate_tos = TRUE, mode_is_closed = FALSE
                        definition_is_valid = TRUE, protocol = 6, priority = 240
                        ass_method = Unknown, fwd_method = Unknown, ret_method = Unknown
                        num_mv_sets = 0, redirection_is_active = FALSE, num_wcs = 0
                        use_source_port = FALSE, ports_defined = TRUE
                        ports[0] = 80
                        ports[1] = 0
                        ports[2] = 0
                        ports[3] = 0
                        ports[4] = 0
                        ports[5] = 0
                        ports[6] = 0
                        ports[7] = 0
*Jun 17 15:41:24.827: FMANRP-WCCP: create ce adjacency: CE = 90.20.1.2, fwd_method = GRE
oce= 0x30692230 adj = 0x306921C0 handle = 0x30692230 obj_id = 135
*Jun 17 15:41:24.827: FMANRP-WCCP: adjacency 90.20.1.2 (4500.0000.0000), router_id 66.66.66.66
 proto=47
*Jun 17 15:41:39.807: FMANRP-WCCP: update mask data, Service Group (0, 0, 0)
                        acl = , propagate_tos = TRUE, mode_is_closed = FALSE
                        definition_is_valid = TRUE, protocol = 6, priority = 240
                        ass_method = Mask, fwd_method = GRE, ret_method = L2
                        num_mv_sets = 1, redirection_is_active = TRUE, num_wcs = 1
                        use_source_port = FALSE, ports_defined = TRUE
                        wc[0] = 90.20.1.2
                        ports[0] = 80
                        ports[1] = 0
                        ports[2] = 0
                        ports[3] = 0
                        ports[4] = 0
                        ports[5] = 0
                        ports[6] = 0
                        ports[7] = 0
*Jun 17 15:41:39.808: FMANRP-WCCP: Service Group (0, 0, 0) generate merged acl from IOS
*Jun 17 15:41:39.808: FMANRP-WCCP: wccp merged_acl(acl=), p=64 t=64 MCP wccp merged_acl,
num_port=1 result_len=64
```

A WCCP service is configured on an interface:

```
*Jun 17 15:45:17.083: FMANRP-WCCP: Config Service Group (0, 0, 0) to interface
GigabitEthernet0/3/1, direction = IN
*Jun 17 15:45:17.084: FMANRP-WCCP: Attach GigabitEthernet0/3/1 interface info for Service
group (0, 0, 0) if_handle 20, direction Input(0x2)
```

A WCCP service is removed from an interface:

```
*Jun 17 15:46:29.815: FMANRP-WCCP: Unconfig Service Group (0, 0, 0) to interface
GigabitEthernet0/3/1, direction = IN
*Jun 17 15:46:29.815: FMANRP-WCCP: Detach GigabitEthernet0/3/1 interface info for Service
group (0, 0, 0) if_handle 20, direction Input(0x2)
```

A WCCP service group is unconfigured:

```
*Jun 17 15:48:17.224: FMANRP-WCCP: (0 0 0) Delete ce = 90.20.1.2
*Jun 17 15:48:17.225: Failed to retrieve service group params while removing ce
*Jun 17 15:48:17.241: FMANRP-WCCP: Unconfig Service Group (0, 0, 0)
```

The following is sample output from **debug platform software wccp messages**command:

```
Router# debug platform software wccp messages
```

A WCCP service is configured:

```
*Jun 17 15:50:57.796: FMANRP-WCCP: send out (0, 0, 0) wccp_svc_cfg (ADD) to fman-rp
                        pri=0, ce_num=0, ass=Unknown, fwd=Unknown, ret=Unknown
                        protocol=6 use_source_port=0 is_closed=0
                        ports[0] = 80
                        ports[1] = 0
                        ports[2] = 0
                        ports[3] = 0
                        ports[4] = 0
                        ports[5] = 0
                        ports[6] = 0
                        ports[7] = 0
*Jun 17 15:51:14.864: FMANRP-WCCP: send out (0, 0, 0) wccp_ce_cfg (ADD) to fman-rp,
ce=90.20.1.2 ce_id=0.0.0.0 rtr_id=66.66.66.66 fwd_method=GRE obj_id=141
*Jun 17 15:51:29.846: FMANRP-WCCP: send out (0, 0, 0) wccp_svc_cfg (MODIFY) to fman-rp
                        pri=0, ce_num=1, ass=Mask, fwd=GRE, ret=L2
                        protocol=6 use_source_port=0 is_closed=0
                        ports[0] = 80
                        ports[1] = 0
                        ports[2] = 0
                        ports[3] = 0
                        ports[4] = 0
                        ports[5] = 0
                        ports[6] = 0
                        ports[7] = 0
*Jun 17 15:51:29.847: FMANRP-WCCP: send out (0, 0, 0) wccp_acl_begin to fman-rp
*Jun 17 15:51:29.886: FMANRP-WCCP:   Service Group (0, 0, 0) send out ACL=WCCP_ACL_0x0, 64
 ACEs to fman-rp
*Jun 17 15:51:29.886: FMANRP-WCCP: send out (0, 0, 0) wccp_acl_end to fman-rp
```

A WCCP service is removed from an interface:

```
*Jun 17 15:53:40.710: FMANRP-WCCP: send out (0, 0, 0) wccp_if_svc_bind (ADD) to fman-rp
if_handle=20 dir=IN
```

A WCCP service is removed from an interface:

```
*Jun 17 15:54:36.924: FMANRP-WCCP: send out (0, 0, 0) wccp_if_svc_bind (DELETE) to fman-rp
  if_handle=20 dir=IN
```

A WCCP service group is unconfigured:

```
*Jun 17 15:55:13.117: FMANRP-WCCP: send out (0, 0, 0) wccp_ce_cfg (DELETE) to fman-rp,
ce=90.20.1.2 ce_id=0.0.0.0 rtr_id=0.0.0.0 fwd_method=Unknown obj_id=0
*Jun 17 15:55:13.128: FMANRP-WCCP: send out (0, 0, 0) wccp_svc_cfg (DELETE) to fman-rp
                        pri=0, ce_num=0, ass=Unknown, fwd=Unknown, ret=Unknown
                        protocol=0 use_source_port=0 is_closed=0
                        ports[0] = 0
                        ports[1] = 0
                        ports[2] = 0
                        ports[3] = 0
                        ports[4] = 0
                        ports[5] = 0
                        ports[6] = 0
                        ports[7] = 0
```

The following is sample output from the **debug platform software wccp detail** command:

```
Router# debug platform software wccp detail
```

WCCP service is configured:

```
*Jun 17 18:42:15.491: FMANRP-WCCP: create ce adjacency: CE = 90.20.1.2, fwd_method = GRE
oce= 0x30692230 adj = 0x306921C0 handle = 0x30692230 obj_id = 181
*Jun 17 18:42:30.472: FMANRP-WCCP: Converted adjacency (0x30692230), to ce_addr (90.20.1.2)
*Jun 17 18:42:30.473: FMANRP-WCCP:  Service Group (0, 0, 0) send out ACL=WCCP_ACL_0x0,
ACE=1, obj_id=181 PERMIT, srcopr 5, dstopr 3 to fman-rp
*Jun 17 18:42:30.473: FMANRP-WCCP: oce 0x30692230 adj 0x306921C0 handle 0x30692230
```

The debug messages appear for each access control entry (ACE) of the merged access control list (ACL) for the service group:

```
*Jun 17 18:42:30.487: FMANRP-WCCP: Converted adjacency (0x30692230), to ce_addr (90.20.1.2)
*Jun 17 18:42:30.487: FMANRP-WCCP:  Service Group (0, 0, 0) send out ACL=WCCP_ACL_0x0,
ACE=64, obj_id=181 PERMIT, srcopr 5, dstopr 3 to fman-rp
*Jun 17 18:42:30.487: FMANRP-WCCP: oce 0x30692230 adj 0x306921C0 handle 0x30692230
```

A WCCP service group is unconfigured:

```
*Jun 17 18:46:34.316: FMANRP-WCCP: (0 0 0) Delete ce = 90.20.1.2
*Jun 17 18:46:34.316: Failed to retrieve service group params while removing ce
```

The following is sample output from the **debug platform software wccp counters**command.

```
Router# debug platform software wccp counters
```

Statistics are collected for the first time on a WCCP-enabled interface:

```
*Jun 17 18:50:18.930: FMANRP-WCCP: Received wccp_if_stats intf 20, redirect(IN) 0 from
fman-fp
```

The following debug messages are displayed every 10 seconds:

```
*Jun 17 18:51:18.929: FMANRP-WCCP: Received (0, 0, 0) svc_grp_stats from fman-fp
  unassigned_count = 0, dropped_closed_count = 0
  bypass_count = 0, bypass_failed_count = 0
  denied_count = 0, redirect_count = 0
  num_entries = 0
*Jun 17 18:51:18.929: FMANRP-WCCP: Received wccp_if_stats intf 20, redirect(IN) 0 from
fman-fp
*Jun 17 18:51:28.929: FMANRP-WCCP: Received (0, 0, 0) svc_grp_stats from fman-fp
  unassigned_count = 0, dropped_closed_count = 0
  bypass_count = 0, bypass_failed_count = 0
  denied_count = 0, redirect_count = 0
  num_entries = 0
```

Related Commands

Command	Description
clear ip wccp	Removes WCCP statistics (counts) maintained on the router for a particular service.
ip wccp	Enables support of the specified WCCP service for participation in a service group.
ip wccp check services all	Enables all WCCP services.
ip wccp outbound-acl-check	Enables execution of ACL applied on the actual outgoing interface of a packet before a decision is taken to redirect a packet.
ip wccp redirect	Enables packet redirection on an outbound or inbound interface using WCCP.
show platform software wccp	Displays global statistics related to WCCP on Cisco ASR 1000 Series Routers.

debug policy-firewall

Note Effective with Cisco IOS Release 12.4(20)T, the **debug policy-firewall** command replaces the **debug ip inspect** command.

To display messages about Cisco software firewall events, including details about the packets of the protocol, use the **debug policy-firewall** command in priviliged EXEC mode. To disable the display of debugging output, use the **no** form of this command.

debug policy-firewall {**function-trace**| **object-creation**| **ha**| **object-deletion**| **list** {**access-list**| **extended-access-list**}| **events**| **timers**| **packet-path**| **protocol** *protocol-name*| **L2-transparent**| **control-plane**| **detailed**}

no debug policy-firewall {**function-trace**| **object-creation**| **object-deletion**| **list** {**access-list**| **extended-access-list**}| **events**| **timers**| **packet-path**| **protocol** *protocol-name*| **L2-transparent**| **control-plane**| **detailed**| **ha**}

Syntax Description

function-trace	Displays messages about software functions called by the firewall.
object-creation	Displays messages about software objects being created by the firewall. Object creation corresponds to the beginning of firewall-inspected sessions.
object-deletion	Displays messages about software objects being deleted by the firewall. Object deletion corresponds to the closing of firewall-inspected sessions.
list	Displays messages about policy firewall conditional debugging.
access-list	Filters the basic list of policy firewall conditional debugging messages. The valid range is from 1 to 199.
extended-access-list	Filters the extended range of policy firewall conditional debugging messages. The valid range is from 1300 to 2699.
events	Displays messages about firewall software events, including information about firewall packet processing or MIB special events.
timers	Displays messages about firewall timer events such as when the firewall idle timeout is reached.
packet-path	Displays messages about the packet-path functions.

protocol *protocol-name*	Displays firewall-inspected protocol events. Displays messages about firewall-inspected protocol events, including details about the packets of the protocol.
L2-transparent	Displays messages about Layer 2 transparent (firewall) bridge mode events.
control-plane	Displays messages about the control plane routines.
detailed	Detailed information is displayed for all the other enabled firewall debug commands. Use this form of the command in conjunction with the other firewall **debug** commands.
ha	Displays firewall high availability (HA) log messages.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(20)T	This command was introduced. This command replaces the **debug ip inspect** command.
15.0(1)M	This command was modified. The **list** and **packet-path** keywords were added.
15.2(3)T	This command was modified. The **ha** keyword was added.

Usage Guidelines

The **debug policy-firewall** command is used to troubleshoot firewall problems. You can use the output of this command to analyze the behavior of the firewall and to diagnose the root cause of the problem.

Examples

The following is sample output from the **debug policy-firewall function-trace** command:

```
Device# debug policy-firewall function-trace

Feb 13 08:13:43: FIREWALL: fw_dp_tcp_init_sis():
Feb 13 08:13:43: FIREWALL: fw_dp_insp_init_sis():
Feb 13 08:13:43: FIREWALL: fw_dp_tcp_inspect(): , i2r = 1
Feb 13 08:13:43: FIREWALL: fw_dp_insp_listen_state():
Feb 13 08:13:43: FIREWALL: fw_dp_insp_ensure_return_traffic():
Feb 13 08:13:43: FIREWALL: fw_dp_insp_process_syn_packet():
Feb 13 08:13:43: FIREWALL: fw_dp_insp_create_tcp_host_entry():
Feb 13 08:13:43: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 0
Feb 13 08:13:43: FIREWALL*: fw_dp_insp_synsent_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 1
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_synrcvd_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_remove_sis_from_host_entry():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_remove_host_entry():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_delete_host_entry():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_handle_icq_control_stream():
```

```
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 0
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_estab_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_handle_icq_control_stream():
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 1
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_estab_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_handle_icq_control_stream():
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 0
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_estab_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_handle_icq_control_stream():
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 1
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_estab_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_handle_icq_control_stream():
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 0
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_estab_state():
Feb 13 08:13:44: FIREWALL*: fw_dp_tcp_inspect(): , i2r = 1
Feb 13 08:13:44: FIREWALL*: fw_dp_insp_estab_state():
Feb 13 08:13:44: %APPFW-6-IM_ICQ_SESSION: im-icq text-chat service session initiator sends
 77 bytes  session 192.168.3.3:36091 192.168.103.3:5190 on zone-pair zp_test_in class test_im
 appl-class test_icq_1
```

The date in each line of the output is the time stamp. This output shows the functions called by the Cisco IOS firewall as a session is inspected. Entries with an asterisk (*) after the word “FIREWALL” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug policy-firewall object-creation**, **debug policy-firewall object-deletion**, **debug policy-firewall timers**, and **debug policy-firewall events** commands:

```
Log Buffer (600000 bytes):
Feb 13 08:16:17: FIREWALL: FW CCE got packet 0x66030694 in process path
Feb 13 08:16:17: FIREWALL: Router gen or router destined pak 0x66030694, let it pass
Feb 13 08:16:17: FIREWALL: FW CCE got packet 0x660311F8 in process path
Feb 13 08:16:17: FIREWALL: Router gen or router destined pak 0x660311F8, let it pass
Feb 13 08:16:17: FIREWALL: FW CCE got packet 0x66030A60 in process path
Feb 13 08:16:17: FIREWALL: Router gen or router destined pak 0x66030A60, let it pass
Feb 13 08:16:19: FIREWALL: FW CCE got packet 0x660328C0 in process path
Feb 13 08:16:19: FIREWALL: Router gen or router destined pak 0x660328C0, let it pass
Feb 13 08:16:21: FIREWALL: FW CCE got packet 0x66031D5C in process path
Feb 13 08:16:21: FIREWALL: Router gen or router destined pak 0x66031D5C, let it pass
Feb 13 08:16:22: FIREWALL: FW CCE got packet 0x66032128 in process path
Feb 13 08:16:22: FIREWALL: Router gen or router destined pak 0x66032128, let it pass
Feb 13 08:16:22: FIREWALL: FW CCE got packet 0x660324F4 in process path
Feb 13 08:16:22: FIREWALL: Router gen or router destined pak 0x660324F4, let it pass
Feb 13 08:16:24: FIREWALL: FW CCE got packet 0x66033424 in process path
Feb 13 08:16:24: FIREWALL: Router gen or router destined pak 0x66033424, let it pass
Feb 13 08:16:25: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:16:25: FIREWALL: fw_dp_insp_sample_session_rate
Feb 13 08:16:26: FIREWALL: FW CCE got packet 0x66032C8C in process path
Feb 13 08:16:26: FIREWALL: Router gen or router destined pak 0x66032C8C, let it pass
Feb 13 08:16:26: FIREWALL: FW CCE got packet 0x6602DCD0 in process path
Feb 13 08:16:26: FIREWALL: Router gen or router destined pak 0x6602DCD0, let it pass
Feb 13 08:16:26: FIREWALL: FW CCE got packet 0x5011DDB4 in process path
Feb 13 08:16:26: FIREWALL: Router gen or router destined pak 0x5011DDB4, let it pass
Feb 13 08:16:28: FIREWALL: FW CCE got packet 0x5011D9E8 in process path
Feb 13 08:16:28: FIREWALL: sis 20491840 : Timer Start: Timer: 20491964 Time: 30000 milisecs
Feb 13 08:16:28: FIREWALL: sis 20491840 : Timer Init Leaf
Feb 13 08:16:28: FIREWALL: sis 20491840 : Allocating L7 sis extensionL4 protocol = 1, L7
protocol = 62, granular = 5
Feb 13 08:16:28: FIREWALL: sis 20491840 : create host entry 669F3180 addr 192.168.103.3
bucket 12 (vrf 0:0) fwfo 0x507E39C0
Feb 13 08:16:29: FIREWALL*: sis 20491840 : Timer Start: Timer: 20491964 Time: 3600000
milisecs
Feb 13 08:16:29: %APPFW-6-IM_ICQ_SESSION: im-icq text-chat service session initiator sends
 77 bytes  session 192.168.3.3:36091 192.168.103.3:5190 on zone-pair zp_test_in class test_im
 appl-class test_icq_1
Feb 13 08:16:29: %APPFW-6-IM_ICQ_SESSION: im-icq text-chat service session initiator gets
198 bytes  session 192.168.103.3:5190 192.168.3.3:36091 on zone-pair zp_test_in class test_im
 appl-class test_icq_1
Feb 13 08:16:29: FIREWALL: FW CCE got packet 0x20159864 in process path
Feb 13 08:16:29: FIREWALL: Router gen or router destined pak 0x20159864, let it pass
Feb 13 08:16:29: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:16:29: FIREWALL: delete host entry 669F3180 addr 192.168.103.3
Feb 13 08:16:30: FIREWALL: FW CCE got packet 0x66033058 in process path
Feb 13 08:16:30: FIREWALL: Router gen or router destined pak 0x66033058, let it pass
Feb 13 08:16:31: FIREWALL: FW CCE got packet 0x660337F0 in process path
Feb 13 08:16:31: FIREWALL: Router gen or router destined pak 0x660337F0, let it pass
Feb 13 08:16:31: FIREWALL: FW CCE got packet 0x20159C30 in process path
Feb 13 08:16:31: FIREWALL: Router gen or router destined pak 0x20159C30, let it pass
Feb 13 08:16:34: FIREWALL: FW CCE got packet 0x20159FFC in process path
Feb 13 08:16:34: FIREWALL: Router gen or router destined pak 0x20159FFC, let it pass
Feb 13 08:16:35: FIREWALL: FW CCE got packet 0x5011E54C in process path
Feb 13 08:16:35: FIREWALL: Router gen or router destined pak 0x5011E54C, let it pass
Feb 13 08:16:36: FIREWALL: FW CCE got packet 0x665E6304 in process path
Feb 13 08:16:36: FIREWALL: Router gen or router destined pak 0x665E6304, let it pass
Feb 13 08:16:36: FIREWALL: FW CCE got packet 0x5011E180 in process path
Feb 13 08:16:36: FIREWALL: Router gen or router destined pak 0x5011E180, let it pass
Feb 13 08:16:38: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:16:38: FIREWALL: fw_dp_insp_sample_session_rate
Feb 13 08:16:38: FIREWALL: FW CCE got packet 0x2015A3C8 in process path
Feb 13 08:16:38: FIREWALL: Router gen or router destined pak 0x2015A3C8, let it pass
Feb 13 08:16:39: FIREWALL: FW CCE got packet 0x5011E918 in process path
Feb 13 08:16:39: FIREWALL: Router gen or router destined pak 0x5011E918, let it pass
Feb 13 08:16:40: FIREWALL: FW CCE got packet 0x665E6E68 in process path
Feb 13 08:16:40: FIREWALL: Router gen or router destined pak 0x665E6E68, let it pass
Feb 13 08:16:40: FIREWALL: FW CCE got packet 0x2015A794 in process path
Feb 13 08:16:40: FIREWALL: Router gen or router destined pak 0x2015A794, let it pass
Feb 13 08:16:43: FIREWALL: FW CCE got packet 0x665E7234 in process path
Feb 13 08:16:43: FIREWALL: Router gen or router destined pak 0x665E7234, let it pass
Feb 13 08:16:44: FIREWALL: FW CCE got packet 0x5011ECE4 in process path
Feb 13 08:16:44: FIREWALL: Router gen or router destined pak 0x5011ECE4, let it pass
Feb 13 08:16:44: FIREWALL: FW CCE got packet 0x2015AB60 in process path
Feb 13 08:16:44: FIREWALL: Router gen or router destined pak 0x2015AB60, let it pass
Feb 13 08:16:45: FIREWALL: FW CCE got packet 0x665E7600 in process path
Feb 13 08:16:45: FIREWALL: Router gen or router destined pak 0x665E7600, let it pass
Feb 13 08:16:48: FIREWALL: FW CCE got packet 0x665E79CC in process path
Feb 13 08:16:48: FIREWALL: Router gen or router destined pak 0x665E79CC, let it pass
Feb 13 08:16:48: FIREWALL: FW CCE got packet 0x5011F47C in process path
Feb 13 08:16:48: FIREWALL: Router gen or router destined pak 0x5011F47C, let it pass
Feb 13 08:16:49: FIREWALL: FW CCE got packet 0x6602E468 in process path
Feb 13 08:16:49: FIREWALL: Router gen or router destined pak 0x6602E468, let it pass
Feb 13 08:16:50: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:16:50: FIREWALL: fw_dp_insp_sample_session_rate
Feb 13 08:16:50: FIREWALL: FW CCE got packet 0x2015B2F8 in process path
Feb 13 08:16:50: FIREWALL: Router gen or router destined pak 0x2015B2F8, let it pass
Feb 13 08:16:52: FIREWALL: FW CCE got packet 0x6602E09C in process path
Feb 13 08:16:52: FIREWALL: Router gen or router destined pak 0x6602E09C, let it pass
Feb 13 08:16:53: FIREWALL: FW CCE got packet 0x6602EC00 in process path
Feb 13 08:16:53: FIREWALL: Router gen or router destined pak 0x6602EC00, let it pass
Feb 13 08:16:54: FIREWALL: FW CCE got packet 0x6602EFCC in process path
Feb 13 08:16:54: FIREWALL: Router gen or router destined pak 0x6602EFCC, let it pass
Feb 13 08:16:55: FIREWALL: FW CCE got packet 0x6602F764 in process path
Feb 13 08:16:55: FIREWALL: Router gen or router destined pak 0x6602F764, let it pass
Feb 13 08:16:57: FIREWALL: FW CCE got packet 0x6602F398 in process path
Feb 13 08:16:57: FIREWALL: Router gen or router destined pak 0x6602F398, let it pass
Feb 13 08:16:57: FIREWALL: FW CCE got packet 0x6602FB30 in process path
Feb 13 08:16:57: FIREWALL: Router gen or router destined pak 0x6602FB30, let it pass
Feb 13 08:16:59: FIREWALL: FW CCE got packet 0x66030E2C in process path
Feb 13 08:16:59: FIREWALL: Router gen or router destined pak 0x66030E2C, let it pass
Feb 13 08:16:59: FIREWALL: FW CCE got packet 0x66030694 in process path
Feb 13 08:16:59: FIREWALL: Router gen or router destined pak 0x66030694, let it pass
Feb 13 08:17:00: FIREWALL*: sis 20491840 : Timer Start: Timer: 20491964 Time: 5000 milisecs
Feb 13 08:17:00: FIREWALL*: sis 20491840 : Timer Start: Timer: 20491964 Time: 1000 milisecs
Feb 13 08:17:01: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:17:01: FIREWALL: sis 20491840 : Idle Timer Expires: Timer: 20491964
Feb 13 08:17:01: FIREWALL: sis 20491840 : Delete sis half_open 0
Feb 13 08:17:01: FIREWALL: sis 20491840 : Timer Stop: Timer: 20491964
Feb 13 08:17:01: FIREWALL: sis 20491840 : Delete sis
Feb 13 08:17:01: FIREWALL: sis 20491840 : session on temporary delete list
Feb 13 08:17:01: FIREWALL: sis 20491840 : Calling l4 cleanup
Feb 13 08:17:01: FIREWALL: FW CCE got packet 0x660311F8 in process path
Feb 13 08:17:01: FIREWALL: Router gen or router destined pak 0x660311F8, let it pass
Feb 13 08:17:02: FIREWALL: FW CCE got packet 0x66030A60 in process path
Feb 13 08:17:02: FIREWALL: Router gen or router destined pak 0x66030A60, let it pass
Feb 13 08:17:02: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:17:02: FIREWALL: fw_dp_insp_sample_session_rate
Feb 13 08:17:04: FIREWALL: FW CCE got packet 0x66031990 in process path
Feb 13 08:17:04: FIREWALL: Router gen or router destined pak 0x66031990, let it pass
Feb 13 08:17:04: FIREWALL: FW CCE got packet 0x660315C4 in process path
Feb 13 08:17:04: FIREWALL: Router gen or router destined pak 0x660315C4, let it pass
Feb 13 08:17:06: FIREWALL: FW CCE got packet 0x660328C0 in process path
Feb 13 08:17:06: FIREWALL: Router gen or router destined pak 0x660328C0, let it pass
Feb 13 08:17:07: FIREWALL: FW CCE got packet 0x66031D5C in process path
Feb 13 08:17:07: FIREWALL: Router gen or router destined pak 0x66031D5C, let it pass
Feb 13 08:17:08: FIREWALL: FW CCE got packet 0x66033424 in process path
Feb 13 08:17:08: FIREWALL: Router gen or router destined pak 0x66033424, let it pass
Feb 13 08:17:09: FIREWALL: FW CCE got packet 0x66032C8C in process path
Feb 13 08:17:09: FIREWALL: Router gen or router destined pak 0x66032C8C, let it pass
Feb 13 08:17:11: FIREWALL: FW CCE got packet 0x6602DCD0 in process path
Feb 13 08:17:11: FIREWALL: Router gen or router destined pak 0x6602DCD0, let it pass
Feb 13 08:17:11: FIREWALL: FW CCE got packet 0x5011DDB4 in process path
Feb 13 08:17:11: FIREWALL: Router gen or router destined pak 0x5011DDB4, let it pass
Feb 13 08:17:13: FIREWALL: FW CCE got packet 0x20159498 in process path
Feb 13 08:17:13: FIREWALL: Router gen or router destined pak 0x20159498, let it pass
Feb 13 08:17:13: FIREWALL: FW CCE got packet 0x665E5F38 in process path
Feb 13 08:17:13: FIREWALL: Router gen or router destined pak 0x665E5F38, let it pass
Feb 13 08:17:14: FIREWALL: fw_dp_insp_handle_timer_event
Feb 13 08:17:14: FIREWALL: fw_dp_insp_sample_session_rate
Feb 13 08:17:16: FIREWALL: FW CCE got packet 0x5011D9E8 in process path
Feb 13 08:17:16: FIREWALL: Router gen or router destined pak 0x5011D9E8, let it pass
Feb 13 08:17:16: FIREWALL: FW CCE got packet 0x20159864 in process path
Feb 13 08:17:16: FIREWALL: Router gen or router destined pak 0x20159864, let it pass
```

The following is sample output from the **debug policy-firewall protocol icq** command:

The event debug output declares the packet path from which the firewall got the packet. The packet path can be either Cisco Express Forwarding or the process path. The **debug policy-firewall** command is used when the firewall sends out a packet that acts like a proxy.

The timer debug output specifies timer-related events. Timers are used to close the sessions created by the firewall. Whenever a timeout happens, the timer debugging output specifies whether it needs to close the session or keep it open for longer.

```
Device# debug policy-firewall protocol icq

Apr  2 23:55:21: CCE*: I2R = 1, state_object = 0x0, data_len = 0
Apr  2 23:55:21: CCE*: ICQ protocol found...
Apr  2 23:55:21: CCE*: cce_dp_named_db_inspect_icq_create_cso
Apr  2 23:55:21: CCE*: I2R = 0, state_object = 0x508A1014, data_len = 10
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 1 , Packet length = 4
Apr  2 23:55:21: CCE*: I2R = 1, state_object = 0x508A1014, data_len = 270
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 1 , Packet length = 264
Apr  2 23:55:21: CCE*: ICQ:Find the client version
Apr  2 23:55:21: CCE*: ICQ:Get the client string
Apr  2 23:55:21: CCE*: ICQ:Object Type = 6,Object Length = 256
Apr  2 23:55:21: CCE*: icq_setstate_on_servicetype
Apr  2 23:55:21: CCE*: ICQ:Obj Data Skipping :prev state =4
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 0,Curr state = 1 , Prev state = 0
Apr  2 23:55:21: CCE*: I2R = 0, state_object = 0x508A1014, data_len = 42
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 2 , Packet length = 36
Apr  2 23:55:21: CCE*: ICQ:Family Service Id = 1,Subtype Id = 3
Apr  2 23:55:21: CCE*: ICQ:curr state = 9
Apr  2 23:55:21: CCE*: I2R = 1, state_object = 0x508A1014, data_len = 56
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 2 , Packet length = 50
Apr  2 23:55:21: CCE*: ICQ:Family Service Id = 1,Subtype Id = 23
Apr  2 23:55:21: CCE*: ICQ:curr state = 22
Apr  2 23:55:21: CCE*: ICQ:service = 1 , version = 4
Apr  2 23:55:21: CCE*: ICQ:service = 19 , version = 4
Apr  2 23:55:21: CCE*: ICQ:service = 2 , version = 1
Apr  2 23:55:21: CCE*: ICQ:service = 3 , version = 1
Apr  2 23:55:21: CCE*: ICQ:service = 21 , version = 1
Apr  2 23:55:21: CCE*: ICQ:Detected ICQ Protocol
Apr  2 23:55:21: CCE*: I2R = 1, state_object = 0x508A1014, data_len = 230
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 2 , Packet length = 224
Apr  2 23:55:21: CCE*: ICQ:Family Service Id = 4,Subtype Id = 6
Apr  2 23:55:21: CCE*: ICQ:curr state = 14
Apr  2 23:55:21: CCE*: icq_process_client_message
Apr  2 23:55:21: CCE*: ICQ:Message Channel ID = 2
Apr  2 23:55:21: CCE*: icq_skip_client_msg
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 5
Apr  2 23:55:21: CCE*: ICQ:length = 190,obj length = 186
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 4,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 3
Apr  2 23:55:21: CCE*: ICQ:length = 0,obj length = 0
Apr  2 23:55:21: CCE*: I2R = 1, state_object = 0x508A1014, data_len = 66
Apr  2 23:55:21: CCE*: ICQ:state = 21
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 2 , Packet length = 60
Apr  2 23:55:21: CCE*: ICQ:Family Service Id = 4,Subtype Id = 6
Apr  2 23:55:21: CCE*: ICQ:curr state = 14
Apr  2 23:55:21: CCE*: icq_process_client_message
Apr  2 23:55:21: CCE*: ICQ:Message Channel ID = 2
Apr  2 23:55:21: CCE*: icq_skip_client_msg
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 5
Apr  2 23:55:21: CCE*: ICQ:length = 26,obj length = 26
Apr  2 23:55:21: CCE*: ICQ:Obj Data Skipping :prev state =19
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 0,Curr state = 1 , Prev state = 0
Apr  2 23:55:21: CCE*: ICQ:service found = 2
Apr  2 23:55:21: CCE*: ICQ: Found IM default service
Apr  2 23:55:21: %APPFW-6-IM_ICQ_SESSION: im-icq un-recognized service session initiator
sends 66 bytes  session 192.168.5.3:25610 63.147.175.30:5190 on zone-pair zp_test_in class
 test_im appl-class test_icq_1
Apr  2 23:55:21: CCE*: I2R = 0, state_object = 0x508A1014, data_len = 36
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 2 , Packet length = 30
Apr  2 23:55:21: CCE*: ICQ:Family Service Id = 4,Subtype Id = 12
Apr  2 23:55:21: CCE*: ICQ:curr state = 9
Apr  2 23:55:21: CCE*: I2R = 0, state_object = 0x508A1014, data_len = 285
Apr  2 23:55:21: CCE*: ICQ:state = 1
Apr  2 23:55:21: CCE*: ICQ:FLAP Channel = 2 , Packet length = 279
Apr  2 23:55:21: CCE*: ICQ:Family Service Id = 4,Subtype Id = 7
Apr  2 23:55:21: CCE*: ICQ:curr state = 14
Apr  2 23:55:21: CCE*: icq_process_client_message
Apr  2 23:55:21: CCE*: ICQ:Message Channel ID = 2
Apr  2 23:55:21: CCE*: icq_skip_client_msg
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 1
Apr  2 23:55:21: CCE*: ICQ:length = 241,obj length = 2
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 239,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 6
Apr  2 23:55:21: CCE*: ICQ:length = 235,obj length = 4
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 231,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 5
Apr  2 23:55:21: CCE*: ICQ:length = 227,obj length = 4
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 223,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 15
Apr  2 23:55:21: CCE*: ICQ:length = 219,obj length = 4
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 215,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 3
Apr  2 23:55:21: CCE*: ICQ:length = 211,obj length = 4
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 207,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 5
Apr  2 23:55:21: CCE*: ICQ:length = 203,obj length = 190
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 13,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 22
Apr  2 23:55:21: CCE*: ICQ:length = 9,obj length = 4
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 5,Curr state = 19 , Prev state = 19
Apr  2 23:55:21: CCE*: ICQ:TLV Service Type = 19
Apr  2 23:55:21: CCE*: ICQ:length = 1,obj length = 1
Apr  2 23:55:21: CCE*: ICQ:Obj Data Skipping :prev state =19
Apr  2 23:55:21: CCE*: ICQ:ICQ Data length = 0,Curr state = 1 , Prev state = 0
Apr  2 23:56:10: CCE*: I2R = 1, state_object = 0x508A1014, data_len = 0
Apr  2 23:56:11: FIREWALL sis 65A1C100: Sis extension deleted
Apr  2 23:56:11: CCE: cce_dp_named_db_inspect_icq_delete_cso
```

The sample output from the **debug policy-firewall protocol winmsgr** command includes information about the instant messenger (IM) service. For example, the following lines declare that the type of IM service the user is running is Windows Messenger (WINMSGR):

The debug output details the different states that the state machine sees while parsing the Layer 7 I Seek You (ICQ) payload.

```
Apr  3 00:21:46: CCE*: WINMSGR:service found = 2
Apr  3 00:21:46: CCE*: WINMSGR: Found IM default service
```

The following is sample output from the **debug policy-firewall protocol winmsgr** command:

```
Device# debug policy-firewall protocol winmsgr

Apr  3 00:21:46: CCE*: I2R = 1, state_object = 0x0, data_len = 0
Apr  3 00:21:46: CCE*: WINMSGR protocol found...
Apr  3 00:21:46: CCE*: cce_dp_named_db_inspect_winmsgr_create_cso
Apr  3 00:21:46: CCE*: I2R = 1, state_object = 0x660CF5B4, data_len = 19
Apr  3 00:21:46: CCE*: WINMSGR:datalen=19,matchflag=11,matchlen=19
Apr  3 00:21:46: CCE*: WINMSGR:Initial trafficfound
Apr  3 00:21:46: CCE*: I2R = 0, state_object = 0x660CF5B4, data_len = 19
Apr  3 00:21:46: CCE*: WINMSGR:datalen=19,matchflag=11,matchlen=19
Apr  3 00:21:46: CCE*: WINMSGR:Initial trafficfound
Apr  3 00:21:46: CCE*: I2R = 1, state_object = 0x660CF5B4, data_len = 82
Apr  3 00:21:46: CCE*: WINMSGR:datalen=82,matchflag=6,matchlen=4
Apr  3 00:21:46: CCE*: WINMSGR:version msg : CVR 31 0x0409 winnt 5.0 i386 MSMSGS 5.1.0701
WindowsMessenger fwuser@example.com
Apr  3 00:21:46: CCE*: I2R = 0, state_object = 0x660CF5B4, data_len = 96
Apr  3 00:21:46: CCE*: WINMSGR:datalen=96,matchflag=6,matchlen=4
Apr  3 00:21:46: CCE*: I2R = 1, state_object = 0x660CF5B4, data_len = 33
Apr  3 00:21:46: CCE*: WINMSGR:datalen=33,matchflag=12,matchlen=33
Apr  3 00:21:46: CCE*: WINMSGR:Initial trafficfound
Apr  3 00:21:46: CCE*: I2R = 0, state_object = 0x660CF5B4, data_len = 162
Apr  3 00:21:46: CCE*: I2R = 1, state_object = 0x660CF5B4, data_len = 324
Apr  3 00:21:46: CCE*: I2R = 0, state_object = 0x660CF5B4, data_len = 37
Apr  3 00:21:46: CCE*: WINMSGR:datalen=37,matchflag=12,matchlen=37
Apr  3 00:21:46: CCE*: WINMSGR:Initial trafficfound
Apr  3 00:21:46: CCE*: I2R = 1, state_object = 0x660CF5B4, data_len = 307
Apr  3 00:21:46: CCE*: WINMSGR:datalen=307,matchflag=5,matchlen=118
Apr  3 00:21:46: CCE*: WINMSGR:service found = 2
Apr  3 00:21:46: CCE*: WINMSGR: Found IM default service
Apr  3 00:21:46: %APPFW-6-IM_WINMSGR_SESSION: im-winmsgr un-recognized service session
initiator sends 307 bytes  session 192.168.5.3:24601 209.165.200.230:1863 on zone-pair
zp_test_in class test_im appl-class test_winmsgr_1
Apr  3 00:21:46: CCE*: I2R = 0, state_object = 0x660CF5B4, data_len = 320
Apr  3 00:21:46: CCE*: I2R = 0, state_object = 0x660CF5B4, data_len = 332
Apr  3 00:21:46: CCE*: WINMSGR:datalen=332,matchflag=5,matchlen=143
Apr  3 00:21:46: CCE*: WINMSGR:service found = 2
Apr  3 00:21:46: CCE*: WINMSGR: Found IM default service
Apr  3 00:21:46: %APPFW-6-IM_WINMSGR_SESSION: im-winmsgr un-recognized service session
initiator gets 332 bytes  session 209.165.200.230:1863 192.168.5.3:24601 on zone-pair
zp_test_in class test_im appl-class test_winmsgr_1
Apr  3 00:23:11: CCE*: I2R = 1, state_object = 0x660CF5B4, data_len = 0
Apr  3 00:23:11: FIREWALL sis 65A1D540: Sis extension deleted
```

The following is sample output from the **debug policy-firewall control-plane** command:

```
Device# debug policy-firewall control-plane

policy_fw:
  Policy-Firewall control-plane debugging is on
voice-gw-118.03#
Syslog logging: enabled (0 messages dropped, 0 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering disabled)
No Active Message Discriminator.
No Inactive Message Discriminator.
    Console logging: disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging:  level debugging, 247 messages logged, xml disabled,
                     filtering disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled
    Persistent logging: disabled
    Trap logging: level informational, 44 message lines logged
Log Buffer (60000000 bytes):
FIREWALL CP: fw_cp_prot_num_to_name()  l4 1, l7 5, gran 0
FIREWALL CP: fw_cp_get_flow_policy_and_class()  Flow policy does not exist
FIREWALL CP: fw_cp_check_create_default_l7_policy()  Could not retrieve flow policy for L4
 policy l4-pmap L4 class l4-cmap
FIREWALL CP: fw_classmap_filter_update_in_policymap()  Adding filter 0x650187F0 to class
l4-cmap in policy l4-pmap
FIREWALL CP: fw_policy_action_cmd()  PPM create action inspect with params 0x64CAF8E8
FIREWALL CP: fw_inspect_class_params()  inspect config-plane CLASS-ADD action
0x66315C5C,params 0x64CAF8E8
FIREWALL CP: fw_validate_class_for_matchprot()  Validating protocols in class l4-cmap
FIREWALL CP: fw_validate_class_for_matchprot()  protocol filter found
FIREWALL CP: fw_inspect_class_params()  Attached config-plane action_params 0x663BD280
FIREWALL CP: fw_cp_create_attach_flow_policy()
FIREWALL CP: fw_cp_get_string_from_random_num()  Random number generated is 2697258553
FIREWALL CP: fw_cp_generate_random_string()  Allocated random str 2697258553 for policy
l4-pmap class l4-cmap
FIREWALL CP: fw_cp_get_random_string()  Found random string  for policy l4-pmap class l4-cmap
FIREWALL CP: fw_cp_get_random_string()  Found random string  for policy l4-pmap class l4-cmap
FIREWALL CP: fw_cp_get_random_string()  Found random string  for policy l4-pmap class l4-cmap
FIREWALL CP: fw_cp_prot_num_to_name()  l4 2, l7 5, gran 0
FIREWALL CP: fw_inspect_int_class_params()
FIREWALL CP: fw_create_attach_template_class()
FIREWALL CP: fw_create_attach_template_class()  Creating template class for trigger
15udp_2697258553 in 15_2697258553
FIREWALL CP: fw_create_attach_template_class()  Trying to create a PPM filter with id
0x64CA73EC
FIREWALL CP: fw_cp_prot_num_to_name()  l4 4, l7 5, gran 0
FIREWALL CP: fw_inspect_int_class_params()
FIREWALL CP: fw_create_attach_template_class()
FIREWALL CP: fw_create_attach_template_class()  Creating template class for trigger
15icmp_2697258553 in 15_2697258553
FIREWALL CP: fw_create_attach_template_class()  Trying to create a PPM filter with id
0x64CA73EC
FIREWALL CP: fw_cp_create_attach_vtcp_classes()  Create policy 15
FIREWALL CP: fw_cp_create_tcp_15()
FIREWALL CP: fw_cp_vtcp_support_get_tcp_init_class()  Creating TCP Class with Pure SYN
filter
FIREWALL CP: fw_inspect_int_class_params()
FIREWALL CP: fw_create_attach_template_class()
FIREWALL CP: fw_create_attach_template_class()  Creating template class for trigger
15tcp_2697258553 in 15_2697258553
FIREWALL CP: fw_create_attach_template_class()  Trying to create a PPM filter with id
0x64CA73A4
FIREWALL CP: fw_cp_create_attach_flow_policy()  Success-creating flow policy
FIREWALL CP: fw_cp_create_attach_flow_policy()  Attach flow policy to trigger class as child
 policy
FIREWALL CP: fw_cp_create_attach_flow_policy()  Success- Attached flow policy to trigger
class
FIREWALL CP: fw_cp_create_attach_flow_policy()  Creating P20 & P21 for vtcp
FIREWALL CP: fw_cp_generate_random_string()  Found random string  for policy l4-pmap class
 l4-cmap
FIREWALL CP: fw_cp_get_flow_policy_and_class()  Found flow policy 0x64FFC838
FIREWALL CP: fw_cp_get_random_string()  Found random string  for policy l4-pmap class l4-cmap
FIREWALL CP: fw_cp_get_random_string()  Found random string  for policy l4-pmap class l4-cmap
FIREWALL CP: fw_cp_get_flow_policy_and_class()  Found flow TCP 0x6585718C and UDP 0x645D1794
 classes
FIREWALL CP: fw_cp_check_create_default_l7_class()  Checking the class l4-cmap
FIREWALL CP: fw_reverse_policy_handle_zp_event()
FIREWALL CP: fw_reverse_policy_handle_zp_event()  Reverse_policy Zone pair add event
FIREWALL CP: fw_get_ppm_policy_on_zp()  Did not find ppm policy on zp zp p_type 0x7
FIREWALL CP: fw_get_name_type_and_client_of_first_class_in_policy()
FIREWALL CP: fw_create_cp_dynamic_class()
FIREWALL CP: fw_create_cp_dynamic_class()  Trying to create a PPM filter with id 0x10000000
FIREWALL CP: fw_create_cp_dynamic_class()  Success
FIREWALL CP: fw_drop_class_params()  action 0x6637A5C0, cmd_params 0x64CA7550, event 0x21
FIREWALL CP: fw_create_noop_feature_object()
FIREWALL CP: fw_create_inspect_feature_object()
FIREWALL CP: fw_create_fo_internal()  Create FO for class 0xC0000002 target_class 0xA0000000
 action CCE_INSPECT_CONFIGURED
FIREWALL CP: fw_cp_get_inspect_params()
FIREWALL CP: fw_cp_get_inspect_params()  Creating the FO with default parameters
FIREWALL CP: fw_create_fo_internal()  Created FO with id 0xAAAA0006 action
CCE_INSPECT_CONFIGURED
FIREWALL CP: fw_cp_store_fo_id()  Enqueue  0xAAAA0006 to fo_param_list
FIREWALL CP: fw_create_noop_feature_object()
FIREWALL CP: fw_create_inspect_int_feature_object()
FIREWALL CP: fw_create_fo_internal()  Create FO for class 0xC0000005 target_class 0xA0000000
 action CCE_INSPECT
FIREWALL CP: fw_cp_get_inspect_params()
FIREWALL CP: fw_cp_get_inspect_params()  Creating the FO with default parameters
FIREWALL CP: fw_create_fo_internal()  Created FO with id 0xAAAA0007 action CCE_INSPECT
FIREWALL CP: fw_cp_store_fo_id()  Enqueue  0xAAAA0007 to fo_param_list
FIREWALL CP: fw_create_noop_feature_object()
FIREWALL CP: fw_create_inspect_int_feature_object()
FIREWALL CP: fw_create_fo_internal()  Create FO for class 0xC0000007 target_class 0xA0000000
 action CCE_INSPECT
FIREWALL CP: fw_cp_get_inspect_params()
FIREWALL CP: fw_cp_get_inspect_params()  Creating the FO with default parameters
FIREWALL CP: fw_create_fo_internal()  Created FO with id 0xAAAA0008 action CCE_INSPECT
FIREWALL CP: fw_cp_store_fo_id()  Enqueue  0xAAAA0008 to fo_param_list
FIREWALL CP: fw_create_noop_feature_object()
FIREWALL CP: fw_create_inspect_int_feature_object()
FIREWALL CP: fw_create_fo_internal()  Create FO for class 0xC0000009 target_class 0xA0000000
 action CCE_INSPECT
FIREWALL CP: fw_cp_get_inspect_params()
FIREWALL CP: fw_cp_get_inspect_params()  Creating the FO with default parameters
FIREWALL CP: fw_create_fo_internal()  Created FO with id 0xAAAA0009 action CCE_INSPECT
FIREWALL CP: fw_cp_store_fo_id()  Enqueue  0xAAAA0009 to fo_param_list
FIREWALL CP: fw_create_drop_feature_object()
FIREWALL CP: fw_create_fo_internal()  Create FO for class 0xC0000003 target_class 0xA0000000
 action CCE_FW_DROP
FIREWALL CP: fw_create_fo_internal()  Created FO with id 0xAAAA000A action CCE_FW_DROP
FIREWALL CP: fw_create_internal_reverse_policy()
FIREWALL CP: fw_create_ppm_reverse_policy()
FIREWALL CP: fw_get_name_type_and_client_of_first_class_in_policy()
FIREWALL CP: fw_create_cp_dynamic_class()
FIREWALL CP: fw_create_noop_feature_object()
FIREWALL CP: fw_create_noop_feature_object()
%SYS-5-CONFIG_I: Configured from console by console
FIREWALL CP: fw_cp_prot_num_to_name()  l4 1, l7 5, gran 0
FIREWALL CP: fw_drop_class_params()  action 0x6637A5C0, cmd_params 0x00000000, event 0x40
FIREWALL CP: fw_get_ppm_policy_on_zp()  Found ppm policy l4-pmap on zp zp p_type 0x7
```

The following is sample output from the **debug policy-firewall L2-transparent** command:

```
Device# debug policy-firewall L2-transparent

*Apr  4 08:28:23.554: L2FW*:insp_l2_fast_inspection: pak 673DBD90, input-interface
FastEthernet1/1, output-interface FastEthernet1/0
*Apr  4 08:28:23.554: L2FW*:Src 17.3.39.1 dst 17.3.39.3 protocol tcp
*Apr  4 08:28:23.554: TBAP: Check AuthProxy is configured on idb=FastEthernet1/1 path=1
linktype=38
*Apr  4 08:28:23.554: L2FW:Input ACL not configured or the ACL is bypassed
*Apr  4 08:28:23.554: L2FW:Output ACL is not configured or ACL is bypassed
*Apr  4 08:28:23.554: L2FW*:IP inspect firewall is not cfged on  input or output
interface.PASS
*Apr  4 08:28:23.554: L2FW* 2:insp_l2_fast_inspection: pak 673DBD90, input-interface
FastEthernet1/1, output-interface FastEthernet1/0
*Apr  4 08:28:23.554:  CCE L2 FW
*Apr  4 08:28:23.554: L2FW* -3:insp_l2_fast_inspection: pak 673DBD90, input-interface
FastEthernet1/1, output-interface FastEthernet1/0
```

The following is sample output from the **debug policy-firewall detailed** command:

```
Device# debug policy-firewall detailed

Log Buffer (600000 bytes):
Feb 13 08:40:01: FIREWALL: ret_val 0 is not FW_DP_INSP_PASS_PAK
<snip>
```

```
Feb 13 08:41:22: FIREWALL: ret_val 0 is not FW_DP_INSP_PASS_PAK
Feb 13 08:41:24: FIREWALL: ret_val 0 is not FW_DP_INSP_PASS_PAK
Feb 13 08:41:25: FIREWALL*: Searching for FSO in class 0x50793C20class group 0x10000000,
target 0x1, cce class type 0x2B
Feb 13 08:41:25: FIREWALL*: not found
Feb 13 08:41:25: FIREWALL*: Try to create session in fastpath
Feb 13 08:41:25: FIREWALL: Searching for FSO in class 0x50793C20class group 0x10000000,
target 0x1, cce class type 0x2B
Feb 13 08:41:25: FIREWALL: not found
Feb 13 08:41:25: FIREWALL: Create FSO
Feb 13 08:41:25: FIREWALL: sis 204925C0 : fw_dp_state_object_link
Feb 13 08:41:25: FIREWALL: sis 204925C0 : FO class 0x50793C20 class group 0x10000000, target
 0x1, FO 0x20255D80
Feb 13 08:41:25: FIREWALL: sis 204925C0 : alert = 1, audit_trail = 0
Feb 13 08:41:25: FIREWALL: sis 204925C0 : l7 protocol 62, granular = 5
Feb 13 08:41:25: FIREWALL: sis 204925C0 : fw_dp_state_object_attach_forward
Feb 13 08:41:25: FIREWALL: sis 204925C0 : fw_dp_state_object_create_and_attach_reverse
Feb 13 08:41:25: FIREWALL: sis 204925C0 : FSO bind success for reverse class 0x50793C80class
 group 0x10000000, target 0x1
Feb 13 08:41:25: FIREWALL: sis 204925C0 :Session Info :
Feb 13 08:41:25: session->fwfo 0x507E39C0
Feb 13 08:41:25: class type 0x2B, target 0x1, policy id 0x10000000, class id 0x50793C20
Feb 13 08:41:25: class type 0x2B, reverse target 0x1, reverse policy id 0x10000000, reverse
 class id 0x50793C80
Feb 13 08:41:25: src addr 192.168.3.3, port 36091, vrf id 0
Feb 13 08:41:25: dst addr 192.168.103.3, port 5190, vrf id 0
Feb 13 08:41:25: L4 Protocol : TCP
Feb 13 08:41:25: FIREWALL: sis 204925C0 : L4 inspection returned 3
Feb 13 08:41:25: FIREWALL*: FSO feature object 0x204925C0 found
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : L4 inspection returned 3
Feb 13 08:41:25: FIREWALL*: FSO feature object 0x204925C0 found
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : max_sessions 2147483647; current sessions 0
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : IM : Token set for L7 named-db
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : cce_sb 0x66A5BA00, pak 0x50028974, data_len 0
in_fast_path 1, dir = 1
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : p_app_data = C174268, p_data_len = 6p_offset =
0
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : Found particle offset token, data1 = 0
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : Opening 0 channels for icq
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : icq L7 inspect result: PASS packet
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : L4 inspection returned 3
Feb 13 08:41:25: FIREWALL*: FSO feature object 0x204925C0 found
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : cce_sb 0x66A5BA00, pak 0x5004CAC8, data_len 10
in_fast_path 1, dir = 2
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : p_app_data = C210848, p_data_len = Ap_offset =
0
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : Found particle offset token, data1 = 0
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : Opening 0 channels for icq
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : icq L7 inspect result: PASS packet
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : L4 inspection returned 3
Feb 13 08:41:25: FIREWALL*: FSO feature object 0x204925C0 found
Feb 13 08:41:25: FIREWALL*: sis 204925C0 : cce_sb 0x66A5BA00, pak 0x50028974, data_len 270
 in_fast_path 1, dir = 1
```

The following is sample output from the **debug policy-firewall ha** command

```
Device# debug policy-firewall ha

*May 19 14:17:19.991: FIREWALL: IOS FW RF stat event: status: RF_STATUS_PEER_COMM
my state: STANDBY HOT peer state: ACTIVE
*May 19 14:17:19.995: FIREWALL: IOS FW RF stat event: status: RF_STATUS_PEER_PRESENCE
my state: STANDBY HOT peer state: DISABLED
*May 19 14:17:19.995: FIREWALL: RG with ID:1 state STANDBY: found
*May 19 14:17:19.995: FIREWALL: Event for RG-1: RF_PROG_ACTIVE_FAST
*May 19 14:17:19.995: FIREWALL: RG with ID:1 state ACTIVE: found
*May 19 14:17:19.995: FIREWALL: Standbyhot to Active transition for RG 1
*May 19 14:17:19.995: FIREWALL sis 30CEEF40: Timer Start: Timer: 30CEEFD4 Time: 30000 ms
*May 19 14:17:19.995: FIREWALL: RG 1 trasitioned to Active
*May 19 14:17:19.995: FIREWALL: RG with ID:1 state ACTIVE: found
*May 19 14:17:19.995: FIREWALL: RG with ID:1 state ACTIVE: found
*May 19 14:17:19.995: FIREWALL: RG with ID:1 state ACTIVE: found
May 19 14:17:30.003: FIREWALL: Event for RG-1: RF_PROG_STANDBY_BULK Configuring Zone Based
 Firewall Redundancy Draft Copy Cisco systems, Inc. Company Confidential
*May 19 14:17:30.003: FIREWALL: ret_val 0 is not PASS_PAK
```

```
*May 19 14:17:30.003: FIREWALL: RG with ID:1 state ACTIVE: found
*May 19 14:17:30.003: FIREWALL: Starting BulkSync for RG 1
*May 19 14:17:30.003: FIREWALL sis 30CEEF40: Bulk sync session 30CEEF40 needs to be failed
 over(add)
*May 19 14:17:30.003: FIREWALL: ret_val 0 is not PASS_PAK
*May 19 14:17:30.003: FIREWALL sis 30CEEF40: Send add session message
(192.168.7.205:32424:0)=>(192.168.107.1:23:0) l4_prot tcp
*May 19 14:17:30.003: FIREWALL: BulkSync done; Send BulkEnd
```

debug policy-firewall mib

To toggle on or off the support for MIBs in a zone-based policy firewall, use the **debug policy-firewall mib** command in privileged EXEC mode. To disable the MIB support, use the **no** form of this command.

debug policy-firewall mib {**event**| **object-creation**| **object-deletion**| **object-retrieval**}

no debug policy-firewall mib {**event**| **object-creation**| **object-deletion**| **object-retrieval**}

Syntax Description

event	Turns on debugging for a firewall MIB event.
object-creation	Turns on debugging for a firewall MIB object creation.
object-deletion	Turns on debugging for a firewall MIB object deletion.
object-retrieval	Turns on debugging for a firewall MIB object retrieval.

Command Default

Privileged EXEC (#)

Command History

Release	Modification
15.1(1)T	This command was introduced.

Usage Guidelines

This command provides debug support for MIBs in zone-based policy firewall similar to the Cisco IOS firewall.

Examples

The following is a sample output from the **debug policy-firewall mib object-retrieval** command:

```
Router# debug policy-firewall mib object-retrieval
Firewall MIB object retrieval debugging is on
```

debug port-channel load-balance

To enable debug output for port-channel load balancing, use the **debug port-channel load-balance** command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

debug port-channel load-balance {**all**| **manual**| **weighted**}

no debug port-channel load-balance {**all**| **manual**| **weighted**}

Syntax Description

all	Turns on debugging for all load-balancing operations.
manual	Turns on debugging for only manual load-balancing operations.
weighted	Turns on debugging for only weighted load-balancing operations.

Command Default

Port-channel debugging is turned off.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.0(1)S	This command was introduced.

Usage Guidelines

Use this command to help troubleshoot load balancing of service instances over port-channel member links.

Examples

The following example shows how to enable debugging for only weighted load-balancing operations:

```
Router# debug port-channel load-balance weighted
Port-channel Load-Balance Weighted debugging is on
```

debug pots

To display information on the telephone interfaces, use the **debug pots**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug pots {**driver**| **csm**} [**1**| **2**]

no debug pots {**driver**| **csm**} [**1**| **2**]

Syntax Description

driver	Displays driver debug information.
csm	Displays Content Switching Module (CSM) debug information.
1	(Optional) Displays information for telephone port 1 only.
2	(Optional) Displays information for telephone port 2 only.

Command Modes

Privileged EXEC

Usage Guidelines

The **debug pots** command displays driver and CSM debug information for telephone ports 1 and 2.

Examples

The following is sample output from the **debug pots driver 1** command. This sample display indicates that the telephone port driver is not receiving caller ID information from the ISDN line. Therefore, the analog caller ID device attached to the telephone port does not display caller ID information.

```
Router# debug pots driver 1
00:01:51:POTS DRIVER port=1 activate ringer: cadence=0 callerId=Unknown
00:01:51:POTS DRIVER port=1 state=Idle drv_event=RING_EVENT
00:01:51:POTS DRIVER port=1 enter_ringing
00:01:51:POTS DRIVER port=1 cmd=19
00:01:51:POTS DRIVER port=1 activate disconnect
00:01:51:POTS DRIVER port=1 state=Ringing drv_event=DISCONNECT_EVENT
00:01:51:POTS DRIVER port=1 cmd=1A
00:01:51:POTS DRIVER port=1 enter_idle
00:01:51:POTS DRIVER port=1 ts connect: 0 0
00:01:51:POTS DRIVER port=1 cmd=D
00:01:51:POTS DRIVER port=1 report onhook
00:01:51:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:01:51:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:01:51:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:01:51:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:01:53:POTS DRIVER port=1 activate ringer: cadence=0 callerId=Unknown
00:01:53:POTS DRIVER port=1 state=Idle drv_event=RING_EVENT
00:01:53:POTS DRIVER port=1 enter_ringing
00:01:53:POTS DRIVER port=1 cmd=19
00:01:55:POTS DRIVER port=1 cmd=1A
00:02:49:POTS DRIVER port=1 state=Ringing drv_event=OFFHOOK_EVENT
00:02:49:POTS DRIVER port=1 cmd=1A
```

```
00:02:49:POTS DRIVER port=1 enter_suspend
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 report offhook
00:02:49:POTS DRIVER port=1 activate connect: endpt=1 calltype=TWO_PARTY_CALL
00:02:49:POTS DRIVER port=1 state=Suspend drv_event=CONNECT_EVENT
00:02:49:POTS DRIVER port=1 enter_connect: endpt=1 calltype=0
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 ts connect: 1 0
00:02:49:POTS DRIVER port=1 activate connect: endpt=1 calltype=TWO_PARTY_CALL
00:02:49:POTS DRIVER port=1 state=Connect drv_event=CONNECT_EVENT
00:02:49:POTS DRIVER port=1 enter_connect: endpt=1 calltype=0
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 ts connect: 1 0
00:02:55:POTS DRIVER port=1 state=Connect drv_event=ONHOOK_EVENT
00:02:55:POTS DRIVER port=1 enter_idle
00:02:55:POTS DRIVER port=1 ts connect: 0 0
00:02:55:POTS DRIVER port=1 cmd=D
00:02:55:POTS DRIVER port=1 report onhook
00:02:55:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:02:55:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:02:55:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:02:55:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
```

The following is sample output from the **debug pots csm 1** command. This sample display indicates that a dial peer contains an invalid destination pattern (555-1111).

```
Router# debug pots csm 1
01:57:28:EVENT_FROM_ISDN:dchanidb=0x66CB38, call_id=0x11, ces=0x2 bchan=0x0, event=0x1,
cause=0x0
01:57:28:Dial peer not found, route call to port 1
01:57:28:CSM_PROC_IDLE:CSM_EVENT_ISDN_CALL, call_id=0x11, port=1
01:57:28:Calling number ‘5551111’
01:57:40:CSM_PROC_RINGING:CSM_EVENT_VDEV_OFFHOOK, call_id=0x11, port=1
01:57:40:EVENT_FROM_ISDN:dchan_idb=0x66CB38, call_id=0x11, ces=0x2 bchan=0x0, event=0x4,
cause=0x0
01:57:40:CSM_PROC_CONNECTING:CSM_EVENT_ISDN_CONNECTED, call_id=0x11, port=1
01:57:47:CSM_PROC_CONNECTING:CSM_EVENT_VDEV_ONHOOK, call_id=0x11, port=1
01:57:201863503872: %ISDN-6-DISCONNECT:Interface BRI0:1 disconnected from unknown, call
lasted 5485 seconds
01:57:47: %ISDN-6-DISCONNECT:Interface BRI0:1 disconnected from unknown, call lasted 5485
seconds
01:57:47:EVENT_FROM_ISDN:dchan _idb=0x66CB38, call_id=0x11, ces=0x2 bchan=0xFFFFFFFF,
event=0x0, cause=0x1
01:57:47:CSM_PROC_NEAR_END_DISCONNECT:CSM_
```

debug pots csm

To activate events from which an application can determine and display the status and progress of calls to and from plain old telephone service (POTS) ports, use the **debugpotscsm**command in privileged EXEC mode.

debug pots csm

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1.(2)XF	This command was introduced on the Cisco 800 series routers.

Examples

To see debugging messages, enter the **loggingconsole**global configuration mode command as follows:

```
Router(config)# logging console
Router(config)# exit
```

Debugging messages are displayed in one of two formats that are relevant to the POTS dial feature:

```
hh:mm:ss: CSM_STATE: CSM_EVENT, call id = ??, port = ?
```

or

```
hh:mm:ss: EVENT_FROM_ISDN:dchan_idb=0x???????, call_id=0x????, ces=? bchan=0x????????,
event=0x?, cause=0x??
```

The following table describes the significant fields shown in the display.

Table 32: debug pots csm Field Descriptions

Command Elements	Description
hh:mm:ss	Timestamp (in hours, minutes, and seconds).
CSM_STATE	One of the call CSM states listed in the field description table.
CSM_EVENT	One of the CSM events listed in the field description table.
call id	Hexadecimal value from 0x00 to 0xFF.
port	Telephone port 1 or 2.

Command Elements	Description
EVENT_FROM_ISDN	A CSM event. The table shows a list of CSM events.
dchan_idb	Internal data structure address.
ces	Connection end point suffix used by ISDN.
bchan	Channel used by the call. A value of 0xFFFFFFFF indicates that a channel is not assigned.
event	A hexadecimal value that is translated into a CSM event. The field description table shows a list of events and the corresponding CSM events.
cause	A hexadecimal value that is given to call-progressing events. The field description table shows a list of cause values and definitions.

The following table shows the values for CSM states.

Table 33: CSM States

CSM State	Description
CSM_IDLE_STATE	Telephone on the hook.
CSM_RINGING	Telephone ringing.
CSM_SETUP	Setup for outgoing call in progress.
CSM_DIALING	Dialing number of outgoing call.
CSM_IVR_DIALING	Interactive voice response (IVR) for Japanese telephone dialing.
CSM_CONNECTING	Waiting for carrier to connect the call.
CSM_CONNECTED	Call connected.
CSM_DISCONNECTING	Waiting for carrier to disconnect the call.
CSM_NEAR_END_DISCONNECTING	Waiting for carrier to disconnect the call.
CSM_HARD_HOLD	Call on hard hold.
CSM_CONSULTATION_HOLD	Call on consultation hold.
CSM_WAIT_FOR_HOLD	Waiting for carrier to put call on hard hold.

CSM State	Description
CSM_WAIT_FOR_CONSULTATION_HOLD	Waiting for carrier to put call on consultation hold.
CSM_CONFERENCE	Waiting for carrier to complete call conference.
CSM_TRANSFER	Waiting for carrier to transfer call.
CSM_APPLIC_DIALING	Call initiated from Cisco IOS command-line interface (CLI).

The following table shows the values for CSM events.

Table 34: CSM Events

CSM Events	Description
CSM_EVENT_INTER_DIGIT_TIMEOUT	Time waiting for dial digits has expired.
CSM_EVENT_TIMEOUT	Near- or far-end disconnect timeout.
CSM_EVENT_ISDN_CALL	Incoming call.
CSM_EVENT_ISDN_CONNECTED	Call connected.
CSM_EVENT_ISDN_DISCONNECT	Far end disconnected.
CSM_EVENT_ISDN_DISCONNECTED	Call disconnected.
CSM_EVENT_ISDN_SETUP	Outgoing call requested.
CSM_EVENT_ISDN_SETUP_ACK	Outgoing call accepted.
CSM_EVENT_ISDN_PROC	Call proceeding and dialing completed.
CSM_EVENT_ISDN_CALL_PROGRESSING	Call being received in band tone.
CSM_EVENT_ISDN_HARD_HOLD	Call on hard hold.
CSM_EVENT_ISDN_HARD_HOLD_REJ	Hold attempt rejected.
CSM_EVENT_ISDN_CHOLD	Call on consultation hold.
CSM_EVENT_ISDN_CHOLD_REJ	Consultation hold attempt rejected.
CSM_EVENT_ISDN_RETRIEVED	Call retrieved.
CSM_EVENT_ISDN_RETRIEVE_REJ	Call retrieval attempt rejected.

CSM Events	Description
CSM_EVENT_ISDN_TRANSFERRED	Call transferred.
CSM_EVENT_ISDN_TRANSFER_REJ	Call transfer attempt rejected.
CSM_EVENT_ISDN_CONFERENCE	Call conference started.
CSM_EVENT_ISDN_CONFERENCE_REJ	Call conference attempt rejected.
CSM_EVENT_ISDN_IF_DOWN	ISDN interface down.
CSM_EVENT_ISDN_INFORMATION	ISDN information element received (used by NTT IVR application).
CSM_EVENT_VDEV_OFFHOOK	Telephone off the hook.
CSM_EVENT_VDEV_ONHOOK	Telephone on the hook.
CSM_EVENT_VDEV_FLASHHOOK	Telephone hook switch has flashed.
CSM_EVENT_VDEV_DIGIT	DTMF digit has been detected.
CSM_EVENT_VDEV_APPLICATION_CALL	Call initiated from Cisco IOS CLI.

The following table shows the values for events that are translated into CSM events.

Table 35: Event Values

Hexadecimal Value	Event	CSM Event
0x0	DEV_IDLE	CSM_EVENT_ISDN_DISCONNECTED
0x1	DEV_INCALL	CSM_EVENT_ISDN_CALL
0x2	DEV_SETUP_ACK	CSM_EVENT_ISDN_SETUP_ACK
0x3	DEV_CALL_PROC	CSM_EVENT_ISDN_PROC
0x4	DEV_CONNECTED	CSM_EVENT_ISDN_CONNECTED
0x5	DEV_CALL_PROGRESSING	CSM_EVENT_ISDN_CALL_PROGRESSING
0x6	DEV_HOLD_ACK	CSM_EVENT_ISDN_HARD_HOLD
0x7	DEV_HOLD_REJECT	CSM_EVENT_ISDN_HARD_HOLD_REJ
0x8	DEV_CHOLD_ACK	CSM_EVENT_ISDN_CHOLD

Hexadecimal Value	Event	CSM Event
0x9	DEV_CHOLD_REJECT	CSM_EVENT_ISDN_CHOLD_REJ
0xa	DEV_RETRIEVE_ACK	CSM_EVENT_ISDN_RETRIEVED
0xb	DEV_RETRIEVE_REJECT	CSM_EVENT_ISDN_RETRIEVE_REJ
0xc	DEV_CONFR_ACK	CSM_EVENT_ISDN_CONFERENCE
0xd	DEV_CONFR_REJECT	CSM_EVENT_ISDN_CONFERENCE_REJ
0xe	DEV_TRANS_ACK	CSM_EVENT_ISDN_TRANSFERRED
0xf	DEV_TRANS_REJECT	CSM_EVENT_ISDN_TRANSFER_REJ

The following table shows cause values that are assigned only to call-progressing events.

Table 36: Cause Values

Hexadecimal Value	Cause Definitions
0x01	UNASSIGNED_NUMBER
0x02	NO_ROUTE
0x03	NO_ROUTE_DEST
0x04	NO_PREFIX
0x06	CHANNEL_UNACCEPTABLE
0x07	CALL_AWARDED
0x08	CALL_PROC_OR_ERROR
0x09	PREFIX_DIALED_ERROR
0x0a	PREFIX_NOT_DIALED
0x0b	EXCESSIVE_DIGITS
0x0d	SERVICE_DENIED
0x10	NORMAL_CLEARING
0x11	USER_BUSY
0x12	NO_USER_RESPONDING

Hexadecimal Value	Cause Definitions
0x13	NO_USER_ANSWER
0x15	CALL_REJECTED
0x16	NUMBER_CHANGED
0x1a	NON_SELECTED_CLEARING
0x1b	DEST_OUT_OF_ORDER
0x1c	INVALID_NUMBER_FORMAT
0x1d	FACILITY_REJECTED
0x1e	RESP_TO_STAT_ENQ
0x1f	UNSPECIFIED_CAUSE
0x22	NO_CIRCUIT_AVAILABLE
0x26	NETWORK_OUT_OF_ORDER
0x29	TEMPORARY_FAILURE
0x2a	NETWORK_CONGESTION
0x2b	ACCESS_INFO_DISCARDED
0x2c	REQ_CHANNEL_NOT_AVAIL
0x2d	PRE_EMPTED
0x2f	RESOURCES_UNAVAILABLE
0x32	FACILITY_NOT_SUBSCRIBED
0x33	BEARER_CAP_INCOMPAT
0x34	OUTGOING_CALL_BARRED
0x36	INCOMING_CALL_BARRED
0x39	BEARER_CAP_NOT_AUTH
0x3a	BEAR_CAP_NOT_AVAIL
0x3b	CALL_RESTRICTION

Hexadecimal Value	Cause Definitions
0x3c	REJECTED_TERMINAL
0x3e	SERVICE_NOT_ALLOWED
0x3f	SERVICE_NOT_AVAIL
0x41	CAP_NOT_IMPLEMENTED
0x42	CHAN_NOT_IMPLEMENTED
0x45	FACILITY_NOT_IMPLEMENT
0x46	BEARER_CAP_RESTRICTED
0x4f	SERV_OPT_NOT_IMPLEMENT
0x51	INVALID_CALL_REF
0x52	CHAN_DOES_NOT_EXIST
0x53	SUSPENDED_CALL_EXISTS
0x54	NO_CALL_SUSPENDED
0x55	CALL_ID_IN_USE
0x56	CALL_ID_CLEARED
0x58	INCOMPATIBLE_DEST
0x5a	SEGMENTATION_ERROR
0x5b	INVALID_TRANSIT_NETWORK
0x5c	CS_PARAMETER_NOT_VALID
0x5f	INVALID_MSG_UNSPEC
0x60	MANDATORY_IE_MISSING
0x61	NONEXISTENT_MSG
0x62	WRONG_MESSAGE
0x63	BAD_INFO_ELEM
0x64	INVALID_ELEM_CONTENTS

Hexadecimal Value	Cause Definitions
0x65	WRONG_MSG_FOR_STATE
0x66	TIMER_EXPIRY
0x67	MANDATORY_IE_LEN_ERR
0x6f	PROTOCOL_ERROR
0x7f	INTERWORKING_UNSPEC

Examples

This section provides debug output examples for three call scenarios, displaying the sequence of events that occur during a POTS dial call or POTS disconnect call.

Examples

In this example call scenario, port 1 is on the hook, the application dial is set to call 4085552221, and the far-end successfully connects.

```
Router# debug pots csm
Router# test pots 1 dial 4085552221#
Router#
```

The following output shows an event indicating that port 1 is being used by the dial application:

```
01:58:27: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
```

The following output shows events indicating that the CSM is receiving the application digits of the number to dial:

```
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
```

The following output shows that the telephone connected to port 1 is off the hook:

```
01:58:39: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
```

The following output shows a call-proceeding event pair indicating that the router ISDN software has sent the dialed digits to the ISDN switch:

```
01:58:40: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0x0, event=0x3,
 cause=0x0
01:58:40: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id =
0x8004, port = 1
```

The following output shows the call-progressing event pair indicating that the telephone at the far end is ringing:

```
01:58:40: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x0
01:58:40: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8004, port
 = 1
```

The following output shows a call-connecting event pair indicating that the telephone at the far end has answered:

```
01:58:48: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x4, cause=0x0
01:58:48: CSM_PROC_CONNECTING: CSM_EVENT_ISDN_CONNECTED, call id = 0x8004, port = 1
```

The following output shows a call-progressing event pair indicating that the telephone at the far end has hung up and that the calling telephone is receiving an in-band tone from the ISDN switch:

```
01:58:55: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x10
01:58:55: CSM_PROC_CONNECTED: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8004, port = 1
```

The following output shows that the telephone connected to port 1 has hung up:

```
01:58:57: CSM_PROC_CONNECTED: CSM_EVENT_VDEV_ONHOOK, call id = 0x8004, port = 1
```

The following output shows an event pair indicating that the call has been terminated:

```
01:58:57: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x0, cause=0x0
01:58:57: CSM_PROC_NEAR_END_DISCONNECT: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8004, port
 = 1
813_local#
```

Examples

In this example scenario, port 1 is on the hook, the application dial is set to call 4085552221, and the destination number is busy.

```
Router# debug pots csm
Router# test pots 1 dial 4085552221#
Router#
```

The following output shows that port 1 is used by the dial application:

```
01:59:42: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
```

The following output shows the events indicating that the CSM is receiving the application digits of the number to call:

```
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
```

The following output shows an event indicating that the telephone connected to port 1 is off the hook:

```
01:59:52: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
```

The following output shows a call-proceeding event pair indicating that the telephone at the far end is busy:

```
01:59:52: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0x0, event=0x3,
 cause=0x11
01:59:52: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id = 0x8005, port = 1
```

The following output shows a call-progressing event pair indicating that the calling telephone is receiving an in-band busy tone from the ISDN switch:

```
01:59:58: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x0
```

```
01:59:58: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8005, port
 = 1
```

The following output shows an event indicating that the calling telephone has hung up:

```
02:00:05: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_VDEV_ONHOOK, call id = 0x8005, port = 1
```

The following output shows an event pair indicating that the call has been terminated:

```
02:00:05: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0xFFFFFFFF,
event=0x0, cause=0x0
02:00:05: CSM_PROC_NEAR_END_DISCONNECT: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8005, port
 = 1
```

Examples

In this example call scenario, port 1 is on the hook, the application dial is set to call 4086661112, the far end successfully connects, and the command **testpotsdisconnect** terminates the call:

```
Router# debug pots csm
Router# test pots 1 dial 4086661112
Router#
```

The following output follows the same sequence of events as shown in Call Scenario 1:

```
1d03h: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
   bchan=0x0, event=0x3, cause=0x0
1d03h: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id = 0x8039, port = 1
1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
   bchan=0xFFFFFFFF, event=0x5, cause=0x0
1d03h: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8039,     port
 = 1
Router# test pots 1 disconnect
```

The **testpotsdisconnect** command disconnects the call before you physically need to put the telephone back on the hook:

```
1d03h: CSM_PROC_CONNECTING: CSM_EVENT_VDEV_APPLICATION_HANGUP_CALL, call id = 0x8039,
 port = 1
1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
   bchan=0xFFFFFFFF, event=0x0, cause=0x0
1d03h: CSM_PROC_DISCONNECTING: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8039,
    port = 1
1d03h: CSM_PROC_DISCONNECTING: CSM_EVENT_TIMEOUT, call id = 0x8039, port = 1
```

debug ppp

To display information on traffic and exchanges in an internetwork implementing the Point-to-Point Protocol (PPP), use the **debug ppp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ppp {**packet**| **negotiation**| **error**| **authentication**| **compression**| **cbcp**}

no debug ppp {**packet**| **negotiation**| **error**| **authentication**| **compression**| **cbcp**}

Syntax Description

packet	Displays PPP packets being sent and received. (This command displays low-level packet dumps.)
negotiation	Displays PPP packets sent during PPP startup, where PPP options are negotiated.
error	Displays protocol errors and error statistics associated with PPP connection negotiation and operation.
authentication	Displays authentication protocol messages, including Challenge Authentication Protocol (CHAP) packet exchanges and Password Authentication Protocol (PAP) exchanges.
compression	Displays information specific to the exchange of PPP connections using Microsoft Point-to-Point Compression (MPPC). This command is useful for obtaining incorrect packet sequence number information where MPPC compression is enabled.
cbcp	Displays protocol errors and statistics associated with PPP connection negotiations using Microsoft Callback (MSCB).

Command Modes

Privileged EXEC

Usage Guidelines

Use the **debug ppp** command when trying to find the following:

- The Network Control Protocols (NCPs) that are supported on either end of a PPP connection
- Any loops that might exist in a PPP internetwork
- Nodes that are (or are not) properly negotiating PPP connections
- Errors that have occurred over the PPP connection
- Causes for CHAP session failures

- Causes for PAP session failures
- Information specific to the exchange of PPP connections using the Callback Control Protocol (CBCP), used by Microsoft clients
- Incorrect packet sequence number information where MPPC compression is enabled

Refer to Internet RFCs 1331, 1332, and 1333 for details concerning PPP-related nomenclature and protocol information.

Caution The **debug ppp compression** command is CPU-intensive and should be used with caution. This command should be disabled immediately after debugging.

Examples

The following is sample output from the **debug ppp packet** command as seen from the Link Quality Monitor (LQM) side of the connection. This example depicts packet exchanges under normal PPP operation.

```
Router# debug ppp packet
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 3 len = 12
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 4 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 4 len = 12
PPP Serial4: O LCP ECHOREP(A) id 4 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 5 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 5 len = 12
PPP Serial4: O LCP ECHOREP(A) id 5 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 6 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 6 len = 12
PPP Serial4: O LCP ECHOREP(A) id 6 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 7 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 7 len = 12
PPP Serial4: O LCP ECHOREP(A) id 7 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The following table describes the significant fields shown in the display.

Table 37: debug ppp packet Field Descriptions

Field	Description
PPP	PPP debugging output.

Field	Description
Serial4	Interface number associated with this debugging information.
(o), O	Packet was detected as an output packet.
(i), I	Packet was detected as an input packet.
lcp_slqr()	Procedure name; running LQM, send a Link Quality Report (LQR).
lcp_rlqr()	Procedure name; running LQM, received an LQR.
input (C021)	Router received a packet of the specified packet type (in hexadecimal notation). A value of C025 indicates packet of type LQM.
state = OPEN	PPP state; normal state is OPEN.
magic = D21B4	Magic Number for indicated node; when output is indicated, this is the Magic Number of the node on which debugging is enabled. The actual Magic Number depends on whether the packet detected is indicated as I or O.
datagramsize 52	Packet length including header.
code = ECHOREQ(9)	Identifies the type of packet received. Both forms of the packet, string and hexadecimal, are presented.
len = 48	Packet length without header.
id = 3	ID number per Link Control Protocol (LCP) packet format.
pkt type 0xC025	Packet type in hexadecimal notation; typical packet types are C025 for LQM and C021 for LCP.
LCP ECHOREQ(9)	Echo Request; value in parentheses is the hexadecimal representation of the LCP type.
LCP ECHOREP(A)	Echo Reply; value in parentheses is the hexadecimal representation of the LCP type.

To elaborate on the displayed output, consider the partial exchange. This sequence shows that one side is using ECHO for its keepalives and the other side is using LQRs.

```
Router# debug ppp packet
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 3 len = 12
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The first line states that the router with debugging enabled has sent an LQR to the other side of the PPP connection:

```
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The next two lines indicate that the router has received a packet of type C025 (LQM) and provides details about the packet:

```
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
```

The next two lines indicate that the router received an ECHOREQ of type C021 (LCP). The other side is sending ECHOs. The router on which debugging is configured for LQM but also responds to ECHOs.

```
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
```

Next, the router is detected to have responded to the ECHOREQ with an ECHOREP and is preparing to send out an LQR:

```
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The following is sample output from the **debug ppp negotiation** command. This is a normal negotiation, where both sides agree on Network Control Program (NCP) parameters. In this case, protocol type IP is proposed and acknowledged.

```
Router# debug ppp negotiation
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = 3D567F8 acked (ok)
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
ppp: ipcp_reqci: returning CONFACK.
 (ok)
PPP Serial4: state = ACKSENT fsm_rconfack(8021): rcvd id 4
```

The following table describes significant fields shown in the display.

Table 38: debug ppp negotiation Field Descriptions

Field	Description
ppp	PPP debugging output.
sending CONFREQ	Router sent a configuration request.
type = 4 (CI_QUALITYTYPE)	Type of LCP configuration option that is being negotiated and a descriptor. A type value of 4 indicates Quality Protocol negotiation; a type value of 5 indicates Magic Number negotiation.

Field	Description
value = C025/3E8	For Quality Protocol negotiation, indicates NCP type and reporting period. In the example, C025 indicates LQM; 3E8 is a hexadecimal value translating to about 10 seconds (in hundredths of a second).
value = 3D56CAC	For Magic Number negotiation, indicates the Magic Number being negotiated.
received config	Receiving node has received the proposed option negotiation for the indicated option type.
acked	Acknowledgment and acceptance of options.
state = ACKSENT	Specific PPP state in the negotiation process.
ipcp_reqci	IPCP notification message; sending CONFACK.
fsm_rconfack (8021)	Procedure fsm_rconfack processes received CONFACKs, and the protocol (8021) is IP.

The first two lines indicate that the router is trying to bring up LCP and will use the indicated negotiation options (Quality Protocol and Magic Number). The value fields are the values of the options themselves. C025/3E8 translates to Quality Protocol LQM. 3E8 is the reporting period (in hundredths of a second). 3D56CAC is the value of the Magic Number for the router.

```
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
```

The next two lines indicate that the other side negotiated for options 4 and 5 as requested and acknowledged both. If the responding end does not support the options, a CONFREJ is sent by the responding node. If the responding end does not accept the value of the option, a Configure-Negative-Acknowledge (CONFNAK) is sent with the value field modified.

```
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = 3D567F8 acked (ok)
```

The next three lines indicate that the router received a CONFAK from the responding side and displays accepted option values. Use the rcvd id field to verify that the CONFREQ and CONFACK have the same ID field.

```
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
```

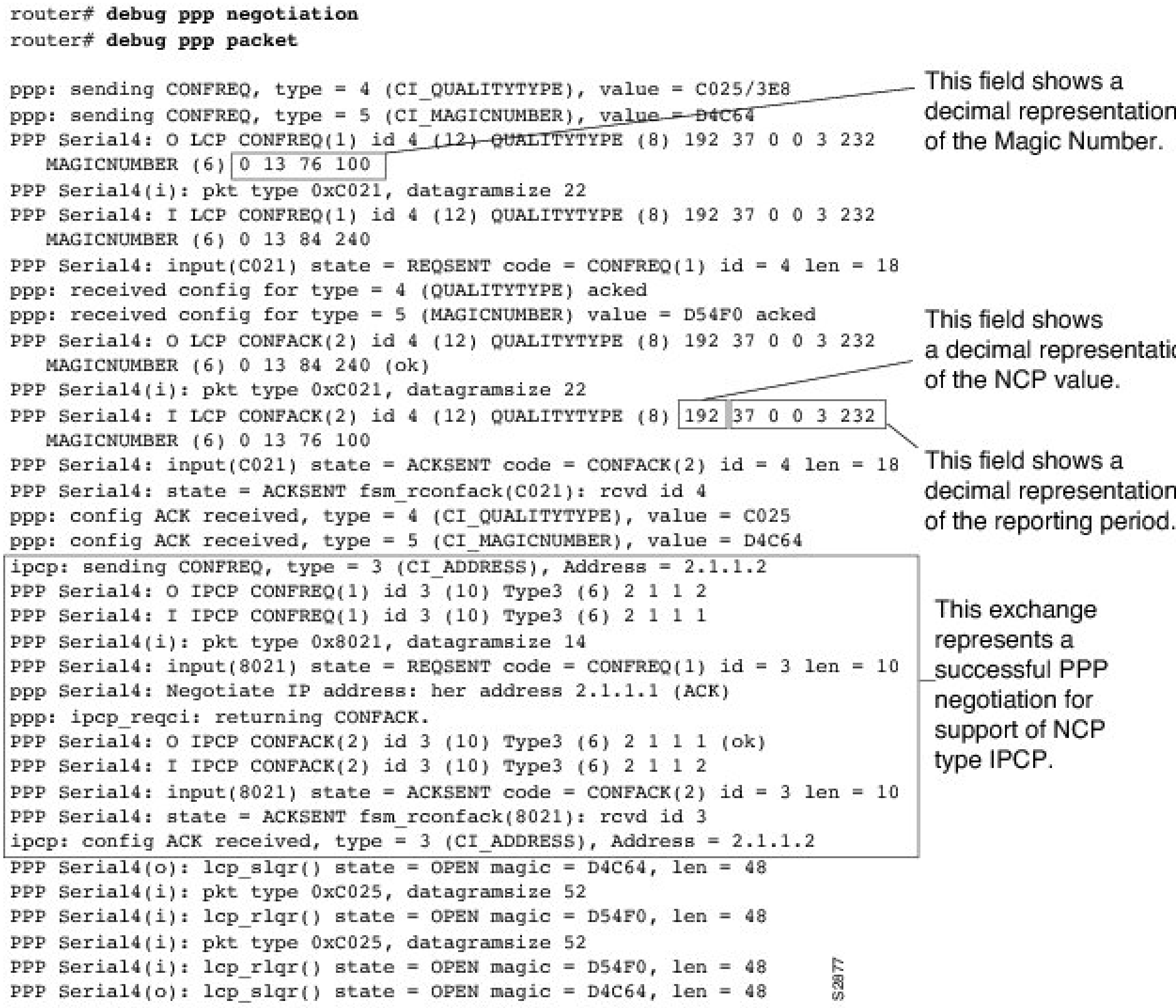
The next line indicates that the router has IP routing enabled on this interface and that the IPCP NCP negotiated successfully:

```
ppp: ipcp_reqci: returning CONFACK.
```

In the last line, the state of the router is listed as ACKSENT.

```
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5\
```

The following is sample output from when the **debug ppp packet** and **debug ppp negotiation** commands are enabled at the same time.



```
router# debug ppp negotiation
router# debug ppp packet

ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = D4C64
PPP Serial4: O LCP CONFREQ(1) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 76 100
PPP Serial4(i): pkt type 0xC021, datagramsize 22
PPP Serial4: I LCP CONFREQ(1) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 84 240
PPP Serial4: input(C021) state = REQSENT code = CONFREQ(1) id = 4 len = 18
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = D54F0 acked
PPP Serial4: O LCP CONFACK(2) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 84 240 (ok)
PPP Serial4(i): pkt type 0xC021, datagramsize 22
PPP Serial4: I LCP CONFACK(2) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 76 100
PPP Serial4: input(C021) state = ACKSENT code = CONFACK(2) id = 4 len = 18
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 4
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = D4C64
ipcp: sending CONFREQ, type = 3 (CI_ADDRESS), Address = 2.1.1.2
PPP Serial4: O IPCP CONFREQ(1) id 3 (10) Type3 (6) 2 1 1 2
PPP Serial4: I IPCP CONFREQ(1) id 3 (10) Type3 (6) 2 1 1 1
PPP Serial4(i): pkt type 0x8021, datagramsize 14
PPP Serial4: input(8021) state = REQSENT code = CONFREQ(1) id = 3 len = 10
ppp Serial4: Negotiate IP address: her address 2.1.1.1 (ACK)
ppp: ipcp_reqci: returning CONFACK.
PPP Serial4: O IPCP CONFACK(2) id 3 (10) Type3 (6) 2 1 1 1 (ok)
PPP Serial4: I IPCP CONFACK(2) id 3 (10) Type3 (6) 2 1 1 2
PPP Serial4: input(8021) state = ACKSENT code = CONFACK(2) id = 3 len = 10
PPP Serial4: state = ACKSENT fsm_rconfack(8021): rcvd id 3
ipcp: config ACK received, type = 3 (CI_ADDRESS), Address = 2.1.1.2
PPP Serial4(o): lcp_slqr() state = OPEN magic = D4C64, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D54F0, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D54F0, len = 48
PPP Serial4(o): lcp_slqr() state = OPEN magic = D4C64, len = 48
```

This field shows a decimal representation of the Magic Number.

This field shows a decimal representatio of the NCP value.

This field shows a decimal representation of the reporting period.

This exchange represents a successful PPP negotiation for support of NCP type IPCP.

The following is sample output from the **debug ppp negotiation** command when the remote side of the connection is unable to respond to LQM requests:

```
Router# debug ppp negotiation
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
```

```
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44C1488
```

The following is sample output when no response is detected for configuration requests (with both the **debug ppp negotiation** and **debug ppp packet** commands enabled):

```
Router# debug ppp negotiation
Router# debug ppp packet
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 14 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E0980 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 15 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E1828 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 16 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E27C8 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 17 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E3768 State= 3
```

The following is sample output from the **debug ppp error** command. These messages might appear when the Quality Protocol option is enabled on an interface that is already running PPP.

```
Router# debug ppp error
PPP Serial3(i): rlqr receive failure. successes = 15
PPP: myrcvdiffp = 159 peerxmitdiffp = 41091
PPP: myrcvdiffo = 2183 peerxmitdiffo = 1714439
PPP: threshold = 25
PPP Serial4(i): rlqr transmit failure. successes = 15
PPP: myxmitdiffp = 41091 peerrcvdiffp = 159
PPP: myxmitdiffo = 1714439 peerrcvdiffo = 2183
PPP: l->OutLQRs = 1 LastOutLQRs = 1
PPP: threshold = 25
PPP Serial3(i): lqr_protrej() Stop sending LQRs.
PPP Serial3(i): The link appears to be looped back.
```

The following table describes the significant fields shown in the display.

Table 39: debug ppp error Field Descriptions

Field	Description
PPP	PPP debugging output.
Serial3(i)	Interface number associated with this debugging information; indicates that this is an input packet.
rlqr receive failure	Request to negotiate the Quality Protocol option is not accepted.
myrcvdiffp = 159	Number of packets received over the time period.

Field	Description
peerxmitdiffp = 41091	Number of packets sent by the remote node over this period.
myrcvdiffo = 2183	Number of octets received over this period.
peerxmitdiffo = 1714439	Number of octets sent by the remote node over this period.
threshold = 25	Maximum error percentage acceptable on this interface. This percentage is calculated by the threshold value entered in the **ppp quality** *number* interface configuration command. A value of 100 - *number* (100 minus *number*) is the maximum error percentage. In this case, a *number* of 75 was entered. This means that the local router must maintain a minimum 75 percent non-error percentage, or the PPP link will be considered down.
OutLQRs = 1	Local router’s current send LQR sequence number.
LastOutLQRs = 1	The last sequence number that the remote node side has seen from the local node.

The following is sample output from the **debug ppp authentication** command. Use this command to determine why an authentication fails.

```
Router# debug ppp authentication
Serial0: Unable to authenticate. No name received from peer
Serial0: Unable to validate CHAP response. USERNAME pioneer not found.
Serial0: Unable to validate CHAP response. No password defined for USERNAME pioneer
Serial0: Failed CHAP authentication with remote.
Remote message is Unknown name
Serial0: remote passed CHAP authentication.
Serial0: Passed CHAP authentication with remote.
Serial0: CHAP input code = 4 id = 3 len = 48
```

In general, these messages are self-explanatory. Fields that can show optional output are outlined in the following table.

Table 40: debug ppp authentication Field Descriptions

Field	Description
Serial0	Interface number associated with this debugging information and CHAP access session in question.
USERNAME pioneer not found.	The name *pioneer* in this example is the name received in the CHAP response. The router looks up this name in the list of usernames that are configured for the router.

Field	Description
Remote message is Unknown name	The following messages can appear: • No name received to authenticate • Unknown name • No secret for given name • Short MD5 response received • MD compare failed
code = 4	Specific CHAP type packet detected. Possible values are as follows: • 1--Challenge • 2--Response • 3--Success • 4--Failure
id = 3	ID number per LCP packet format.
len = 48	Packet length without header.

The following shows sample output from the **debug ppp** command usingthe **cbcp** keyword**.** This output depicts packet exchanges under normal PPP operation where the Cisco access server is waiting for the remote PC to respond to the MSCB request. The router also has **debug ppp negotiation** and **service timestamps msec** commands enabled.

```
Router# debug ppp cbcp
Dec 17 00:48:11.302: As8 MCB: User mscb Callback Number - Client ANY
Dec 17 00:48:11.306: Async8 PPP: O MCB Request(1) id 1 len 9
Dec 17 00:48:11.310: Async8 MCB: O  1  1  0  9  2  5  0  1  0
Dec 17 00:48:11.314: As8 MCB: O Request Id 1 Callback Type Client-Num delay 0
Dec 17 00:48:13.342: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:13.346: Async8 PPP: O MCB Request(1) id 2 len 9
Dec 17 00:48:13.346: Async8 MCB: O  1  2  0  9  2  5  0  1  0
Dec 17 00:48:13.350: As8 MCB: O Request Id 2 Callback Type Client-Num delay 0
Dec 17 00:48:15.370: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:15.374: Async8 PPP: O MCB Request(1) id 3 len 9
Dec 17 00:48:15.374: Async8 MCB: O  1  3  0  9  2  5  0  1  0
Dec 17 00:48:15.378: As8 MCB: O Request Id 3 Callback Type Client-Num delay 0
Dec 17 00:48:17.398: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:17.402: Async8 PPP: O MCB Request(1) id 4 len 9
Dec 17 00:48:17.406: Async8 MCB: O  1  4  0  9  2  5  0  1  0
Dec 17 00:48:17.406: As8 MCB: O Request Id 4 Callback Type Client-Num delay 0
Dec 17 00:48:19.426: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:19.430: Async8 PPP: O MCB Request(1) id 5 len 9
Dec 17 00:48:19.430: Async8 MCB: O  1  5  0  9  2  5  0  1  0
Dec 17 00:48:19.434: As8 MCB: O Request Id 5 Callback Type Client-Num delay 0
Dec 17 00:48:21.454: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:21.458: Async8 PPP: O MCB Request(1) id 6 len 9
Dec 17 00:48:21.462: Async8 MCB: O  1  6  0  9  2  5  0  1  0
Dec 17 00:48:21.462: As8 MCB: O Request Id 6 Callback Type Client-Num delay 0
Dec 17 00:48:23.482: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:23.486: Async8 PPP: O MCB Request(1) id 7 len 9
Dec 17 00:48:23.490: Async8 MCB: O  1  7  0  9  2  5  0  1  0
Dec 17 00:48:23.490: As8 MCB: O Request Id 7 Callback Type Client-Num delay 0
Dec 17 00:48:25.510: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:25.514: Async8 PPP: O MCB Request(1) id 8 len 9
Dec 17 00:48:25.514: Async8 MCB: O  1  8  0  9  2  5  0  1  0
Dec 17 00:48:25.518: As8 MCB: O Request Id 8 Callback Type Client-Num delay 0
Dec 17 00:48:26.242: As8 PPP: I pkt type 0xC029, datagramsize 18
Dec 17 00:48:26.246: Async8 PPP: I MCB Response(2) id 8 len 16
Dec 17 00:48:26.250: Async8 MCB: I  2  8  0  10  2  C  C  1  32  34  39  32  36  31  33  0

Dec 17 00:48:26.254: As8 MCB: Received response
Dec 17 00:48:26.258: As8 MCB: Response CBK-Client-Num 2 12 12,  addr 1-2492613
Dec 17 00:48:26.262: Async8 PPP: O MCB Ack(3) id 9 len 16
Dec 17 00:48:26.266: Async8 MCB: O  3  9  0  10  2  C  C  1  32  34  39  32  36  31  33  0

Dec 17 00:48:26.270: As8 MCB: O Ack Id 9 Callback Type Client-Num delay 12
Dec 17 00:48:26.270: As8 MCB: Negotiated MCB with peer
Dec 17 00:48:26.390: As8 LCP: I TERMREQ [Open] id 4 len 8 (0x00000000)
Dec 17 00:48:26.390: As8 LCP: O TERMACK [Open] id 4 len 4
Dec 17 00:48:26.394: As8 MCB: Peer terminating the link
Dec 17 00:48:26.402: As8 MCB: Initiate Callback for mscb at 2492613 using Async
```

The following is sample output from the **debug ppp compression** command with **service timestamps** enabled and shows a typical PPP packet exchange between the router and Microsoft client where the MPPC header sequence numbers increment correctly:

```
Router# debug ppp compression
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2003/0x0003
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2004/0x0004
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2005/0x0005
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2006/0x0006
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2007/0x0007
```

The following table describes the significant fields shown in the display.

Table 41: debug ppp compression Field Descriptions

Field	Description
interface	Interface enabled with MPPC.
Decomp - hdr/	Decompression header and bit settings.
exp_cc#	Expected coherency count.
0x2003	Received sequence number.
0x0003	Expected sequence number.

The following shows sample output from **debug ppp negotiation** and **debug ppp error** commands, which can be used to troubleshoot initial PPP negotiation and setup errors. This example shows a virtual interface (virtual interface 1) during normal PPP operation and CCP negotiation.

```
Router# debug ppp negotiation error
Vt1 PPP: Unsupported or un-negotiated protocol. Link arp
VPDN: Chap authentication succeeded for p5200
Vi1 PPP: Phase is DOWN, Setup
Vi1 VPDN: Virtual interface created for dinesh@cisco.com
Vi1 VPDN: Set to Async interface
Vi1 PPP: Phase is DOWN, Setup
Vi1 VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
Vi1 CCP: Re-Syncing history using legacy method
```

```
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
Vi1 PPP: Treating connection as a dedicated line
Vi1 PPP: Phase is ESTABLISHING, Active Open
Vi1 LCP: O CONFREQ [Closed] id 1 len 25
Vi1 LCP:    ACCM 0x000A0000 (0x0206000A0000)
Vi1 LCP:    AuthProto CHAP (0x0305C22305)
Vi1 LCP:    MagicNumber 0x000FB69F (0x0506000FB69F)
Vi1 LCP:    PFC (0x0702)
Vi1 LCP:    ACFC (0x0802)
Vi1 VPDN: Bind interface direction=2
Vi1 PPP: Treating connection as a dedicated line
Vi1 LCP: I FORCED CONFREQ len 21
Vi1 LCP:    ACCM 0x000A0000 (0x0206000A0000)
Vi1 LCP:    AuthProto CHAP (0x0305C22305)
Vi1 LCP:    MagicNumber 0x12A5E4B5 (0x050612A5E4B5)
Vi1 LCP:    PFC (0x0702)
Vi1 LCP:    ACFC (0x0802)
Vi1 VPDN: PPP LCP accepted sent & rcv CONFACK
Vi1 PPP: Phase is AUTHENTICATING, by this end
Vi1 CHAP: O CHALLENGE id 1 len 27 from "l_4000"
Vi1 CHAP: I RESPONSE id 20 len 37 from "dinesh@cisco.com"
Vi1 CHAP: O SUCCESS id 20 len 4
Vi1 PPP: Phase is UP
Vi1 IPCP: O CONFREQ [Closed] id 1 len 10
Vi1 IPCP:    Address 15.2.2.3 (0x03060F020203)
Vi1 CCP: O CONFREQ [Not negotiated] id 1 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 IPCP: I CONFREQ [REQsent] id 1 len 34
Vi1 IPCP:    Address 0.0.0.0 (0x030600000000)
Vi1 IPCP:    PrimaryDNS 0.0.0.0 (0x810600000000)
Vi1 IPCP:    PrimaryWINS 0.0.0.0 (0x820600000000)
Vi1 IPCP:    SecondaryDNS 0.0.0.0 (0x830600000000)
Vi1 IPCP:    SecondaryWINS 0.0.0.0 (0x840600000000)
Vi1 IPCP: Using the default pool
Vi1 IPCP: Pool returned 11.2.2.5
Vi1 IPCP: O CONFREJ [REQsent] id 1 len 16
Vi1 IPCP:    PrimaryWINS 0.0.0.0 (0x820600000000)
Vi1 IPCP:    SecondaryWINS 0.0.0.0 (0x840600000000)
Vi1 CCP: I CONFREQ [REQsent] id 1 len 15
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP:    Stacker history 1 check mode EXTENDED (0x1105000104)
Vi1 CCP: Already accepted another CCP option, rejecting this STACKER
Vi1 CCP: O CONFREJ [REQsent] id 1 len 9
Vi1 CCP:    Stacker history 1 check mode EXTENDED (0x1105000104)
Vi1 IPCP: I CONFACK [REQsent] id 1 len 10
Vi1 IPCP:    Address 15.2.2.3 (0x03060F020203)
Vi1 CCP: I CONFACK [REQsent] id 1 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: I CONFREQ [ACKrcvd] id 2 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: O CONFACK [ACKrcvd] id 2 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: State is Open
Vi1 IPCP: I CONFREQ [ACKrcvd] id 2 len 22
Vi1 IPCP:    Address 0.0.0.0 (0x030600000000)
Vi1 IPCP:    PrimaryDNS 0.0.0.0 (0x810600000000)
Vi1 IPCP:    SecondaryDNS 0.0.0.0 (0x830600000000)
Vi1 IPCP: O CONFNAK [ACKrcvd] id 2 len 22
Vi1 IPCP:    Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:    PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:    SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: I CONFREQ [ACKrcvd] id 3 len 22
Vi1 IPCP:    Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:    PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:    SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: O CONFACK [ACKrcvd] id 3 len 22
Vi1 IPCP:    Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:    PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:    SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: State is Open
Vi1 IPCP: Install route to 11.2.2.5
```

debug ppp bap

To display general Bandwidth Allocation Control Protocol (BACP) transactions, use the **debugpppbap** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ppp bap [**error**| **event**| **negotiation**]

no debug ppp bap [**error**| **event**| **negotiation**]

Syntax Description

error	(Optional) Displays local errors.
event	(Optional) Displays information about protocol actions and transitions between action states (pending, waiting, idle) on the link.
negotiation	(Optional) Displays successive steps in negotiations between peers.

Command Modes

Privileged EXEC

Usage Guidelines

Do not use this command when memory is scarce or in very high traffic situations.

Examples

The following types of events generate the debugging messages displayed in the figures in this section:

- A dial attempt failed.
- A BACP group was created.
- A BACP group was removed.
- The precedence of the group changed.
- Attempting to dial a number.
- Received a BACP message.
- Discarding a BACP message.
- Received an unknown code.
- Cannot find the appropriate BACP group on input.
- Displaying the response type.
- Incomplete mandatory options notification.
- Invalid outgoing message type.
- Unable to build an output message.

- Sending a BACP message.
- Details about the sent message (type of message, its identifier, the virtual access interface that sent it).

The following is sample output from the **debugpppbap** command:

```
Router# debug ppp bap
BAP Virtual-Access1: group "laudrup" (2) (multilink) without precedence created
BAP laudrup: sending CallReq, id 2, len 38 on BRI3:1 to remote
BAP Virtual-Access1: received CallRsp, id 2, len 13
BAP laudrup: CallRsp, id 2, ACK
BAP laudrup: attempt1 to dial 19995776677 on BRI3
 ---> reason BAP - Multilink bundle overloaded
BAP laudrup: sending StatusInd, id 2, len 44 on Virtual-Access1 to remote
BAP Virtual-Access1: received StatusRsp, id 2, len 1
BAP laudrup: StatusRsp, id 2, ACK
```

The following table describes the significant fields shown in the display.

Table 42: debug ppp bap Field Descriptions

Field	Description
BAP Virtual-Access1:	Identifier of the virtual access interface in use.
group “laudrup”	Name of the BACP group.
sending CallReq	Action initiated; in this case, sending a call request.
on BRI3:1 to remote	Physical interface being used.
BAP laudrup: attempt1 to dial 19995776677 on BRI3 ---> reason BAP - Multilink bundle overloaded	Call initiated, number being dialed, and physical interface being used. Reason for initiating the BACP call.
BAP laudrup: sending StatusInd, id 2, len 44 on Virtual-Access1 to remote	Details about the sent message: It was a status indication message, had identifier 2, had a BACP datagram length 44, and was sent on virtual access interface 1. You can display information about the virtual access interface by using the **showinterfacesvirtual-access** EXEC command. (The length shown at the end of each negotiated option includes the 2-byte type and length header.)

The **debugpppbapevent** command might show state transitions and protocol actions, in addition to the basic **debugpppbap** command.

The following is sample output from the **debugpppbapevent** command:

```
Router# debug ppp bap event
BAP laudrup: Idle --> AddWait
BAP laudrup: AddWait --> AddPending
BAP laudrup: AddPending --> Idle
```

The following is sample output from the **debugpppbapevent** command:

```
Router# debug ppp bap event
Peer does not support a message type
No response to a particular request
No response to all request retransmissions
Not configured to initiate link addition
Expected action by peer has not occurred
Exceeded number of retries
No links available to call out
Unable to provide phone numbers for callback
Maximum number of links in the group
Minimum number of links in the group
Unable to process link addition at present
Unable to process link removal at present
Not configured/unable to initiate link removal
Link addition completed notification
Link addition failed notification
Determination of location of the group config
Link with specified discriminator not in group
Link removal failed
Call failure with status
Failed to dial specified number
Discarding retransmission
Unable to find received identifier
Received StatusInd when no call pending
Discarding message with no phone delta
Unable to send message in particular state
Received a zero identifier
Request has precedence
```

The error messages displayed might be added to the basic output when the **debugpppbaperror** command is used. Because the errors are very rare, you might never see these messages.

```
Router# debug ppp bap error
Unable to find appropriate request for received response
Invalid message type of queue
Received request is not part of the group
Add link attempt failed to locate group
Remove link attempt failed to locate group
Unable to inform peer of link addition
Changing of precedence cannot locate group
Received short header/illegal length/short packet
Invalid configuration information length
Unable to NAK incomplete options
Unable to determine current number of links
No interface list to dial on
Attempt to send invalid data
Local link discriminator is not in group
Received response type is incorrect for identifier
```

The messages displayed might be added to the basic output when the **debugpppbapnegotiation** command is used:

```
Router# debug ppp bap negotiation
BAP laudrup: adding link speed 64 kbps for type 0x1 len 5
BAP laudrup: adding reason "User initiated addition", len 25
BAP laudrup: CallRsp, id 4, ACK
BAP laudrup: link speed 64 kbps for types 0x1, len 5 (ACK)
BAP laudrup: phone number "1: 0 2: ", len 7 (ACK)
BAP laudrup: adding call status 0, action 0 len 4
BAP laudrup: adding 1 phone numbers "1: 0 2: " len 7
BAP laudrup: adding reason "Successfully added link", len 25
BAP laudrup: StatusRsp, id 4, ACK
```

Additional negotiation messages might also be displayed for the following:

```
Received BAP message
Sending message
Decode individual options for send/receive
Notification of invalid options
```

The following shows additional reasons for a particular BAP action that might be displayed in an “adding reason” line of the **debugpppbapnegotiation** command output:

```
"Outgoing add request has precedence"
"Outgoing remove request has precedence"
"Unable to change request precedence"
"Unable to determine valid phone delta"
"Attempting to add link"
"Link addition is pending"
"Attempting to remove link"
"Link removal is pending"
"Precedence of peer marked CallReq for no action"
"Callback request rejected due to configuration"
"Call request rejected due to configuration"
"No links of specified type(s) available"
"Drop request disallowed due to configuration"
"Discriminator is invalid"
"No response to call requests"
"Successfully added link"
"Attempt to dial destination failed"
"No interfaces present to dial out"
"No dial string present to dial out"
"Mandatory options incomplete"
"Load has not exceeded threshold"
"Load is above threshold"
"Currently attempting to dial destination"
"No response to CallReq from race condition"
```

The following table describes the reasons for a BACP Negotiation Action.

Table 43: Explanation of Reasons for BACP Negotiation Action

Reason	Explanation
“Outgoing add request has precedence”	Received a CallRequest or CallbackRequest while we were waiting on a CallResponse or CallbackResponse to a sent request. We are the favored peer from the initial BACP negotiation, so we are issuing a NAK to our peer request.
“Outgoing remove request has precedence”	Received a LinkDropQueryRequest while waiting on a LinkDropQueryResponse to a sent request. We are the favored peer from the initial BACP negotiation, therefore we are issuing a NAK to our peer request.
“Unable to change request precedence”	Received a CallRequest, CallbackRequest, or LinkDropQueryRequest while waiting on a LinkDropQueryResponse to a sent request. Our peer is deemed to be the favored peer from the initial BACP negotiation and we were unable to change the status of our outgoing request in response to the favored request, so we are issuing a NAK. (This is an internal error and should never be seen.)
“Unable to determine valid phone delta”	Received a CallRequest from our peer but are unable to provide the required phone delta for the response, so we are issuing a NAK. (This is an internal error and should never be seen.)

Reason	Explanation
“Attempting to add link”	Received a LinkDropQueryRequest while attempting to add a link; a NAK is issued.
“Link addition is pending”	Received a LinkDropQueryRequest, CallRequest, or CallbackRequest while attempting to add a link as the result of a previous operation; a NAK is issued in the response.
“Attempting to remove link”	Received a CallRequest or CallbackRequest while attempting to remove a link; a NAK is issued.
“Link removal is pending”	Received a CallRequest, CallbackRequest, or LinkDropQueryRequest while attempting to remove a link as the result of a previous operation; a NAK is issued in the response.
“Precedence of peer marked CallReq for no action”	Received an ACK to a previously unfavored CallRequest; we are issuing a CallStatusIndication to inform our peer that there will be no further action on our part as per this response.
“Callback request rejected due to configuration”	Received a CallbackRequest but we are configured not to accept them; a REJect is issued to our peer.
“Call request rejected due to configuration”	Received a CallRequest but we are configured not to accept them; a REJect is issued to our peer.
“No links of specified type(s) available”	We received a CallRequest but no links of the specified type and speed are available; a NAK is issued.
“Drop request disallowed due to configuration”	Received a LinkDropQueryRequest but we are configured not to accept them; a NAK is issued to our peer.
“Discriminator is invalid”	Received a LinkDropQueryRequest but the local link discriminator is not contained within the bundle; a NAK is issued.
“No response to call requests”	After no response to our CallRequest message, a CallStatusIndication is sent to the peer informing that no more action will be taken on behalf of this operation.
“Successfully added link”	Sent as part of the CallStatusIndication informing our peer that we successfully completed the addition of a link to the bundle as the result of the transmission of a CallRequest or the reception of a CallbackRequest.

Reason	Explanation
"Attempt to dial destination failed"	Sent as part of the CallStatusIndication informing our peer that we failed in an attempt to add a link to the bundle as the result of the transmission of a CallRequest or the reception of a CallbackRequest. The retry field with the CallStatusIndication informs the peer of our intentions.
"No interfaces present to dial out"	There are no available interfaces to dial out on to attempt to add a link to the bundle, and we will not retry the dial attempt.
"No dial string present to dial out"	We do not have a dial string to dial out with to attempt to add a link to the bundle, and we are not going to retry the dial attempt. (This is an internal error and should never be seen.)
"Mandatory options incomplete"	Received a CallRequest, CallbackRequest, LinkDropQueryRequest, or CallStatusIndication and the mandatory options are not present, so a NAK is issued in the response. (A CallStatusResponse is an ACK, however).
"Load has not exceeded threshold"	Received a CallRequest or CallbackRequest but we are issuing a NAK in the response. We are monitoring the load of the bundle, and so we determine when links should be added to the bundle.
"Load is above threshold"	Received a LinkDropQueryRequest but we are issuing a NAK in the response. We are monitoring the load of the bundle, and so we determine when links should be removed from the bundle.
"Currently attempting to dial destination"	Received a CallbackRequest which is a retransmission of one that we previously ACK'd and are dialing the number suggested in the request. We are issuing an ACK because we did so previously, even though our peer never saw the previous response.
"No response to CallReq from race condition"	We issued a CallRequest but failed to receive a response, and we are issuing a CallStatusIndication to inform our peer of our intention not to proceed with the operation.

debug ppp ip address-save

To display debug information about the IPv4 Address Conservation in Dual Stack Environments feature such as authorization, authentication, and IPv4 address allocation messages on the broadband remote access server (BRAS), use the **debug ppp ip address-save** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ppp ip address-save

no debug ppp ip address-save

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.5S	This command was introduced.

Usage Guidelines

Use the **debug ppp ip address-save** command to display authorization, authentication, and IPv4 address allocation messages on the BRAS. This command shows that the IPv4 Address Conservation in Dual Stack Environments feature has been enabled and displays the events that are triggered by enabling the feature. See the "Related Commands" section for **debug** commands that should be used in conjunction with this command

Examples

The following is sample output from the **debug ppp ip address-save** command:

```
Router# debug ppp ip address-save

Vi2.1 IPCP AUTH: Adding password in AAA author request
Vi2.1 IPCP AUTH: Added password and AAA VSA [enable] in author request
Vi2.1 PPP: Added IPv4 address [10.1.1.25] to include in acct record
Vi2.1 PPP: Triggering interim acct request
Vi2.1 PPP: IPCP going down, resetting neg authorized flag
Vi2.1 PPP: Peer IPv4 address in author data = 10.1.1.25
Vi2.1 PPP: Removing IPv4 address from Accounting DB
Vi2.1 PPP: Triggering interim acct request
Vi2.1 PPP: IPCP went down, checking status of other NCPs
```

The output is self-explanatory.

Related Commands

Command	Description
debug ppp authentication	Displays authentication protocol messages, including CHAP packet exchanges and PAP exchanges.

Command	Description
debug ppp authorization	Displays information about authorization attributes received from the RADIUS server.
debug ppp negotiation	Displays PPP packets sent during PPP startup, where PPP options are negotiated.
debug radius	Displays accounting and authentication information and client/server interaction events on the RADIUS server.
ppp ip address-save aaa-acct-vsa	Enables IPv4 address conservation on the BRAS.

debug ppp multilink events

To display information about events affecting multilink groups established for Bandwidth Allocation Control Protocol (BACP), use the**debugpppmultilinkevents** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ppp multilink events

no debug ppp multilink events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Caution Do not use this command when memory is scarce or in very high traffic situations.

Examples

The following is sample output from the **debugpppmultilinkevents** command:

```
Router# debug ppp multilink events
MLP laudrup: established BAP group 4 on Virtual-Access1, physical BRI3:1
MLP laudrup: removed BAP group 4
```

Other event messages include the following:

```
Unable to find bundle for BAP group identifier
Unable to find physical interface to start BAP
Unable to create BAP group
Attempt to start BACP when inactive or running
Attempt to start BACP on non-MLP interface
Link protocol has gone down, removing BAP group
Link protocol has gone down, BAP not running or present
```

The following table describes the significant fields shown in the display.

Table 44: debug ppp multilink events Field Descriptions

Field	Description
MLP laudrup	Name of the multilink group.
established BAP group 4	Internal identifier. The same identifiers are used in the **showpppbapgroup** command output.
Virtual-Access1	Dynamic access interface number.
physical BRI3:1	Bundle was established from a call on this interface.

Field	Description
removed BAP group 4	When the bundle is removed, the associated BACP group (with its ID) is also removed.

debug ppp multilink fragments

To display information about individual multilink fragments and important multilink events, use the **debug ppp multilink fragments** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ppp multilink fragments

no debug ppp multilink fragments

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Caution The **debug ppp multilink fragments** command has some memory overhead and should not be used when memory is scarce or in very high traffic situations.

Examples

The following is sample output from the **debug ppp multilink fragments** command when used with the **ping** EXEC command. The debug output indicates that a multilink PPP packet on interface BRI 0 (on the B channel) is an input (I) or output (O) packet. The output also identifies the sequence number of the packet and the size of the fragment.

```
Router# debug ppp multilink fragments

Router# ping 7.1.1.7
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 7.1.1.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/34/36 ms
Router#
2:00:28: MLP BRI0: B-Channel 1: O seq 80000000: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000001: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000001: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000000: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000002: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000003: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000003: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000002: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000004: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000005: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000005: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000004: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000006: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000007: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000007: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000006: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000008: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000009: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000009: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000008: size 58
```

debug ppp multilink negotiation

Note Effective with release 11.3, the **debugpppmultilinknegotiation** command is not available in Cisco IOS software.

To display information about events affecting multilink groups established controlled by Bandwidth Allocation Control Protocol (BACP), use the**debugpppmultilinknegotiation** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ppp multilink negotiation

no debug ppp multilink negotiation

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3	This command was removed and is not available in Cisco IOS software.

Usage Guidelines

Caution Do not use this command when memory is scarce or in very high traffic situations.

Examples

The following sample output shows Link Control Protocol (LCP) and Network Control Program (NCP) messages that might appear in **debugpppmultilinknegotiation** command. These messages show information about PPP negotiations between the multilink peers.

```
Router# debug ppp multilink negotiation
ppp: sending CONFREQ, type = 23 (CI_LINK_DISCRIMINATOR), value = 0xF
PPP BRI3:1: received config for type = 23 (LINK_DISCRIMINATOR) value = 0xA acked
Router# debug ppp multilink negotiation
ppp: sending CONFREQ, type = 1 (CI_FAVORED_PEER), value = 0x647BD090
PPP Virtual-Access1: received CONFREQ, type 1, value = 0x382BBF5 (ACK)
PPP Virtual-Access1: BACP returning CONFACK
ppp: config ACK received, type = 1 (CI_FAVORED_PEER), value = 0x647BD090
PPP Virtual-Access1: BACP up
```

The following table describes the significant fields shown in the display.

Table 45: debug ppp multilink negotiation Field Descriptions

Field	Description
sending CONFREQ, type = 23 (CI_LINK_DISCRIMINATOR), value = 0xF	Sending a configuration request and the value of the link discriminator. Each peer assigns a discriminator value to identify a specific link. The values are significant to each peer individually but do not have to be shared.
PPP BRI3:1:	Physical interface being used.
CI_FAVORED_PEER	When the PPP NCP negotiation occurs over the first link in a bundle, the BACP peers use a Magic Number akin to that used by LCP to determine which peer should be favored when both implementations send a request at the same time. The peer that negotiated the higher number is deemed to be favored. That peer should issue a negative acknowledgment to its unfavored peer, which in turn should issue a positive acknowledgment, if applicable according to other link considerations.
PPP Virtual-Access1: BACP returning CONFACK	Returning acknowledgment that BACP is configured.
PPP Virtual-Access1: BACP up	Indicating that the BACP NCP is open.

debug ppp redundancy

To debug PPP synchronization on the networking device, use the **debug ppp redundancy**command in privileged EXEC mode. To disable the display of debugging output, use the **no** form of this command.

debug ppp redundancy [**detailed**| **event**]

no debug ppp redundancy [**detailed**| **event**]

Syntax Description

detailed	(Optional) Displays detailed debug messages related to specified PPP redundancy events.
event	(Optional) Displays information about protocol actions and transitions between action states (pending, waiting, idle) on the link.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced on the Cisco 7500, 10000, and 12000 series Internet routers.
12.2(18)S	This command was integrated into Cisco IOS Release 12.2(18)S on Cisco 7500 series routers.
12.2(20)S	Support was added for the Cisco 7304 router. The Cisco 7500 series router is not supported in Cisco IOS Release 12.2(20)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Examples

The following example displays detailed debug messages related to specified PPP redundancy events:

```
Router# debug ppp redundancy detailed
```

debug ppp unique address

To display debugging information about duplicate addresses received from RADIUS, use the **debug ppp unique address**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 policy

no debug ipv6 policy

Syntax Description

This command has no arguments or keywords.

Command Default

Information about duplicate addresses received from RADIUS is not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.2S	This command was introduced.

Usage Guidelines

The **debug p**pp unique address command enables you to view debugging information about duplicate addresses received from RADIUS.

Examples

The following example enables debugging output about duplicate addresses received from RADIUS:

```
Router# debug ppp unique address
```

debug pppatm

To enable debug reports for PPP over ATM (PPPoA) events, errors, and states, either globally or conditionally, on an interface or virtual circuit (VC), use the **debugpppatm** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug pppatm {**event**| **error**| **state**} [**interface atm** *interface-number* [*subinterface-number*]] **vc** {[*vpi*/*vci*] *vci*| *virtual-circuit-name*}

no debug pppatm {**event**| **error**| **state**} [**interface atm** *interface-number* [*subinterface-number*]] **vc** {[*vpi*/*vci*] *vci*| *virtual-circuit-name*}

Syntax Description

event	PPPoA events.
error	PPPoA errors.
state	PPPoA state.
interface atm *interface-number* [*subinterface-number*]	(Optional) Specifies a particular ATM interface by interface number and optionally a subinterface number separated by a period.
vc [*vpi*/]*vcivirtual-circuit-name*	(Optional) Virtual circuit (VC) keyword followed by a virtual path identifier (VPI), virtual channel identifier (VCI), and VC name. A slash mark is required after the VPI.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T	This command was introduced.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.

Usage Guidelines

Each specific PPPoA debug report must be requested on a separate command line; see the “Examples” section.

Examples

The following is example output of a PPPoA session with event, error, and state debug reports enabled on ATM interface 1/0.10:

```
Router# debug pppatm event interface atm1/0.10
Router# debug pppatm error interface atm1/0.10
Router# debug pppatm state interface atm1/0.10
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = Clear Session
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = Disconnecting
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = AAA gets dynamic attrs
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = AAA gets dynamic attrs
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = SSS Cleanup
00:03:08: PPPATM: ATM1/0.10 0/101 [0], State = DOWN
00:03:08: PPPATM: ATM1/0.10 0/101 [0], Event = Up Pending
00:03:16: PPPATM: ATM1/0.10 0/101 [0], Event = Up Dequeued
00:03:16: PPPATM: ATM1/0.10 0/101 [0], Event = Processing Up
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = Access IE allocated
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = Set Pkts to SSS
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets retrived attrs
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets nas port details
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA unique id allocated
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = No AAA method list set
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = SSS Request
00:03:16: PPPATM: ATM1/0.10 0/101 [2], State = NAS_PORT_POLICY_INQUIRY
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = SSS Msg Received = 1
00:03:16: PPPATM: ATM1/0.10 0/101 [2], State = PPP_START
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = PPP Msg Received = 1
00:03:16: PPPATM: ATM1/0.10 0/101 [2], State = LCP_NEGOTIATION
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = PPP Msg Received = 4
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = HW Switch support FORW = 0
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = Access IE get nas port
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = PPP Msg Received = 5
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = Set Pkts to SSS
00:03:27: PPPATM: ATM1/0.10 0/101 [2], State = FORWARDED
```

The following table describes the significant fields shown in the display.

Table 46: debug pppatm Field Descriptions

Field	Description
Event	Reports PPPoA events for use by Cisco engineering technical assistance personnel.
State	Reports PPPoA states for use by Cisco engineering technical assistance personnel.

Related Commands

Command	Description
atm pppatm passive	Places an ATM subinterface into passive mode.
show pppatm summary	Displays PPPoA session counts.

debug pppatm redundancy

To debug PPP over ATM (PPPoA) redundancy events on a dual Route Processor High Availability (HA) system and display cluster control manager (CCM) events and messages, use the **debug pppatm redundancy**command in privileged EXEC mode. To disable the display of debugging output, use the **no** form of this command.

debug pppatm redundancy [**interface atm** *interface-number* [**vc** {*vpi*/*vci*| *vci*}]]

no debug pppatm redundancy [**interface atm** *interface-number* [**vc** {*vpi*/*vci*| *vci*}]]

Syntax Description

interface atm *interface-number*	(Optional) Specifies a particular ATM interface by interface number.
vc	(Optional) Specifies the virtual circuit (VC).
vpi/vci	(Optional) Virtual path identifier (VPI) and virtual channel identifier (VCI) value. The range is from 0 to 255.
vci	(Optional) VCI. The range is from 1 to 65535.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(31)SB2	This command was introduced.
Cisco IOS XE Release 3.3S	This command was modified. The **interface atm** *interface-number*keyword-argument pair, **vc**keyword, and *vpi/vci*and *vci*arguments were added.

Usage Guidelines

The CCM provides the capability to facilitate and synchronize session bring-up on the standby processor of a dual Route Processor HA system. Use the **debug pppatm redundancy** command to display CCM events and messages for PPPoA sessions on HA systems.

To create sessions on the standby processor with the same virtual-access (sub)interface as that on the active processor, base virtual-access interface creation on the standby processor is delayed until the first PPPoA session synchronizes to the standby processor. For each session, PPPoA synchronizes information elements such as virtual access (VAccess) descriptor, physical software for interface descriptor block (swidb) descriptor, switch handle, segment handle, and ATM virtual circuit's (VC) virtual path identifier (VPI) and virtual channel identifier (VCI) numbers to the standby processor. The **interface atm**keywords and *interface-number* argument specify a particular ATM interface by interface number and the **vc**keyword specifies the VC.

Note The debug pppatm redundancy command does not display output on the active processor during normal synchronization; that is, the command displays output on the active processor only during an error condition.

Note This command is used only by Cisco engineers for internal debugging of CCM processes.

Examples

The following is sample output from the debug pppatm redundancy command from a Cisco 10000 series router active processor, along with sample output from the **show pppatm redundancy** command from the standby processor. No field descriptions are provided because command output is used for Cisco internal debugging purposes only.

```
Router# debug pppatm redundancy
PPP over ATM redundancy debugging is on
Router-stby# show pppatm redundancy
0 : Session recreate requests from CCM
0 : Session up events invoked
0 : Sessions reaching PTA
0 : Sessions closed by CCM
0 : Session down events invoked
0 : Queued sessions waiting for base hwidb creation
0 : Sessions queued for VC up notification so far
0 : Sessions queued for VC encap change notification so far
0 : VC activation notifications received from ATM
0 : VC encap change notifications received from ATM
0 : Total queued sessions waiting for VC notification(Encap change+VC Activation)
```

Related Commands

Command	Description
debug pppatm	Enables debug reports for PPPoA events, errors, and states, either globally or conditionally, on an interface or VC.

debug pppoe

To display debugging information for PPP over Ethernet (PPPoE) sessions, use the **debugpppoe**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug pppoe {{**data**| **errors**| **events**| **packets**} [**rmac** *remote-mac-address*| **interface** *type number* [**vc** {[*vpi*/] *vci*| *vc-name*}] [**vlan** *vlan-id*]]| **elog**}

no debug pppoe {{**data**| **errors**| **events**| **packets**} [**rmac** *remote-mac-address*| **interface** *type number* [**vc** {[*vpi*/] *vci*| *vc-name*}] [**vlan** *vlan-id*]]| **elog**}

Syntax Description

data	Displays data packets of PPPoE sessions.
errors	Displays PPPoE protocol errors that prevent a session from being established, or displays errors that cause an established session to be closed.
events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
packets	Displays each PPPoE protocol packet that is exchanged.
rmac *remote-mac-address*	(Optional) Remote MAC address. Debugging information for PPPoE sessions sourced from this address will be displayed.
interface *type number*	(Optional) Interface for which PPPoE session debugging information will be displayed.
vc	(Optional) Displays debugging information for PPPoE sessions for a specific permanent virtual circuit (PVC).
vpi /	(Optional) ATM network virtual path identifier (VPI) for the PVC. The *vpi* value defaults to 0.
vci	(Optional) ATM network virtual channel identifier (VCI) for the PVC.
vc-name	(Optional) Name of the PVC.
vlan *vlan-id*	(Optional) IEEE 802.1Q VLAN identifier.
elog	Displays PPPoE error logs.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(13)T	This command was introduced. This command replaces the **debugvpdnpppoe-data**, **debugvpdnpppoe-error,debugvpdnpppoe-events,**and**debugvpdnpppoe-packet**commands available in previous Cisco IOS releases.
12.2(15)T	This command was modified to display debugging information on a per-MAC address, per-interface, and per-VC basis.
12.3(2)T	The **vlan***vlan-id* keyword and argument were added.
12.3(7)XI3	This command was integrated into Cisco IOS Release 12.3(7)XI3.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRC	This command was integrated into Cisco IOS Release 12.2(33)SRC.
Cisco IOS XE Release 2.1	This command was implemented on Cisco ASR 1000 series routers.

Examples

The following examples show sample output from the**debugpppoe** command:

```
Router# debug pppoe events interface atm 1/0.10 vc 101

PPPoE protocol events debugging is on
Router#
00:41:55:PPPoE 0:I PADI  R:00b0.c2e9.c470 L:ffff.ffff.ffff 0/101 ATM1/0.10
00:41:55:PPPoE 0:O PADO, R:00b0.c2e9.c470 L:0001.c9f0.0c1c 0/101 ATM1/0.10
00:41:55:PPPoE 0:I PADR  R:00b0.c2e9.c470 L:0001.c9f0.0c1c 0/101 ATM1/0.10
00:41:55:PPPoE :encap string prepared
00:41:55:[3]PPPoE 3:Access IE handle allocated
00:41:55:[3]PPPoE 3:pppoe SSS switch updated
00:41:55:[3]PPPoE 3:AAA unique ID allocated
00:41:55:[3]PPPoE 3:No AAA accounting method list
00:41:55:[3]PPPoE 3:Service request sent to SSS
00:41:55:[3]PPPoE 3:Created  R:0001.c9f0.0c1c L:00b0.c2e9.c470 0/101 ATM1/0.10
00:41:55:[3]PPPoE 3:State REQ_NASPORT    Event MORE_KEYS
00:41:55:[3]PPPoE 3:O PADS  R:00b0.c2e9.c470 L:0001.c9f0.0c1c 0/101 ATM1/0.10
00:41:55:[3]PPPoE 3:State START_PPP    Event DYN_BIND
00:41:55:[3]PPPoE 3:data path set to PPP
00:41:57:[3]PPPoE 3:State LCP_NEGO    Event PPP_LOCAL
00:41:57:PPPoE 3/SB:Sent vtemplate request on base Vi2
00:41:57:[3]PPPoE 3:State CREATE_VA    Event VA_RESP
00:41:57:[3]PPPoE 3:Vi2.1 interface obtained
00:41:57:[3]PPPoE 3:State PTA_BIND    Event STAT_BIND
00:41:57:[3]PPPoE 3:data path set to Virtual Acess
00:41:57:[3]PPPoE 3:Connected PTA
Router# debug pppoe errors interface atm 1/0.10
PPPoE protocol errors debugging is on
```

```
Router#
00:44:30:PPPoE 0:Max session count(1) on mac(00b0.c2e9.c470) reached.
00:44:30:PPPoE 0:Over limit or Resource low. R:00b0.c2e9.c470 L:ffff.ffff.ffff 0/101 ATM1/0.10
```

The following table describes the significant fields shown in the displays.

Table 47: debug pppoe Field Descriptions

Field	Description
PPPoE	PPPoE debug message header.
0:	PPPoE session ID.
I PADI	Incoming PPPoE Active Discovery Initiation packet.
R:	Remote MAC address.
L:	Local MAC address.
0/101	VPI VCI of the PVC.
ATM1/0.10	Interface type and number.
O PADO	Outgoing PPPoE Active Discovery Offer packet.
I PADR	Incoming PPPoE Active Discovery Request packet.
[3]	Unique user session ID. The same ID is used for identifying sessions across different applications such as PPPoE, PPP, Layer 2 Tunneling Protocol (L2TP), and Subscriber Service Switch (SSS). The same session ID appears in the output for the **showpppoesession**, **showssssession**, and **showvpdnsession** commands.
PPPoE 3	PPPoE session ID.
Created	PPPoE session is created.
O PADS	Outgoing PPPoE Active Discovery Session-confirmation packet.
Connected PTA	PPPoE session is established.
Max session count(1) on mac(00b0.c2e9.c470) reached	PPPoE session is rejected because of per-MAC session limit.

Related Commands

Command	Description
encapsulation aal5autoppp virtual-template	Enables PPPoA/PPPoE autosense.
pppoe enable	Enables PPPoE sessions on an Ethernet interface or subinterface.
protocol pppoe (ATM VC)	Enables PPPoE sessions to be established on PVCs.
show pppoe session	Displays information about active PPPoE sessions.
show sss session	Displays Subscriber Service Switch session status.
show vpdn session	Displays session information about L2TP, L2F protocol, and PPPoE tunnels in a VPDN.

debug pppoe redundancy

To debug PPP over Ethernet (PPPoE) redundancy events on a dual Route Processor High Availability (HA) system and display cluster control manager (CCM) events and messages, use the **debug pppoe redundancy**command in privileged EXEC mode. To disable the display of debugging output, use the **no** form of this command.

debug pppoe redundancy

no debug pppoe redundancy

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(31)SB2	This command was introduced.
12.2(33)SRC	This command was integrated into Cisco IOS Release 12.2(33)SRC.
Cisco IOS XE Release 3.3S	This command was integrated into Cisco IOS XE Release 3.3S.

Usage Guidelines

The CCM provides the capability to facilitate and synchronize session initiation on the standby processor of a dual Route Processor HA system. Use the **debug pppoe redundancy** command to display CCM events and messages for PPPoE sessions.

Note This command is used only by Cisco engineers for internal debugging of CCM processes.

Examples

The following is sample output from the **debug pppoe redundancy** command from a Cisco 10000 series router active processor. No field descriptions are provided because command output is used for Cisco internal debugging purposes only.

```
Router# debug pppoe redundancy
Nov 22 17:21:11.327: PPPoE HA[0xBE000008] 9: Session ready to sync data
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = PADR, length = 58
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = SESSION ID, length = 2
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = SWITCH HDL, length = 4
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = SEGMENT HDL, length = 4
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = PHY SWIDB DESC, length = 20
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = VACCESS DESC, length = 28
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: Sync collection for ready events
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = PADR, length = 58
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = SESSION ID, length = 2
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = SWITCH HDL, length = 4
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = SEGMENT HDL, length = 4
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = PHY SWIDB DESC, length = 20
Nov 22 17:21:11.351: PPPoE HA[0xBE000008] 9: code = VACCESS DESC, length = 28
```

The following is sample output from the **debug pppoe redundancy** command from a Cisco 10000 series router standby processor:

```
Router# debug pppoe redundancy
Nov 22 17:21:11.448: PPPoE HA[0x82000008]: Recreating session: retrieving data
Nov 22 17:21:11.464: PPPoE HA[0x82000008] 9: Session ready to sync data
```

The following is sample output from the **debug pppoe redundancy** command from a Cisco 7600 series router active processor.

```
Router# debug pppoe redundancy
Dec 17 15:14:37.060: PPPoE HA[0x131B01B1] 28039: Session ready to sync data
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = PADR, length = 48
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = SESSION ID, length = 2
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = SWITCH HDL, length = 4
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = SEGMENT HDL, length = 4
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = PHY SWIDB DESC, length = 20
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = VACCESS DESC, length = 28
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: Sync collection for ready events
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = PADR, length = 48
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = SESSION ID, length = 2
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = SWITCH HDL, length = 4
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = SEGMENT HDL, length = 4
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = PHY SWIDB DESC, length = 20
Dec 17 15:14:37.076: PPPoE HA[0x131B01B1] 28039: code = VACCESS DESC, length = 28
```

The following is sample output from the **debug pppoe redundancy** command from a Cisco 7600 series router standby processor:

```
Router-stby# debug pppoe redundancy
Dec 17 15:14:37.180: STDBY: PPPoE HA[0xE41B019B]: Recreating session: retrieving data
Dec 17 15:14:37.204: STDBY: PPPoE HA[0xE41B019B] 28039: Session ready to sync data
```

debug presence

To display debugging information about the presence service, use the **debug presence**command in privileged EXEC mode. To disable debugging messages, use the **no** form of this command.

debug presence {**all**| **asnl**| **errors**| **event**| **info**| **timer**| **trace**| **xml**}

no debug presence {**all**| **asnl**| **errors**| **event**| **info**| **timer**| **trace**| **xml**}

Syntax Description

all	Displays all presence debugging messages.
asnl	Displays trace event logs in the Application Subscribe Notify Layer (ASNL).
errors	Displays presence error messages.
event	Displays presence event messages.
info	Displays general information about presence service.
timer	Displays presence timer information.
trace	Displays a trace of all presence activities.
xml	Displays messages related to the eXtensible Markup Language (XML) parser for presence service.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.4(11)XJ	This command was introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.4(15)T	This command was integrated into Cisco IOS Release 12.4(15)T.

Examples

The following example shows output from the **debug presence asnl** command:

```
Router# debug presence asnl
*Sep  4 07:15:24.295: //PRESENCE:[17]:/presence_get_sccp_status: line is closed
```

```
*Sep  4 07:15:24.295: //PRESENCE:[17]:/presence_handle_line_update: line status changes,
send NOTIFY
*Sep  4 07:15:24.295: //PRESENCE:[17]:/presence_set_line_status: new line status [busy ]
*Sep  4 07:15:24.299: //PRESENCE:[17]:/presence_asnl_callback: type [5]
*Sep  4 07:15:24.299: //PRESENCE:[17]:/presence_asnl_callback: ASNL_RESP_NOTIFY_DONE
*Sep  4 07:15:24.299: //PRESENCE:[24]:/presence_get_sccp_status: line is closed
*Sep  4 07:15:24.299: //PRESENCE:[24]:/presence_handle_line_update: line status changes,
send NOTIFY
*Sep  4 07:15:24.299: //PRESENCE:[24]:/presence_set_line_status: new line status [busy ]
*Sep  4 07:15:24.299: //PRESENCE:[24]:/presence_asnl_callback: type [5]
*Sep  4 07:15:24.299: //PRESENCE:[24]:/presence_asnl_callback: ASNL_RESP_NOTIFY_DONE
*Sep  4 07:15:24.299: //PRESENCE:[240]:/presence_get_sccp_status: line is closed
*Sep  4 07:15:24.299: //PRESENCE:[240]:/presence_handle_line_update: line status changes,
send NOTIFY
*Sep  4 07:15:24.299: //PRESENCE:[240]:/presence_set_line_status: new line status [busy ]
*Sep  4 07:15:24.299: //PRESENCE:[766]:/presence_get_sccp_status: line is closed
*Sep  4 07:15:24.299: //PRESENCE:[766]:/presence_handle_line_update: line status changes,
send NOTIFY
*Sep  4 07:15:24.299: //PRESENCE:[766]:/presence_set_line_status: new line status [busy ]
*Sep  4 07:15:24.359: //PRESENCE:[766]:/presence_asnl_callback: type [5]
*Sep  4 07:15:24.359: //PRESENCE:[766]:/presence_asnl_callback: ASNL_RESP_NOTIFY_DONE
*Sep  4 07:15:24.811: //PRESENCE:[240]:/presence_asnl_callback: type [5]
*Sep  4 07:15:24.811: //PRESENCE:[240]:/presence_asnl_callback: ASNL_RESP_NOTIFY_DONE
*Sep  4 07:15:26.719: //PRESENCE:[17]:/presence_get_sccp_status: line is open
*Sep  4 07:15:26.719: //PRESENCE:[17]:/presence_handle_line_update: line status changes,
send NOTIFY
*Sep  4 07:15:26.719: //PRESENCE:[17]:/presence_set_line_status: new line status [idle ]
*Sep  4 07:15:26.719: //PRESENCE:[17]:/presence_asnl_callback: type [5]
*Sep  4 07:15:26.719: //PRESENCE:[17]:/presence_asnl_callback: ASNL_RESP_NOTIFY_DONE
*Sep  4 07:15:26.719: //PRESENCE:[24]:/presence_get_sccp_status: line is open
*Sep  4 07:15:26.719: //PRESENCE:[24]:/presence_handle_line_update: line status changes,
send NOTIFY
*Sep  4 07:15:26.719: //PRESENCE:[24]:/presence_set_line_status: new line status [idle ]
*Sep  4 07:15:26.723: //PRESENCE:[24]:/presence_asnl_callback: type [5]
*Sep  4 07:15:26.723: //PRESENCE:[24]:/presence_asnl_callback: ASNL_RESP_NOTIFY_DONE
```

The following example shows output from the **debug presence event**command:

```
Router# debug presence event
*Sep  4 07:16:02.715: //PRESENCE:[0]:/presence_sip_line_update: SIP nothing to update
*Sep  4 07:16:02.723: //PRESENCE:[17]:/presence_handle_notify_done: sip stack response code
 [29]
*Sep  4 07:16:02.723: //PRESENCE:[24]:/presence_handle_notify_done: sip stack response code
 [29]
*Sep  4 07:16:02.791: //PRESENCE:[240]:/presence_handle_notify_done: sip stack response
code [17]
*Sep  4 07:16:02.791: //PRESENCE:[766]:/presence_handle_notify_done: sip stack response
code [17]
*Sep  4 07:16:04.935: //PRESENCE:[0]:/presence_sip_line_update: SIP nothing to update
*Sep  4 07:16:04.943: //PRESENCE:[17]:/presence_handle_notify_done: sip stack response code
 [29]
*Sep  4 07:16:04.943: //PRESENCE:[24]:/presence_handle_notify_done: sip stack response code
 [29]
*Sep  4 07:16:04.995: //PRESENCE:[240]:/presence_handle_notify_done: sip stack response
code [17]
*Sep  4 07:16:04.999: //PRESENCE:[766]:/presence_handle_notify_done: sip stack response
code [17]
```

The following example shows output from the **debug presence info**command:

```
Router# debug presence info
*Sep  4 07:16:20.887: //PRESENCE:[17]:/presence_handle_line_update: get line status from
ccvdbPtr
*Sep  4 07:16:20.887: //PRESENCE:[17]:/presence_get_sccp_status: dn_tag 2
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
element <presence>
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
element <dm:person>
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
element <status>
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
element <e:activities>
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
```

```
element <tuple>
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
element <status>
*Sep  4 07:16:20.887: //PRESENCE:[16]:/presence_start_element_handler: line 1: unknown
element <e:activities>
*Sep  4 07:16:20.887: //PRESENCE:[0]:/presence_asnl_free_resp:
*Sep  4 07:16:20.887: //PRESENCE:[24]:/presence_handle_line_update: get line status from
ccvdbPtr
*Sep  4 07:16:20.887: //PRESENCE:[24]:/presence_get_sccp_status: dn_tag 2
*Sep  4 07:16:20.891: //PRESENCE:[23]:/presence_start_element_handler: line 1: unknown
element <presence>
```

The following example shows output from the **debug presence timer**command:

```
Router# debug presence timer
*Sep  4 07:16:41.271: //PRESENCE:[17]:/presence_asnl_notify_body_handler: expires time 3600
*Sep  4 07:16:41.271: //PRESENCE:[24]:/presence_asnl_notify_body_handler: expires time 3600
*Sep  4 07:16:41.271: //PRESENCE:[240]:/presence_asnl_notify_body_handler: expires time 607
*Sep  4 07:16:41.275: //PRESENCE:[766]:/presence_asnl_notify_body_handler: expires time 602
*Sep  4 07:16:43.331: //PRESENCE:[17]:/presence_asnl_notify_body_handler: expires time 3600
*Sep  4 07:16:43.331: //PRESENCE:[24]:/presence_asnl_notify_body_handler: expires time 3600
*Sep  4 07:16:43.331: //PRESENCE:[240]:/presence_asnl_notify_body_handler: expires time 605
*Sep  4 07:16:43.331: //PRESENCE:[766]:/presence_asnl_notify_body_handler: expires time 600
```

The following example shows output from the **debug presence trace**command:

```
Router# debug presence trace
*Sep  4 07:16:56.191: //PRESENCE:[17]:/presence_line_update:
*Sep  4 07:16:56.191: //PRESENCE:[24]:/presence_line_update:
*Sep  4 07:16:56.191: //PRESENCE:[240]:/presence_line_update:
*Sep  4 07:16:56.191: //PRESENCE:[766]:/presence_line_update:
*Sep  4 07:16:56.199: //PRESENCE:[17]:/presence_get_node_by_subid:
*Sep  4 07:16:56.199: //PRESENCE:[17]:/presence_handle_line_update:
*Sep  4 07:16:56.199: //PRESENCE:[17]:/presence_get_sccp_status:
*Sep  4 07:16:56.199: //PRESENCE:[17]:/presence_asnl_notify_body_handler:
*Sep  4 07:16:56.199: //PRESENCE:[24]:/presence_get_node_by_subid:
*Sep  4 07:16:56.199: //PRESENCE:[24]:/presence_handle_line_update:
*Sep  4 07:16:56.199: //PRESENCE:[24]:/presence_get_sccp_status:
*Sep  4 07:16:56.199: //PRESENCE:[24]:/presence_asnl_notify_body_handler:
*Sep  4 07:16:56.199: //PRESENCE:[240]:/presence_get_node_by_subid:
*Sep  4 07:16:56.199: //PRESENCE:[240]:/presence_handle_line_update:
*Sep  4 07:16:56.199: //PRESENCE:[240]:/presence_get_sccp_status:
*Sep  4 07:16:56.199: //PRESENCE:[240]:/presence_asnl_notify_body_handler:
*Sep  4 07:16:56.199: //PRESENCE:[766]:/presence_get_node_by_subid:
*Sep  4 07:16:56.203: //PRESENCE:[766]:/presence_handle_line_update:
*Sep  4 07:16:56.203: //PRESENCE:[766]:/presence_get_sccp_status:
*Sep  4 07:16:56.203: //PRESENCE:[766]:/presence_asnl_notify_body_handler:
*Sep  4 07:16:59.743: //PRESENCE:[17]:/presence_line_update:
*Sep  4 07:16:59.743: //PRESENCE:[24]:/presence_line_update:
*Sep  4 07:16:59.743: //PRESENCE:[240]:/presence_line_update:
*Sep  4 07:16:59.743: //PRESENCE:[766]:/presence_line_update:
```

The following example shows output from the **debug presence trace**command:

```
Router# debug presence trace
*Sep  4 07:17:17.351: //PRESENCE:[17]:/presence_xml_encode:
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_presence: keyword = presence
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_person: keyword = person
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_generic: keyword = Closed
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_activities: keyword = activities
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_otp: keyword = On-the-phone
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_tuple: keyword = tuple
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_status: keyword = status
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_generic: keyword = Closed
*Sep  4 07:17:17.355: //PRESENCE:[17]:/xml_encode_otp: keyword = On-the-phone
*Sep  4 07:17:17.355: <?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf" entity="sip:6003@1.4.171.34"
xmlns:e="urn:ietf:params:xml:ns:pidf:status:rpid"
xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model">
   <dm:person>
      <status>
         <basic>Closed</basic>
```

```
            </status>
            <e:activities>
               <e:on-the-phone/>
            </e:activities>
         </dm:person>
         <tuple id="cisco-cme">
            <status>
               <basic>Closed</basic>
               <e:activities>
                  <e:on-the-phone/>
               </e:activities>
            </status>
         </tuple>
      </presence>
```

Related Commands

Command	Description
presence	Enables presence service on the router and enters presence configuration mode.
presence enable	Allows the router to accept incoming presence requests.
show presence global	Displays configuration information about the presence service.
show presence subscription	Displays information about active presence subscriptions.

debug priority

To display priority queueing output, use the **debug priority**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug priority

no debug priority

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following example shows how to enable priority queueing output:

```
Router# debug priority
Priority output queueing debugging is on
```

The following is sample output from the **debug priority** command when the Frame Relay PVC Interface Priority Queueing (FR PIPQ) feature is configured on serial interface 0:

```
Router# debug priority
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 200 -> medium
00:49:05:PQ:Serial0 output (Pk size/Q 24/1)
00:49:05:PQ:Serial0 dlci 300 -> normal
00:49:05:PQ:Serial0 output (Pk size/Q 24/2)
00:49:05:PQ:Serial0 dlci 400 -> low
00:49:05:PQ:Serial0 output (Pk size/Q 24/3)
```

Related Commands

Command	Description
debug custom-queue	Displays custom queueing output .

debug private-hosts

To enable debug messages for the Private Hosts feature, use the **debug private-hosts** command in privileged EXEC mode.

debug private-hosts {**all**| **events**| **acl**| **api**}

Syntax Description

all	Enable debug messages for all Private Hosts errors and events.
events	Enable debug messages for issues related to Private Hosts events.
acl	Enable debug messages for issues and events related to ACLs.
api	Enable debug messages for issues related to the application programming interface.

Command Default

This command has no default settings.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SRB	This command was introduced.

Examples

The following example shows sample command output:

```
Router# debug private-hosts all

private-hosts events debugging is on
private-hosts api debugging is on
private-hosts acl debugging is on
Router#
```

Related Commands

Command	Description
debug fm private-hosts	Enables debug messages for the Private Hosts feature manager.

debug proxy h323 statistics

To enable proxy RTP statistics, use the **debug proxy h323 statistics**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug proxy h323 statistics

no debug proxy h323 statistics

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(2)NA	This command was introduced.

Usage Guidelines

Enter the **show proxy h323 detail-call** EXEC command to see the statistics.

debug pvcd

To display the permanent virtual circuit (PVC) Discovery events and Interim Local Management Interface (ILMI) MIB traffic used when discovering PVCs, use the **debug pvcd** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug pvcd

no debug pvcd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

This command is primarily used by Cisco technical support representatives.

Examples

The following is sample output from the **debug pvcd** command:

```
Router# debug pvcd
PVCD: PVCD enabled w/ Subif
PVCD(2/0): clearing event queue
PVCD: 2/0 Forgetting discovered PVCs...
PVCD: Removing all dynamic PVCs on 2/0
PVCD: Restoring MIXED PVCs w/ default parms on 2/0
PVCD: Marking static PVCs as UNKNWN on 2/0
PVCD: Marking static PVC 0/50 as UNKNWN on 2/0 ...
PVCD: Trying to discover PVCs on 2/0...
PVCD: pvcd_discoverPVCs
PVCD: pvcd_ping
PVCD: fPortEntry.5.0 = 2
PVCD: pvcd_getPeerVccTableSize
PVCD: fLayerEntry.5.0 = 13
PVCD:end allocating VccTable size 13
PVCD:  pvcd_getPeerVccTable
PVCD:******* 2/0: getNext on fVccEntry = NULL TYPE/VALUE numFileds = 19 numVccs = 13
PVCD: Creating Dynamic PVC 0/33 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/33: DYNAMIC
PVCD: After _create_pvc() VC 0/33: DYNAMIC0/33 on 2/0 :  UBR PCR = -1
PVCD: Creating Dynamic PVC 0/34 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/34: DYNAMIC
PVCD: After _create_pvc() VC 0/34: DYNAMIC0/34 on 2/0 :  UBR PCR -1
PVCD: Creating Dynamic PVC 0/44 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/44: DYNAMIC
PVCD: After _create_pvc() VC 0/44: DYNAMIC0/44 on 2/0 :  UBR PCR = -1
PVCD: PVC 0/50 with INHERITED_QOSTYPE
PVCD: _oi_state_change ( 0/50, 1 = ILMI_VC_UP )
PVCD: Creating Dynamic PVC 0/60 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/60: DYNAMIC
PVCD: After _create_pvc() VC 0/60: DYNAMIC0/60 on 2/0 :  UBR PCR = -1
PVCD: Creating Dynamic PVC 0/80 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/80: DYNAMIC
PVCD: After _create_pvc() VC 0/80: DYNAMIC0/80 on 2/0 :  UBR PCR = -1
PVCD: Creating Dynamic PVC 0/99 on 2/0
```

debug pvdm2dm

To view contents of packets flowing through PVDMII-xxDM digital modem devices, use the **debug pvdm2dm** command in privileged EXEC mode. To disable debug activity, use the **no** form of this command.

debug pvdm2dm {**packet modem**| **pvdm** *slot*/*port*| *pvdm slot*}

no debug pvdm2dm

Syntax Description

packet	Debugs packets
modem	Debugs modem packets
pvdm	Debugs PVDM packets
slot	Router slot for pvdms/modems
port	Modem number
pvdm slot	PVDM number

Command Default

Disabled

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(9)T	This command was introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

To debug the contents of modem packets for a specific modem, use the following command:

- **debug pvdm2dm packet modem** *<slot>/<port>*

By removing the specific modem number at the end, one can enable packet debugging for all the modems available on the router:

- **debug pvdm2dm packet modem**

The following command enables packet debugging for all packets flowing through a particular PVDMII-xxDM device:

- **debug pvdm2dm packet pvdm** *<slot>/<pvdm slot>*

The following command enables debugging of packets flowing through any PVDMII-xxDM device:

- **debug pvdm2dm packet pvdm**

The following command enables debugging of packets flowing through any PVDMII-xxDM device and any PVDMII-xxDM-based modem channel:

- **debug pvdm2dm packet**

To see what debug flags are set, and to view the contents of debugged packets, use the **show debugging** command.

Examples

The following example sets debugging for a specific modem. The following **show debugging** command displays the debug flag that is set, and gives a typical printout for one debugged packet:

```
Router# debug pvdm2dm packet modem 0/322
Router# show debugging
PVDM2 DM:
  Modem 0/322 packet debugging is on
Router#
May 24 17:35:16.318: pvdm2_dm_tx_dsp_pak_common: bay 0, dsp 0 May 24 17:35:16.318:
pvdm2_dm_dump_pak_hex: pak: 43E1F6FC size 8 May 24 17:35:16.318: 00 08 00 00 00 1C 00 00
May 24 17:35:16.322:
```

The following example sets debugging for all PVDMII-xxDM modems available on the router.

```
Router# debug pvdm2dm packet
Router# show debugging
PVDM2 DM:
  Modem 0/322 packet debugging is on
  Modem 0/323 packet debugging is on
  Modem 0/324 packet debugging is on
  .
  .
  .
  Modem 0/355 packet debugging is on
  Modem 0/356 packet debugging is on
  Modem 0/357 packet debugging is on
Router#
```

The following example sets debugging for a particular PVDMII-xxDM device.

```
Router# debug pvdm2dm packet pvdm 0/0
Router# show debugging
PVDM2 DM:
  PVDM2 0/0 packet debugging is on
Router#
```

The following example sets debugging for all PVDMII-xxDM devices in the router.

```
Router# debug pvdm2dm packet pvdm
Router# show debugging
PVDM2 DM:
  PVDM2 0/0 packet debugging is on
  PVDM2 0/1 packet debugging is on
  PVDM2 0/2 packet debugging is on
Router#
```

In all of these examples, the output describing the debugged packets is similar to that of the first example, except that the packet contents will vary.

Related Commands

Command	Description
show debugging	Displays information about the type of debugging enabled for your router.

debug pw-udp

To debug pseudowire User Datagram Protocol (UDP) virtual circuits (VCs), use the **debug pw-udp**command in privileged EXEC mode.

debug pw-udp {**errors**| **events**| **fsm**}

Syntax Description

errors	Specifies pseudowire UDP errors.
events	Specifies pseudowire UDP events.
fsm	Specifies pseudowire UDP finite state machine (FSM).

Command Default

Debugging for pseudowire UDP VCs is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(2)S	This command was introduced.

Usage Guidelines

To debug pseudowire UDP VCs, you must configure the **debug pw-udp**command in conjunction with the following set of **debug** commands before configuring Circuit Emulation Service over UDP (CESoUDP):

On both active and standby route processors (RPs):

- **debug xconnect event**
- **debug xconnect error**
- **debug acircuit event**
- **debug acircuit error**
- **debug acircuit checkpoint**
- **debug pw-udp checkpoint**
- **debug ssm cm events**
- **debug ssm cm errors**
- **debug ssm sm errors**
- **debug ssm sm events**

- **debug sss error**
- **debug sss event**
- **debug sss fsm**
- **debug cem ac event**
- **debug cem ac error**
- **debug cem ha event**
- **debug cem ha error**

On the Circuit Emulation over Packet (CeOP) line card:

- **debug ssm cm events**
- **debug ssm cm errors**
- **debug ssm sm errors**
- **debug ssm sm events**

For more information about each of these debug commands, see the Cisco IOS Debug Command Reference Guide.

Examples

The following example shows how to debug pseudowire UDP VCs on the active RP:

```
Router#
debug xconnect event
Xconnect author event debugging is on
Router#
debug xconnect error
Xconnect author errors debugging is on
Router#
debug acircuit event
Attachment Circuit events debugging is on
Router#
debug acircuit error
Attachment Circuit errors debugging is on
Router#
debug cem ac event
CEM AC Events debugging is on
Router#
debug cem ac error
CEM AC Error debugging is on
Router#
debug cem ha event
CEM redundancy events debugging is on
Router#
debug cem ha error
CEM redundancy error debugging is on
Router#
debug pw-udp event
PW UDP events debugging is on
Router#
debug pw-udp error
PW UDP errors debugging is on
Router#
debug pw-udp fsm
PW UDP fsm debugging is on
Router#
debug ssm cm events
SSM Connection Manager events debugging is on
```

```
Router#
debug ssm cm errors
SSM Connection Manager errors debugging is on
Router#
debug ssm sm errors
SSM Segment Handler Manager errors debugging is on
Router#
debug ssm sm events
SSM Segment Handler Manager events debugging is on
Router#
debug sss error
SSS Manager errors debugging is on
Router#
debug sss event
SSS Manager events debugging is on
Router#
debug sss fsm
SSS Manager fsm debugging is on
Router#
00:05:01: STDBY: CEMHA RF: CID 116, Seq 219, Event RF_EVENT_CLIENT_PROGRESSION, Op 7, State
 STANDBY COLD-BULK, Peer ACTIVE
00:05:01: STDBY: CEMHA CF: CF client 182, entity 0 received msg
00:05:01: STDBY: CEMHA CF: CF client 182, entity 0 received msg
00:05:01: STDBY: CEMHA CF: CF client 182, entity 0 received msg
00:05:01: STDBY: CEMHA CF: CF client 182, entity 0 received msg
00:05:01: STDBY: CEMHA CF: CF client 182, entity 0 received msg
00:05:01: STDBY: CEMHA CF: CF client 182, entity 0 recei
00:05:01: STDBY: CEMHA CF Received Interface Update event=0x10
00:05:01: STDBY: AC CESP[CE4/2/0]: Activated CEM group 0
00:05:01: STDBY: AC CESP[CE4/2/0]: Setup switching of ckt 0
00:05:01: STDBY: AC CESP ERROR[CE4/2/0]: (CEM4/2/0): Setup Switching 0 cannot proceed sw/seg:
 0/0, Flag 10, SSM 0
00:05:01: STDBY: AC CESP ERROR[CE4/2/0]: CEM 0 Data switching setup failed
00:05:01: STDBY: CEMHA CF Received T1/E1 Update event=0x20
00:05:01: STDBY: CEMHA CF Received Interface Update event=0x10
00:05:01: STDBY: AC CESP[CE4/2/1]: Activated CEM group 0
00:05:01: STDBY: AC CESP[CE4/2/1]: Setup switching of ckt 0
00:05:01: STDBY: AC CESP ERROR[CE4/2/1]: (CEM4/2/1): Setup Switching 0 cannot proceed sw/seg:
 0/0, Flag 10, SSM 0
00:05:01: STDBY: AC CESP ERROR[CE4/2/1]: CEM 0 Data switching setup failed
00:05:01: STDBY: CEMHA CF Received T1/E1 Update event=0x20
00:05:01: STDBY: CEMHA(CEM4/2/1):Decode received VC AC for evtype 8 cem_id = 0,
 pw_state = 1, seg 3007, switch 2002, ac_wait_flags = 10 ,is_standby = NO, red_seg 0,
red_switch 0
00:05:01: STDBY: CEMHA: cem_id0, before decode sw/segment: 0/0, seg_state = 2, red sw/segment:
 0/0
00:05:01: STDBY: SSM SM ID LOCK: [CEM HA:id_lock_util_init:0] locker <ALL>: instance created
 for <SSM SM ID LOCK>
00:05:01: STDBY: SSM CM[12295]: reserve ID: Locking SSM ID
00:05:01: STDBY: SSM SM ID LOCK: [CEM HA:id_lock:12295] locker <SIP>: count 0 --> 1
00:05:01: STDBY: CEMHA CF Received Interface Update event=0x10
00:05:01: STDBY: AC CESP[CE4/2/1]: Activated CEM group 0
00:05:01: STDBY: CEMHA CF Received T1/E1 Update event=0x20
00:05:01: STDBY: CEMHA CF Received Interface Update event=0x10
00:05:01: STDBY: AC CESP[CE4/2/1]: Activated CEM group 0
00:05:01: STDBY: CEMHA CF Received T1/E1 Update event=0x20
00:05:01: STDBY: CEMHA CF: Received bulk sync complete - sending ack
00:05:01: STDBY: CEMHA: Create CEM Circuit verification Background process...
00:05:01: STDBY: SSM CM: reserve seg(12295) sw(8194) IDs
00:05:01: STDBY: CEMHA : CEM HA Background Process
00:05:02: STDBY: CEMHA: CF sync successfully completed
00:05:03: STDBY: XCL2 CID 119 Seq 224 Event RF_EVENT_CLIENT_PROGRESSION Op 7 State STANDBY
 COLD-BULK Peer ACTIVE
00:05:03: STDBY: PW UDP HA: HA Coexistence. Skip ISSU Negotiation on standby RP
00:05:06: STDBY: CEM HA: (CEM4/2/0) CEM 0x0 Platform chkpt data has
                     arrived for cktid=0
00:05:06: STDBY: CEM PW: Remove from WaitQ, ckt_type 19
00:05:06: STDBY: CEM HA: (CEM4/2/1) CEM 0x0 Platform chkpt data has
                     arrived for cktid=0
00:05:06: STDBY: CEM PW: Remove from WaitQ, ckt_type 19
00:05:06: STDBY: AC CESP[CE4/2/1]: Setup switching of ckt 0
00:05:06: STDBY: AC: [CE4/2/1, 0]: Setup switching
00:05:06: STDBY: AC: [CE4/2/1, 0]: Our AIE EF000002 Peer's AIE 2B000004 Peer's peer 00000000
```

```
00:05:06: STDBY: AC: [CE4/2/1, 0]: Using switch hdl 8194
00:05:06: STDBY: SSM CM[12295]: provision segment: standby RP received existing id from
active RP
00:05:06: STDBY: AC: [CE4/2/1, 0]: Successfully setup switching API
00:05:06: STDBY: AC: [CE4/2/1, 0]: Allocated segment hdl 12295
00:05:06: STDBY: AC CESP[CE4/2/1]: CKT UP ID: 0
00:05:06: STDBY: AC CESP[CE4/2/1]: Send ACMGR NOTIF, ckt_type 19, ckt_id 0 UP
00:05:06: STDBY: AC: Update seg 12295 plane with circuit Up status
00:05:06: STDBY: SSM SH[12295]: X: alloc sbase 0x500386A0 hdl 3007
00:05:06: STDBY: SSM CM[12295]: [CESoPSN Basic] provision first allocated base now, reserved
 earlier
00:05:06: STDBY: SSM CM[12295]: CM FSM: st Idle, ev Prov seg->Down
00:05:06: STDBY: SSM SH[12295]: init segment base
00:05:06: STDBY: SSM SH[ADJ:CESoPSN Basic:12295]: init segment class
00:05:06: STDBY: SSM CM[ADJ:CESoPSN Basic:12295]: provision segment 1
00:05:06: STDBY: SSM SM[ADJ:CESoPSN Basic:12295]: Provision segment: Idle -> Prov
00:05:06: STDBY: SSM SM[ADJ:CESoPSN Basic:12295]: provision segment
00:05:06: STDBY: SSM CM[12295]: segment status update Up
00:05:06: STDBY: SSM CM[12295]: CM FSM: st Down, ev Upd seg->Down
00:05:06: STDBY: SSM CM[ADJ:CESoPSN Basic:12295]: update segment status
00:05:06: STDBY: SSM SM[ADJ:CESoPSN Basic:12295]: Update segment: no state change, Prov
00:05:06: STDBY: SSM ADJ[ADJ:CESoPSN Basic:CE4/2/1: Type L:12295]: update segment status:
Up
00:05:06: STDBY: SSM ADJ[ADJ:CESoPSN Basic:CE4/2/1: Type L:12295]: ATM Async is supported
00:05:06: STDBY: SSM ADJ[ADJ:CESoPSN Basic:CE4/2/1: Type L:12295]: Platform requesting not
 to send unready: 1
00:05:06: STDBY: SSM ADJ[ADJ:CESoPSN Basic:CE4/2/1: Type L:12295]: circuit Up event
00:05:06: STDBY: SSM ADJ[ADJ:CESoPSN Basic:CE4/2/1: Type L:12295]: send segment ready
00:05:06: STDBY: SSM CM[12295]: [ADJ] shQ request send ready event
00:05:06: STDBY: ACMGR [CE4/2/1]: Receive <Circuit Status> msg
00:05:06: STDBY: ACMGR [CE4/2/1]: circuit up event, FSP state chg sip up to both up, action
 is peer p2p up, circuit remote up
00:05:06: STDBY: SSS MGR [uid:4]: Handling peer-to-peer event
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: receive p2p msg type: circuit status
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: Success to obtain circuit type 19 from AC
Access IE
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: event local ac up, state changed from
provisioned to activating, action local_ac_up
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: Waiting for vc checkpoint data
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: Success to obtain circuit type 19 from AC
Access IE
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: event need checkpoint, state changed from
activating to checkpoint wait, action clean_up_checkpoint_resource
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: Cleanup Checkpoint Resources
00:05:06: STDBY: PW UDP MGR [10.1.1.153, 200]: local ac status is changed from none to UP
00:05:06: STDBY: SSM CM[12295]: SM msg event send ready event
00:05:06: STDBY: SSM SM[ADJ:CESoPSN Basic:12295]: segment ready
00:05:06: STDBY: SSM SM[ADJ:CESoPSN Basic:12295]: Found segment data: Prov -> Ready
00:05:07: STDBY: CEMHA RF: CID 116, Seq 219, Event RF_EVENT_CLIENT_PROGRESSION, Op 8, State
 STANDBY HOT, Peer ACTIVE
00:05:07: STDBY: XCL2 CID 119 Seq 224 Event RF_EVENT_CLIENT_PROGRESSION Op 8 State STANDBY
 HOT Peer ACTIVE
00:05:07: STDBY: PW UDP HA: HA Coexistence. Skip ISSU Negotiation on standby RP
```

Related Commands

Command	Description
encapsulation (pseudowire)	Specifies an encapsulation type for tunneling Layer 2 traffic over a pseudowire.
udp port	Configures the UDP port information on the xconnect class.
show pw-udp vc	Displays information about pseudowire UDP VCs.

debug pxf atom

To display debug messages relating to Parallel eXpress Forwarding (PXF) Any Transport over MPLS (AToM), use the debug pxf atom command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf atom [**ac**| **mpls**]

no debug pxf atom [**ac**| **mpls**]

Syntax Description

ac	(Optional) Displays AToM information related to attachment circuit (AC) events.
mpls	(Optional) Displays AToM information related to MPLS Forwarding Infrastructure (MFI) events.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF AToM AC events debug messages:

```
Router# debug pxf atom ac
PXF ATOM AC debugging is on
```

Related Commands

Command	Description
show mpls l2transport	Displays information about AToM virtual circuits (VCs) that have been enabled to route Layer 2 packets on a router, including platform-independent AToM status and capabilities of a particular interface.
show mpls l2transport vc	Displays information about AToM VCs that are enabled to route Layer 2 packets on a router.
show pxf cpu atom	Displays PXF AToM information for an interface or VCCI.

Command	Description
show pxf cpu mpls label	Displays PXF forwarding information for a label.
show pxf cpu statistics atom	Displays PXF CPU AToM statistics.

debug pxf backwalks

To display debug messages relating to Parallel eXpress Forwarding (PXF) backwalk requests, use the debug pxf backwalks command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf backwalks

no debug pxf backwalks

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF backwalk requests debug messages:

```
Router# debug pxf backwalks
PXF BACKWALK debugging is on
```

Related Commands

Command	Description
show pxf cpu statistics backwalk	Displays PXF CPU backwalk requests statistics.

debug pxf bba

To display debug messages relating to Parallel eXpress Forwarding (PXF) Broadband Access Aggregation (BBA) features, use the debug pxf bba command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf bba [**ac sh_counter**| **ac sh_error**| **ac sh event**| **elog**| **l2f startstop debug**| **l2x fh error**| **l2x fh event**| **l2x sh counter**| **l2x sh error**| **l2x sh event**| **lt sh error**| **lt sh event**]

no debug pxf bba [**ac sh_counter**| **ac sh_error**| **ac sh event**| **elog**| **l2f startstop debug**| **l2x fh error**| **l2x fh event**| **l2x sh counter**| **l2x sh error**| **l2x sh event**| **lt sh error**| **lt sh event**]

Syntax Description

ac_sh_counter	(Optional) Displays attachment circuit (AC) segment counters.
ac_sh_error	(Optional) Displays AC segment errors.
ac_sh_event	(Optional) Displays AC segment events.
elog	(Optional) Displays event logging messages.
l2f_startstop_debug	(Optional) Displays Layer 2 Forwarding (L2F) tunneling events.
l2x_fh_error	(Optional) Displays L2F/L2TP (L2x) feature errors.
l2x_fh_event	(Optional) Displays L2x feature events.
l2x_sh_counter	(Optional) Displays L2x segment counters.
l2x_sh_error	(Optional) Displays L2x segment errors.
l2x_sh_event	(Optional) Displays L2x segment events.
lt_sh_error	(Optional) Displays LT segment errors.
lt_sh_event	(Optional) Displays LT segment events.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display AC segment counters debug messages:

```
Router# debug pxf bba ac_sh_counter
AC segment counters debugging is on
*Jan 19 13:18:26.698: c10k_get_ac_segment_counters: pppox vcci 2709 rx pkts = 0
rx byte = 0 tx pkts = 0 tx bytes = 0
*Jan 19 13:18:26.698: c10k_get_ac_segment_counters: pppox vcci 2709 tx drop pkts
 = 0 tx drop bytes = 0
*Jan 19 13:18:26.698: c10k_get_ac_segment_counters: pppox vcci 2710 rx pkts = 0
rx byte = 0 tx pkts = 0 tx bytes = 0
*Jan 19 13:18:26.698: c10k_get_ac_segment_counters: pppox vcci 2710 tx drop pkts
 = 0 tx drop bytes = 0
*Jan 19 13:18:36.698: c10k_get_ac_segment_counters: pppox vcci 2709 rx pkts = 0
rx byte = 0 tx pkts = 0 tx bytes = 0
*Jan 19 13:18:36.698: c10k_get_ac_segment_counters: pppox vcci 2709 tx drop pkts
 = 0 tx drop bytes = 0
*Jan 19 13:18:36.698: c10k_get_ac_segment_counters: pppox vcci 2710 rx pkts = 0
rx byte = 0 tx pkts = 0 tx bytes = 0
.
.
.
```

The following example shows how to display L2F tunneling debug messages:

```
Router# debug pxf bba l2f_startstop_debug
L2F feature debugging is on
*Jan 20 12:04:18.976: hwcnts.rx_pkts :0 hwcnts.rx_bytes :0
hwcnts.tx_pkts :0 hwcnts.tx_bytes: 0
hwcnts.tx_drop_pkts :0 hwcnts.tx_drop_bytes: 0
pcntrs->rx_pkts: 0 pcntrs->rx_bytes: 0
pcntrs->tx_pkts: 0 pcntrs->tx_bytes: 0
pcntrs->tx_drop_pkts: 0 pcntrs->tx_drop_bytes: 0
*Jan 20 12:04:18.976: hwcnts.rx_pkts :0 hwcnts.rx_bytes :0
hwcnts.tx_pkts :0 hwcnts.tx_bytes: 0
hwcnts.tx_drop_pkts :0 hwcnts.tx_drop_bytes: 0
pcntrs->rx_pkts: 0 pcntrs->rx_bytes: 0
pcntrs->tx_pkts: 0 pcntrs->tx_bytes: 0
pcntrs->tx_drop_pkts: 0 pcntrs->tx_drop_bytes: 0
.
.
.
```

Related Commands

Command	Description
show pxf cpu bba	Displays PXF CPU (RP) BBA information.

debug pxf cef

To display debug messages relating to Parallel eXpress Forwarding (PXF) Cisco Express Forwarding (CEF), use the debug pxf cef command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf cef [**fibroot**| **rpf**]

no debug pxf cef [**fibroot**| **rpf**]

Syntax Description

fibroot	Displays PXF CEF Forwarding Information Base (FIB) root information.
rpf	Displays PXF CEF Reverse Path Forwarding (RPF) information.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF CEF debug messages:

```
Router# debug pxf cef
PXF CEF debugging is on
```

Related Commands

Command	Description
show ip cef	Displays summary information about the FIB entries.
show pxf cpu cef	Displays PXF CPU memory usage, CEF, and External Column Memory (XCM) information.

debug pxf dma

To display debug messages relating to Parallel eXpress Forwarding (PXF) direct memory access (DMA) operations, use the debug pxf dma command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf dma

no debug pxf dma

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)XI	This command was introduced.
12.2(31)SB	This command was integrated into Cisco IOS Release 12.2(31)SB.

Examples

The following example shows how to display PXF DMA ASIC debug messages:

```
Router# debug pxf dma
PXF DMA ASIC debugging is on
*Jan  4 08:05:06.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:06.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:07.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:07.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:08.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:08.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:09.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:09.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:10.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:10.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:11.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:11.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:12.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:12.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:13.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:13.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:14.314: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:14.814: get ftbb reg: slot 3, subslot 1
*Jan  4 08:05:14.982: Entering c10k_cobalt_send.
*Jan  4 08:05:14.982:   Packet decode: datagramstart 0x0A0301BE length 76
*Jan  4 08:05:14.982:   Header decode: Chan 0, VCCI 2515
*Jan  4 08:05:14.982:   Header decode: flags 0x0001
*Jan  4 08:05:14.982:   c10k_cobalt_send: Checked the idb state.
*Jan  4 08:05:14.982:   c10k_cobalt_send: Checked the FromRP Q count.
.
```

```
.
.
```

Related Commands

Command	Description
show pxf dma	Displays the current state of the DMA buffers, error counters, and registers on the PXF.

debug pxf iedge

To display debug messages relating to Parallel eXpress Forwarding (PXF) Intelligent Edge (iEdge) operations, use the debug pxf iedge command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf iedge [stats]

no debug pxf iedge [stats]

Syntax Description

stats	(Optional) Includes PXF iEdge statistics in the output.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF iEdge debug messages:

```
Router# debug pxf iedge
iEdge Feature Debug debugging is on
```

Related Commands

Command	Description
show pxf cpu iedge	Displays PXF iEdge information for an interface or policy.

debug pxf ipv6

To display debug messages relating to Parallel eXpress Forwarding (PXF) IPv6 provisioning, use the debug pxf ipv6 command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf ipv6 [**acl**| **fib**| **hash**]

no debug pxf ipv6 [**acl**| **fib**| **hash**]

Syntax Description

acl	(Optional) Displays PXF IPv6 access control list (ACL) information.
fib	(Optional) Displays PXF Forwarding Information Base (FIB) information.
hash	(Optional) Displays PXF IPv6 hash information.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF IPv6 ACL debug messages:

```
Router# debug pxf ipv6 acl
PXF IPV6 ACL debugging is on
```

Related Commands

Command	Description
show ipv6 interface	Displays IPv6 interface settings.
show ipv6 route	Displays IPv6 routing table contents.
show pxf cpu ipv6	Displays PXF CPU IPv6 statistics.

debug pxf l2less-error

To display debug messages relating to Parallel eXpress Forwarding (PXF) Layer 2 Less (L2less) drop packet errors, use the debug pxf l2less-error command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf l2less-error

no debug pxf l2less-error

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)XI	This command was introduced.

Usage Guidelines

The Route Processor (RP) uses the L2less packet handler to handle tunneling encapsulated packets that do not have the original IP and Layer 2 information associated with them. The L2less handler takes the packet with a specific header, updates the statistics (interface packet and byte counts), and enqueues the packet to the IP input queue.

Examples

The following example shows how to display PXF L2less drop packet errors debug messages:

```
Router# debug pxf l2less-error
PXF l2less-error debugging is on
```

Related Commands

Command	Description
show pxf statistics	Displays chassis-wide, summary PXF statistics.

debug pxf microcode

To display debug message relating to Parallel eXpress Forwarding (PXF) microcode operations, use the debug pxf microcode command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf microcode

no debug pxf microcode

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)XI	This command was introduced.

Examples

The following example shows how to display PXF microcode debug messages:

```
Router# debug pxf microcode
PXF microcode debugging is on
```

Related Commands

Command	Description
microcode reload	Reloads the Cisco IOS image from a line card on a Cisco router.
show pxf microcode	Displays identifying information for the microcode currently loaded on the PXF.

debug pxf mnode

To display debug messages relating to Parallel eXpress Forwarding (PXF) multiway node (mnode) operations, use the debug pxf mnode command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf mnode

no debug pxf mnode

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Usage Guidelines

The mnodes are used in the multiway tree (Mtrie) library. Each mnode has a number of buckets that point to lower level mnodes or to multiway leaves (mleaves). The mleaves can be null leaves which indicate empty buckets.

Examples

The following example shows how to display PXF mnode debug messages:

```
Router# debug pxf mnode
PXF MNODE debugging is on
```

Related Commands

Command	Description
show pxf cpu cef	Displays PXF CPU memory usage, Cisco Express Forwarding, and XCM information.

debug pxf mpls

To display debug messages relating to Parallel eXpress Forwarding (PXF) Multiprotocol Label Switching (MPLS) operations, use the debug pxf mpls command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf mpls [**csc** {**event**| **stats**}| **lspv**]

no debug pxf mpls [**csc** {**event**| **stats**}| **lspv**]

Syntax Description

csc {**event** | **stats**}	(Optional) Displays PXF Cisco Signaling Controller (CSC) events and statistics.
lspv	Displays Link State Path Vector (LSPV) debug messages from the PXF MPLS Label Switched Path (LSP) Ping/Traceroute feature.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF MPLS CSC statistics debug messages:

```
Router# debug pxf mpls csc stats
PXF MPLS CSC STATS debugging is on
```

Related Commands

Command	Description
ping mpls	Checks MPLS LSP connectivity.
show mpls interfaces	Displays information about the interfaces that have been configured for label switching.
show pxf cpu mpls	Displays PXF MPLS (FIB) entry information.
trace mpls	Discovers MPLS LSP routes that packets will take when traveling to their destinations.

debug pxf mroute

To display debug messages relating to Parallel eXpress Forwarding (PXF) multicast route (mroute) operations, use the debug pxf mroute command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf mroute [**mdb**| **mdt**| **midb**| **punt**]

no debug pxf mroute [**mdb**| **mdt**| **midb**| **punt**]

Syntax Description

mdb	(Optional) Displays PXF multicast descriptor block (MDB) event messages.
mdt	(Optional) Displays PXF multicast distribution tree (MDT) messages.
midb	(Optional) Displays PXF multicast interface descriptor block (MIDB) messages.
punt	(Optional) Displays PXF multicast punted packets information.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF multicast distribution tree (MDT) debug messages:

```
Router# debug pxf mroute mdt
PXF mroute mdt creation debugging is on
```

Related Commands

Command	Description
clear ip mroute	Deletes entries from the IP multicast routing table.
show ip mroute	Displays the contents of the IP multicast routing table.

Command	Description
show pxf cpu mroute	Displays PXF multicast routing information for a particular group or range of groups.

debug pxf multilink

To display debug messages relating to Parallel eXpress Forwarding (PXF) multilink operations, use the debug pxf multilink command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf multilink [**all**| **atm**| **frame-relay**| **frfl2**| **lfi**| **ppp**| **queue**| **rates**]

no debug pxf multilink [**all**| **atm**| **frame-relay**| **frfl2**| **lfi**| **ppp**| **queue**| **rates**]

Syntax Description

all	(Optional) Displays all PXF multilink messages.
atm	(Optional) Displays PXF multilink ATM messages.
frame-relay	(Optional) Displays PXF multilink Frame Relay messages.
frf12	(Optional) Displays PXF Frame Relay Forum FRF.12-based fragmentation information on Frame Relay permanent virtual circuits (PVCs).
lfi	(Optional) Displays PXF Link Fragmentation and Interleaving (LFI) messages.
ppp	(Optional) Displays PXF multilink PPP messages.
queue	(Optional) Displays PXF multilink queue messages.
rates	(Optional) Displays PXF multilink queue rate messages.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF multilink ATM debug messages:

```
Router# debug pxf multilink atm
Router#
```

Related Commands

Command	Description
frame-relay fragment	Enables fragmentation of Frame Relay frames on a Frame Relay map class.
show ppp multilink	Displays bundle information for the MLP bundles.
show pxf statistics	Displays chassis-wide, summary PXF statistics.

debug pxf netflow

To enable debugging of NetFlow Parallel eXpress Forwarding (PXF)operations, use the debug pxf netflow command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf netflow {**records**| **time**}

no debug pxf netflow {**records**| **time**}

Syntax Description

records	Displays NetFlow PXF records information.
time	Displays NetFlow PXF time synchronization information.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)XI	This command was introduced.

Examples

The following example enables NetFlow PXF records debugging:

```
Router# debug pxf netflow records
PXF netflow records debugging is on
Router#
```

Related Commands

Command	Description
show pxf netflow	Displays NetFlow PXF counters information.

debug pxf pbr

To display debug messages relating to Parallel eXpress Forwarding (PXF) policy-based routing (PBR), use the debug pxf pbr command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf pbr [**sacl**| **trace**]

no debug pxf pbr [**sacl**| **trace**]

Syntax Description

sacl	(Optional) Displays PXF PBR super access control list (ACL) messages.
trace	(Optional) Displays PXF PBR trace information.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF PBR trace debug messages:

```
Router# debug pxf pbr trace
PXF PBR Trace debugging is on
```

Related Commands

Command	Description
show pxf cpu pbr action	Displays the PBR actions configured on the PXF for all PBR route maps.

debug pxf qos

To display debug messages relating to Parallel eXpress Forwarding (PXF) quality of service (QoS) operations, use the debug pxf qos command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf qos [**ipc**| **trace**]

no debug pxf qos [**ipc**| **trace**]

Syntax Description

ipc	(Optional) Displays PXF QoS interprocess communication (IPC) information.
trace	(Optional) Displays PXF QoS trace information

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2S	This command was introduced.

Examples

The following example shows how to display PXF QoS IPC debug messages:

```
Router# debug pxf qos trace
PXF QoS IPC Events debugging is on
Router#
*Apr 30 23:23:44:  c10k_bandwidth_notification_handler: cmdtype=4 event=0x30 acA
*Apr 30 23:23:44:  c10k_priority_notification_handler: cmdtype=4 event=0x30 actA
*Apr 30 23:23:44:  c10k_bandwidth_notification_handler: cmdtype=4 event=0x30 acA
*Apr 30 23:23:44:  c10k_bandwidth_notification_handler: cmdtype=4 event=0x30 acA
*Apr 30 23:23:44:  c10k_priority_notification_handler: cmdtype=4 event=0x30 actA
*Apr 30 23:23:44:  c10k_bandwidth_notification_handler: cmdtype=4 event=0x30 acA
.
.
.
```

Related Commands

Command	Description
show pxf cpu qos	Displays External Column Memory (XCM) contents related to a particular policy.
show pxf statistics	Displays chassis-wide, summary PXF statistics.

debug pxf stats

To display debug messages relating to Parallel eXpress Forwarding (PXF) statistics collector events, use the debug pxf stats command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf stats

no debug pxf stats

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)XI	This command was introduced.

Examples

The following example shows how to display PXF statistics debug messages:

```
Router# debug pxf stats
PXF hardware statistics debugging is on
```

Related Commands

Command	Description
clear pxf	Clears PXF counters and statistics.
show pxf cpu statistics	Displays PXF CPU statistics.
show pxf statistics	Displays chassis-wide, summary PXF statistics.

debug pxf subblocks

To display debug messages relating to Parallel eXpress Forwarding (PXF) bridged subinterfaces (encapsulation types), use the debug pxf subblocks command in privileged EXEC mode. To disable the debugging, use the no form of this command.

debug pxf subblocks

no debug pxf subblocks

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled (debugging is not enabled).

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)XI	This command was introduced.

Examples

The following example shows how to display PXF bridged subinterfaces (encapsulation type) debug messages:

```
Router# debug pxf subblocks
PXF hardware subblock debugging is on
```

Related Commands

Command	Description
show pxf cpu statistics	Displays PXF CPU statistics.
show pxf cpu subblocks	Displays PXF CPU statistics for bridged subinterfaces (encapsulation types).

debug pxf tbridge

To enable debugging of Parallel eXpress Forwarding (PXF) transparent bridging, use the **debug pxf tbridge**command in privileged EXEC mode. To disable debugging for the PXF transparent bridge, use the **no** form of this command.

debug pxf tbridge

no debug pxf tbridge

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.
12.2(31)SB	This command was integrated into Cisco IOS Release 12.2(31)SB and implemented on the Cisco 10000 series router.

Examples

The following sample output from the **debug pxf tbridge** command shows that the Bridge Group Virtual Interface (BVI) 100 has been removed from the Software Mac-address Filter (SMF) table:

```
Router# debug pxf tbridge

*Feb  8 18:39:04.710: rpmxf_tbridge_add_remove_bvi_from_smf: Deleting BVI entry 100 from
SMF table.
*Feb  8 18:39:04.710: rpmxf_tbridge_add_remove_bvi_from_smf: BVI 100 ICM programming
*Feb  8 18:39:04.710: rpmxf_tbridge_add_remove_bvi_from_smf: Successfully removed SMF entry
 for bvi 100
*Feb  8 18:39:04.710: rpmxf_tbridge_add_remove_bvi_from_smf: Deleting BVI entry 100 from
SMF table.
*Feb  8 18:39:04.710: rpmxf_tbridge_add_remove_bvi_from_smf: BVI 100 ICM programming
*Feb  8 18:39:04.710: rpmxf_tbridge_add_remove_bvi_from_smf: Successfully removed SMF entry
 for bvi 100
*Feb  8 18:39:05.178: %SYS-5-CONFIG_I: Configured from console by vty0
(CROI_MASTER_000A004B)
*Feb  8 18:39:06.710: %LINK-5-CHANGED: Interface BVI100, changed state to administratively
 down
*Feb  8 18:39:07.710:%LINEPROTO-5-UPDOWN: Line protocol on Interface BVI100, changed state
 to down
```

The following sample output from the **debug pxf tbridge** command shows that BVI is configured and that the SMF entry has been updated:

```
Router# debug pxf tbridge

*Feb  8 18:39:16.398:
```

```
Note: A random mac address of 0000.0ceb.c0f8 has been chosen for BVI in bridge group  100
since there is no mac address associated with the selected interface.
*Feb  8 18:39:16.398: Ensure that this address is unique.
*Feb  8 18:39:16.398: rpmxf_tbridge_smf_update: SMF update for Switch1.1: BVI 100 Mac Address
 0000.0ceb.c0f8
*Feb  8 18:39:16.398: rpmxf_tbridge_smf_update: BVI 100 ICM programming
*Feb  8 18:39:16.398: rpmxf_tbridge_smf_update: Successfully updated SMF entry for bvi 100
*Feb  8 18:39:16.398: rpmxf_tbridge_smf_update: SMF update for Switch1.1:
BVI 100 Mac Address 0000.0ceb.c0f8
*Feb  8 18:39:16.398: rpmxf_tbridge_smf_update: BVI 100 ICM programming
*Feb  8 18:39:16.398: rpmxf_tbridge_smf_update: Successfully updated SMF entry for bvi 100
*Feb  8 18:39:16.886: %SYS-5-CONFIG_I: Configured from console by vty0
(CROI_MASTER_000A004B)
*Feb  8 18:39:18.394: %LINK-3-UPDOWN: Interface BVI100, changed state to up
*Feb  8 18:39:19.394: %LINEPROTO-5-UPDOWN: Line protocol on Interface BVI100, changed state
 to up
```

Related Commands

Command	Description
show pxf cpu statistics	Displays PXF CPU statistics for a configured router.
show pxf cpu subblock	Displays PXF CPU subblocks for a bridged subinterface.
show pxf cpu tbridge	Displays PXF CPU statistics for transparent bridging.
show pxf statistics	Displays chassis-wide, summary PXF statistics.