
debug serial interface

Use the **debug serial interface EXEC** command to display information on a serial connection failure. The **no** form of this command disables debugging output.

[no] debug serial interface

Usage Guidelines

If the **show interface serial** command shows that the line and protocol are down, you can use the **debug serial interface** command to isolate a timing problem as the cause of a connection failure. If the keepalive values in the mineseq, yourseen, and myseen fields are not incrementing in each subsequent line of output, there is a timing or line problem at one end of the connection.

Note While the **debug serial interface** command typically does not generate a lot of output, nevertheless use it cautiously during production hours. When SMDS is enabled, for example, it can generate considerable output.

The output of the **debug serial interface** command can vary, depending on the type of WAN configured for an interface: Frame Relay, HDLC, HSSI, SMDS, or X.25. The output also can vary depending on the type of encapsulation configured for that interface. The hardware platform also can affect **debug serial interface** output.

The following sections show sample **debug serial interface** displays for various configurations and describe the possible output the command can generate for these configurations.

Debug Serial Interface for Frame Relay Encapsulation

The following message is displayed if the encapsulation for the interface is Frame Relay (or HDLC) and the router attempts to send a packet containing an unknown packet type:

```
Illegal serial link type code xxx
```

Debug Serial Interface for HDLC

Figure 2-193 shows sample **debug serial interface** output for an HDLC connection when keepalives are enabled.

In Figure 2-193, the **debug serial interface** display shows that the remote router is not receiving all the keepalives the router is sending. When the difference in the values in the myseq and mineseq fields exceeds three, the line goes down and the interface is reset.

Figure 2-193 Sample Debug Serial Interface Output for HDLC

```

router# debug serial interface

Serial1: HDLC myseq 636119, mineseen 636119, yourseen 515032, line up
Serial1: HDLC myseq 636120, mineseen 636120, yourseen 515033, line up
Serial1: HDLC myseq 636121, mineseen 636121, yourseen 515034, line up
Serial1: HDLC myseq 636122, mineseen 636122, yourseen 515035, line up
Serial1: HDLC myseq 636123, mineseen 636123, yourseen 515036, line up
Serial1: HDLC myseq 636124, mineseen 636124, yourseen 515037, line up
Serial1: HDLC myseq 636125, mineseen 636125, yourseen 515038, line up
Serial1: HDLC myseq 636126, mineseen 636126, yourseen 515039, line up

1 missed keepalive Serial1: HDLC myseq 636127, mineseen 636127, yourseen 515040, line up
Serial1: HDLC myseq 636128, mineseen 636127, yourseen 515041, line up
Serial1: HDLC myseq 636129, mineseen 636129, yourseen 515042, line up

3 missed keepalives; line goes down and interface is reset
Serial1: HDLC myseq 636130, mineseen 636130, yourseen 515043, line up
Serial1: HDLC myseq 636131, mineseen 636130, yourseen 515044, line up
Serial1: HDLC myseq 636132, mineseen 636130, yourseen 515045, line up
Serial1: HDLC myseq 636133, mineseen 636130, yourseen 515046, line down
Serial1: HDLC myseq 636127, mineseen 636127, yourseen 515040, line up
Serial1: HDLC myseq 636128, mineseen 636127, yourseen 515041, line up
Serial1: HDLC myseq 636129, mineseen 636129, yourseen 515042, line up
    
```

Table 2-97 describes significant fields shown in Figure 2-193.

Table 2-97 Debug Serial Interface Field Descriptions for HDLC

| Field | Description |
|-----------------|--|
| Serial1 | Interface through which the serial connection is taking place. |
| HDLC | The serial connection is an HDLC connection. |
| myseq 636119 | The myseq counter increases by one each time the router sends a keepalive packet to the remote router. |
| mineseen 636119 | The value of the mineseen counter reflects the last myseq sequence number the remote router has acknowledged receiving from the router. The remote router stores this value in its yourseen counter and sends that value in a keepalive packet to the router. |
| yourseen 515032 | The yourseen counter reflects the value of the myseq sequence number the router has received in a keepalive packet from the remote router. |
| line up | The connection between the routers is maintained. Value changes to “line down” if the values of the myseq and myseen fields in a keepalive packet differ by more than three. Value returns to “line up” when the interface is reset. If the line is in loopback mode, (“looped”) appears after this field. |

Table 2-98 describes additional error messages that the **debug serial interface** command can generate for HDLC.

Table 2-98 Debug Serial Interface Error Messages for HDLC

| Field | Description |
|---|---|
| Illegal serial link type code <i>xxx</i> , PC = <i>0xnnnnnn</i> | This message is displayed if the router attempts to send a packet containing an unknown packet type. |
| Illegal HDLC serial type code <i>xxx</i> , PC = <i>0xnnnn</i> | This message is displayed if an unknown packet type is received. |
| Serial 0: attempting to restart | This message is displayed periodically if the interface is down. The hardware is then reset to hopefully correct the problem. |
| Serial 0: Received bridge packet sent to <i>nnnnnnnnnn</i> | This message is displayed if a bridge packet is received over a serial interface configured for HDLC, and bridging is not configured on that interface. |

Debug Serial Interface for HSSI

On an HSSI interface, the **debug serial interface** command can generate the following additional error message:

```
HSSI0: Reset from 0xnnnnnnnn
```

This message indicates that the HSSI hardware has been reset. The *0xnnnnnnnn* variable is the address of the routine requesting that the hardware be reset; this value is useful only to development engineers.

Debug Serial Interface for ISDN Basic Rate

Table 2-99 describes error messages that the **debug serial interface** command can generate for ISDN Basic Rate.

Table 2-99 Debug Serial Interface Error Messages for ISDN Basic Rate

| Message | Description |
|---|---|
| BRI: D-chan collision | A collision on the ISDN D-channel has occurred; the software will retry transmission. |
| Received SID Loss of Frame Alignment int. | The ISDN hardware has lost frame alignment. This usually indicates a problem with the ISDN network. |
| Unexpected IMP int: ipr = <i>0xnn</i> | The ISDN hardware received an unexpected interrupt. The <i>0xnn</i> variable indicates the value returned by the interrupt register. |
| BRI(d): RX Frame Length Violation. Length= <i>n</i> BRI(d): RX Nonoctet Aligned Frame BRI(d): RX Abort Sequence BRI(d): RX CRC Error BRI(d): RX Overrun Error BRI(d): RX Carrier Detect Lost | Any of these messages can be displayed when a receive error occurs on one of the ISDN channels. The (d) indicates which channel it is on. These messages can indicate a problem with the ISDN network connection. |
| BRI0: Reset from <i>0xnnnnnnnn</i> | The BRI hardware has been reset. The <i>0xnnnnnnnn</i> variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers. |

Table 2-99 Debug Serial Interface Error Messages for ISDN Basic Rate (Continued)

| Message | Description |
|--|--|
| BRI(d): Bad state in SCMs scm1= <i>x</i> scm2= <i>x</i> scm3= <i>x</i> BRI(d): Bad state in SCOns scon1= <i>x</i> scon2 = <i>x</i> scon3= <i>x</i> BRI(d): Bad state ub SCR; SCR= <i>x</i> | Any of these messages can be displayed if the ISDN hardware is not in the proper state. The hardware is then reset. If the message is displayed constantly, it usually indicates a hardware problem. |
| BRI(d): Illegal packet encapsulation= <i>n</i> | This message is displayed if a packet is received, but the encapsulation used for the packet is not recognized. It can indicate that the interface is misconfigured. |

Debug Serial Interface for an MK5025 Device

Table 2-100 describes the additional error messages that the **debug serial interface** command can generate for an MK5025 device.

Table 2-100 Debug Serial Interface Err or Messages for an MK5025 Device

| Message | Description |
|--|---|
| MK5(d): Reset from 0x <i>nnnnnnnn</i> | This message indicates that the hardware has been reset. The 0x <i>nnnnnnnn</i> variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers. |
| MK5(d): Illegal packet encapsulation= <i>n</i> | This message is displayed if a packet is received, but the encapsulation used for the packet is not recognized. Possibly an indication that the interface is misconfigured. |
| MK5(d): No packet available for packet realignment | This message is displayed in cases where the serial driver attempted to get a buffer (memory) and was unable to do so. |
| MK5(d): Bad state in CSR0=(<i>x</i>) | This message is displayed if the hardware is not in the proper state. The hardware is then reset. If this message is displayed constantly, it usually indicates a hardware problem. |
| MK5(d): New serial state= <i>n</i> | This message is displayed to indicate that the hardware has interrupted the software. It displays the state that the hardware is reporting. |
| MK5(d): DCD is down. MK5(d): DCD is up. | If the interrupt indicates that the state of carrier has changed, one of these messages is displayed to indicate the current state of DCD. |

Debug Serial Interface for SMDS Encapsulation

When encapsulation is set to SMDS, **debug serial interface** displays SMDS packets that are sent and received, as well as any error messages resulting from SMDS packet transmission.

The error messages that the **debug serial interface** command can generate for SMDS follow.

The following message indicates that a new protocol requested SMDS to encapsulate the data for transmission. SMDS is not yet able to encapsulate the protocol.

```
SMDS: Error on Serial 0, encapsulation bad protocol = x
```

The following message indicates that SMDS was asked to encapsulate a packet, but no corresponding destination E.164 SMDS address was found in any of the static SMDS tables or in the ARP tables:

```
SMDS send: Error in encapsulation, no hardware address, type = x
```

The following message indicates that a protocol such as CLNS or IP has been enabled on an SMDS interface, but the corresponding multicast addresses have not been configured. The *n* variable displays the link type for which encapsulation was requested. This value is only significant to Cisco as an internal protocol type value.

```
SMDS: Send, Error in encapsulation, type=n
```

The following messages can occur when a corrupted packet is received on an SMDS interface. The router expected *x*, but received *y*.

```
SMDS: Invalid packet, Reserved NOT ZERO, x y
SMDS: Invalid packet, TAG mismatch x y
SMDS: Invalid packet, Bad TRAILER length x y
```

The following messages can indicate an invalid length for an SMDS packet:

```
SMDS: Invalid packet, Bad BA length x
SMDS: Invalid packet, Bad header extension length x
SMDS: Invalid packet, Bad header extension type x
SMDS: Invalid packet, Bad header extension value x
```

The following messages are displayed when the **debug serial interface** command is enabled:

```
Interface Serial 0 Sending SMDS L3 packet:
SMDS: dgsizex type:0xn src:y dst:z
```

If the **debug serial interface** command is enabled, the following message can be displayed when a packet is received on an SMDS interface, but the destination SMDS address does not match any on that interface:

```
SMDS: Packet n, not addressed to us
```


debug service-module

Use the **debug service-module EXEC** command to display debugging information that monitors the detection and clearing of network alarms on the integrated channel service unit/data service unit (CSU/DSU) modules installed in a Cisco 2524 or Cisco 2525 router. The **no** form of this command disables debugging output.

[no] debug service-module

Usage Guidelines

Use this command to enable and disable debug logging for the serial 0 and serial 1 interfaces when an integrated CSU/DSU is present. This command enables debugging on all interfaces.

Network alarm status can also be viewed through the use of the **show service-module** command.

Note The debug output varies depending on the type of service module installed in the router.

Example

Figure 2-195 shows sample **debug service-module** output.

Figure 2-195 Sample Debug Service-Module Output

```
router# debug service-module
Service module debugging is on
SERVICE_MODULE(1): loss of signal ended after duration 00:05:36
SERVICE_MODULE(1): oos/oof ended after duration 01:05:14
SERVICE_MODULE(0): Unit has no clock
SERVICE_MODULE(0): detects loss of signal
SERVICE_MODULE(0): loss of signal ended after duration 00:00:33
```

debug smrp all

Use the **debug smrp all** EXEC command to display information about Simple Multicast Routing Protocol (SMRP) activity. The **no** form of this command disables debugging output.

[no] debug smrp all

Usage Guidelines

Because the **debug smrp all** command displays all SMRP debugging output, it is processor intensive and should not be enabled when memory is scarce or in very high traffic situations.

For general debugging, use the **debug smrp all** command and turn off excessive transactions with the **no debug smrp transaction** command. This combination of commands will display various state changes and events without displaying every transaction packet. For debugging a specific feature such as a routing problem, use the **debug smrp route** and **debug smrp transaction** commands to see if packets are sent and received and which specific routes are affected. The **show smrp traffic** command is highly recommended as a troubleshooting method because it displays the SMRP counters.

For examples of the type of output you may see, refer to each of the commands listed in the “Related Commands” section.

Related Commands

debug smrp group
debug smrp mcache
debug smrp neighbor
debug smrp port
debug smrp route
debug smrp transaction

debug smrp group

Use the **debug smrp group** EXEC command to display information about SMRP group activity. The **no** form of this command disables debugging output.

[no] debug smrp group

Usage Guidelines

The **debug smrp group** command displays information when a group is created or deleted and when a forwarding entry for a group is created, changed, or deleted.

For more information, refer to the **show smrp** command described in the *Network Protocols Command Reference, Part 2*

Sample Display

Figure 2-196 shows sample **debug smrp group** output of a port being created and deleted on group AT 20.34. (AT signifies that this is an AppleTalk network group.)

Figure 2-196 Sample Debug SMRP Group Output

```
router# debug smrp group

SMRP: Group AT 20.34, created on port 20.1 by 20.2
SMRP: Group AT 20.34, deleted on port 20.1
```

Table 2-101 lists the messages that may be generated with the **debug smrp group** command concerning the forwarding table.

Table 2-101 Debug SMRP Group Message Descriptions

| Messages | Descriptions |
|---|---|
| Group <i>address</i> , deleted on port <i>address</i> | Group entry was deleted from the group table for the specified port. |
| Group <i>address</i> , forward state changed from <i>state</i> to <i>state</i> | Group's state changed. Possible states are join, forward, and leave. See the show smrp command for more information. |
| Group <i>address</i> , deleted forward entry | Group was deleted from the forwarding table. |
| Group <i>address</i> , created on port <i>address</i> by <i>address</i> | Group entry was created in the table for the specified port. |
| Group <i>address</i> , added by <i>address</i> to the group | A secondary router has added this group to its group table. |
| Group <i>address</i> , discard join request from <i>address</i> , not responsible | Discard Join Group request if the router is not the primary router on the local connected network or if it is not the port parent of the route. |
| Group <i>address</i> , join request from <i>address</i> | Request to join the group was received. |
| Group <i>address</i> , forward is found | Forward entry for the group was found in the forwarding table. |
| Group <i>address</i> , forward state is already joining, ignored | Request to join the group is in progress, so the second request was discarded. |

Table 2-101 Debug SMRP Group Message Descriptions (Continued)

| Messages | Descriptions |
|--|--|
| Group <i>address</i> , no forward found | Forward entry for the group was not found in the forwarding table. |
| Group <i>address</i> , join request discarded, fw discarded, fwd parent port not operational | Request to join the group was discarded because the parent port is not available. |
| Group <i>address</i> , created forward entry - parent <i>address</i> child <i>address</i> | Forward entry was created in the forwarding table for the parent and child address. |
| Group <i>address</i> , creator no longer up on <i>address</i> | Group creator has not been heard from for a specified time and is deemed no longer available. See the show smrp globals command for information on how often creators are polled. |
| Group <i>address</i> , pruning duplicate path on <i>address</i> | Duplicate path was removed. If we are forwarding and we are a child port, and our port parent address is not pointing to our own port address, we are in a duplicate path. |
| Group <i>address</i> , member no longer up on <i>address</i> | Group member has not been heard from for a specified time and is deemed no longer available. See the show smrp globals command for information on how often members are polled. |
| Group <i>address</i> , no more child ports in forward entry | Forward entry for group no longer has any child ports. As a result, the forward entry is no longer necessary. |

Related Command

debug smrp all

debug smrp mcache

Use the **debug smrp mcache** EXEC command to display information about SMRP multicast fast-switching cache entries. The **no** form of this command disables debugging output.

[no] debug smrp mcache

Usage Guidelines

Use the **show smrp mcache** command (described in the *Network Protocols Command Reference, Part 2*) to display the entries in the SMRP multicast cache, and use the **debug smrp mcache** command to see whether the cache is being populated and invalidated.

Sample Display

Figure 2-197 shows sample **debug smrp mcache** output. In this example, the cache is created and populated for group AT 11.124. (AT signifies that this is an AppleTalk network group.)

Figure 2-197 Sample Debug SMRP Multicast Route Cache Output

```
router# debug smrp mcache

SMRP: Cache created
SMRP: Cache populated for group AT 11.124
      mac - 090007400b7c00000c1740d9
      net - 001fef7500000014ff020a0a0a
SMRP: Forward cache entry created for group AT 11.124
SMRP: Forward cache entry validated for group AT 11.124
SMRP: Forward cache entry invalidated for group AT 11.124
SMRP: Forward cache entry deleted for group AT 11.124
```

Table 2-102 lists all the messages that can be generated with the **debug smrp mcache** command concerning the multicast cache.

Table 2-102 Debug SMRP Mcache Message Descriptions

| Messages | Descriptions |
|--|--|
| Cache populated for group <i>address</i> | SMRP packet was received on a parent port that has fast switching enabled. As a result, the cache was created and the MAC and network headers were stored for all child ports that have fast switching enabled. Use the show smrp port appletalk command with the optional interface type and number to display the switching path. |
| Cache memory allocated | Memory was allocated for the multicast cache. |
| Forward cache entry created/deleted for group <i>address</i> | Forward cache entry for the group was added to or deleted from the cache. |
| Forward cache entry validated for group <i>address</i> | Forward cache entry is validated and is now ready for fast switching. |
| Forward cache entry invalidated for group <i>address</i> | Cache entry is invalidated because some change (such as port was shut down) occurred to one of the ports. |

Related Command

debug smrp all

debug smrp neighbor

Use the **debug smrp neighbor** EXEC command to display information about SMRP neighbor activity. The **no** form of this command disables debugging output.

[no] debug smrp neighbor

Usage Guidelines

The **debug smrp neighbor** command displays information when a neighbor operating state changes. A neighbor is an adjacent router. For more information, refer to the **show smrp neighbor** command described in the *Network Protocols Command Reference, Part 2*.

Sample Display

Figure 2-198 shows sample **debug smrp neighbor** output. In this example, the neighbor on port 30.02 has changed state from normal operation to secondary operation.

Figure 2-198 Sample Debug SMRP Neighbor Output

```
router# debug smrp neighbor
SMRP: Neighbor 30.2, state changed from "normal op" to "secondary op"
```

Table 2-103 lists all the messages that can be generated with the **debug smrp neighbor** command concerning the neighbor table.

Table 2-103 Debug SMRP Neighbor Message Descriptions

| Messages | Descriptions |
|---|---|
| Neighbor <i>address</i> , state changed from <i>state</i> to <i>state</i> | Neighbor's state changed. Possible states are primary operation, secondary operation, normal operation, primary negotiation, secondary negotiation, and down. See the show smrp neighbor command for more information. |
| Neighbor <i>address</i> , neighbor added/deleted | Neighbor was added to or removed from the neighbor table. |
| SMRP neighbor up/down | Neighbor is available for service or unavailable. |
| Neighbor <i>address</i> , no longer up | Neighbor is unavailable because it has not been heard from for a specified duration. See the show smrp globals command for information on neighbor timers. |

Related Command

debug smrp all

debug smrp port

Use the **debug smrp port** EXEC command to display information about SMRP port activity. The **no** form of this command disables debugging output.

[no] debug smrp port

Usage Guidelines

The **debug smrp port** command displays information when a port operating state changes.

For more information, refer to the **show smrp port** command described in the *Network Protocols Command Reference, Part 2*.

Sample Display

Figure 2-199 shows sample **debug smrp port** output. In this example, port 30.1 has changed state from secondary negative to secondary operation to primary negative.

Figure 2-199 Sample Debug SMRP Port Output

```
router# debug smrp port
SMRP: Port 30.1, state changed from "secondary neg" to "secondary op"
SMRP: Port 30.1, secondary router changed from 0.0 to 30.1
SMRP: Port 30.1, state changed from "secondary op" to "primary neg"
```

Table 2-104 lists all the messages that can be generated with the **debug smrp port** command concerning the port table.

Table 2-104 Debug SMRP Port Message Descriptions

| Messages | Descriptions |
|--|---|
| Port <i>address</i> , port created/deleted | Port entry was added to or removed from the port table. |
| Port <i>address</i> , line protocol changed to <i>state</i> | Line protocol for the port is up or down. |
| Port <i>address</i> , state changed from <i>state</i> to <i>state</i> | Port's state changed. Possible states are primary operation, secondary operation, normal operation, primary negotiation, secondary negotiation, and down. See the show smrp port command for more information. |
| Port <i>address</i> , primary/secondary router changed from <i>address</i> to <i>address</i> | Primary or secondary router's port address changed. |

Related Command

debug smrp all

debug smrp route

Use the **debug smrp route** EXEC command to display information about SMRP routing activity. The **no** form of this command disables debugging output.

[no] debug smrp route

Usage Guidelines

For more information, refer to the **show smrp route** command described in the *Network Protocols Command Reference, Part 2*.

Sample Display

Figure 2-200 shows sample **debug smrp route** output. In this example, poison notification is received from port 30.2. Poison notification is the receipt of a poisoned route on a nonparent port.

Figure 2-200 Sample Debug SMRP Route Output

```
router# debug smrp route
SMRP: Route AT 20-20, poison notification from 30.2
SMRP: Route AT 30-30, poison notification from 30.2
```

Table 2-105 lists all the messages that can be generated with the **debug smrp route** command concerning the routing table. In Table 2-105, the term *route* does not refer to an address but rather it is a network range.

Table 2-105 Debug SMRP Route Message Descriptions

| Messages | Descriptions |
|--|---|
| Route <i>address</i> , deleted/created as local network | Route entry was removed from or added to the routing table. |
| Route <i>address</i> , from <i>address</i> has invalid distance value | Route entry from the specified address has an incorrect distance value and was ignored. |
| Route <i>address</i> , unknown route poisoned by <i>address</i> ignored | Route entry received from the specified address is bad and was ignored. |
| Route <i>address</i> , created via <i>address</i> - hop <i>number</i> tunnel <i>number</i> | New route entry added to the routing table with the specified number of hops and tunnels. |
| Route <i>address</i> , from <i>address</i> - overlaps existing route | Route entry received from the specified address overlaps an existing route and was ignored. |
| Route <i>address</i> , poisoned by <i>address</i> | Route entry has been poisoned by neighbor. Poisoned routes have distance of 255. |
| Route <i>address</i> , poison notification from <i>address</i> | A poison notification is a poisoned route that is received from a non-parent port. |
| Route <i>address</i> , worsened by parent <i>address</i> | The distance to the route has worsened (become higher), received from the parent neighbor. |
| Route <i>address</i> , improved via <i>address</i> - <i>number</i> -> <i>number</i> hop, <i>number</i> -> <i>number</i> tunnel | The distance to the route has improved (become lower), received from a neighbor. |

Table 2-105 Debug SMRP Route Message Descriptions (Continued)

| Messages | Descriptions |
|--|---|
| Route <i>address</i> , switched to <i>address</i> - higher address than <i>address</i> | A tie condition exists, and because this router had the highest network address, it was used to forward the packet. |
| Route <i>address</i> , parent port changed <i>address</i> -> <i>address</i> | Parent port address change occurred. The parent port address of a physical network segment determines which router should handle Join Group and Leave Group requests. |
| SMRP bad distance vector | Packet has an invalid distance vector and was ignored. |
| Route <i>address</i> , has been poisoned | Route has been poisoned. Poisoned routes are purged from the routing table after a specified time. See the show smrp globals command for more information. |

Related Command**debug smrp all**

debug smrp transaction

Use the **debug smrp transaction** EXEC command to display information about SMRP transactions. The **no** form of this command disables debugging output.

[no] debug smrp transaction

Sample Display

Figure 2-201 shows sample **debug smrp transaction** output. In this example, a secondary node request is sent out to all routers on port 30.1.

Figure 2-201 Sample Debug SMRP Transaction Output

```
router# debug smrp transaction
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
```

Table 2-106 lists all the messages that can be generated with the **debug smrp route** command.

Table 2-106 Debug SMRP Transaction Message Descriptions

| Messages | Descriptions |
|---|---|
| Transaction for port <i>address</i> , <i>packet-type command-type</i> (<i>grp/sec number</i>) sent to/received from <i>address</i> | Port message concerning a packet or command was sent to or received from the specified address. |
| Transaction for group <i>address</i> on port <i>address</i> , (<i>seq number</i>) sent to/received from <i>address</i> | Group message for a specified port was sent to or received from the specified address. |
| Unrecognized transaction for port <i>address</i> | An unrecognized message was received and ignored by the port. |
| Discarded incomplete request | An incomplete message was received and ignored. |
| Response in wrong state in HandleRequest | A message was received with the wrong state and was ignored. |
| SMRP bad packet type | An SMRP packet was received with a bad packet type and was ignored. |
| Packet discarded, Bad Port ID | A packet was received with a bad port ID and was ignored. |
| Packet discarded, Check Packet failed | A packet was received with a failed check packet and was ignored. |

Related Command

debug smrp all

debug snmp packet

To display information about every SNMP packet sent or received by the router, use the **debug snmp packet** EXEC command. The **no** form of this command disables debugging output.

[no] debug snmp packet

Sample Display

Figure 2-202 shows sample output from the **debug snmp packet** command. In this example, the router receives a get-next request from the host at 172.16.63.17 and responds with the requested information.

Figure 2-202 Sample Debug SNMP Packet Output

```
Router# debug snmp packet

SNMP: Packet received via UDP from 172.16.63.17 on Ethernet0
SNMP: Get-next request, reqid 23584, errstat 0, erridx 0
  sysUpTime = NULL TYPE/VALUE
  system.1 = NULL TYPE/VALUE
  system.6 = NULL TYPE/VALUE
SNMP: Response, reqid 23584, errstat 0, erridx 0
  sysUpTime.0 = 2217027
  system.1.0 = Cisco Internetwork Operating System Software
  system.6.0 =
SNMP: Packet sent via UDP to 172.16.63.17
```

Based on the kind of packet sent or received, the output may vary. For get-bulk requests, a line similar to the following is displayed:

```
SNMP: Get-bulk request, reqid 23584, nonrptr 10, maxreps 20
```

For traps, a line similar to the following is displayed:

```
SNMP: V1 Trap, ent 1.3.6.1.4.1.9.1.13, gentrap 3, spectrap 0
```

Table 2-107 describes significant fields in these displays.

Table 2-107 Debug SNMP Packet Field Descriptions

| Field | Description |
|------------------|---|
| Get-next request | <p>Indicates what type of SNMP PDU the packet is. Possible types are:</p> <ul style="list-style-type: none"> • Get request • Get-next request • Response • Set request • V1 Trap • Get-bulk request • Inform request • V2 Trap <p>Depending on the type of PDU, the rest of this line displays different fields. The indented lines following this line list the MIB object names and corresponding values.</p> |

Table 2-107 Debug SNMP Packet Field Descriptions (Continued)

| Field | Description |
|--------------|---|
| reqid | Request identification number. This number is used by the SNMP manager to match responses with requests. |
| errstat | Error status. All PDU types other than response will have an errstat of 0. If the agent encounters an error while processing the request, it will set errstat in the response PDU to indicate the type of error. |
| erridx | Error index. This value will always be 0 in all PDUs other than responses. If the agent encounters an error, the erridx will be set to indicate which varbind in the request caused the error. For example, if the agent had an error on the 2nd varbind in the request PDU, the response PDU will have an erridx equal to 2. |
| nonrptr | Non-repeater value. This value and the maximum repetition value are used to determine how many varbinds are returned. Refer to RFC 1905 for details. |
| maxreps | Maximum repetition value. This value and the non-repeater value are used to determine how many varbinds are returned. Refer to RFC 1905 for details. |
| ent | Enterprise object identifier. Refer to RFC 1215 for details. |
| gentrap | Generic trap value. Refer to RFC 1215 for details. |
| spectrap | Specific trap value. Refer to RFC 1215 for details. |

debug sntp adjust

Use the **debug sntp adjust** EXEC command to display information about SNTP clock adjustments. The **no** form of this command disables debugging output.

[no] debug sntp adjust

Sample Display

Figure 2-203 shows sample **debug sntp adjust** command output when an offset to the time reported by the configured NTP server is calculated. The offset indicates the difference between the router time and the actual time (as kept by the server) and is displayed in milliseconds. The clock time is then successfully changed to the accurate time by adding the offset to the current router time.

Figure 2-203 Sample Debug SNTP Adjust Output—Small Offset from Configured Server

```
Router# debug sntp adjust

Delay calculated, offset 3.48
Clock slewed.
```

Figure 2-204 show sample **debug sntp adjust** command output when an offset to the time reported by a broadcast server is calculated. Since the packet is a broadcast packet, no transmission delay can be calculated. However, in this case, the offset is too large, so the clock is reset to the correct time.

Figure 2-204 Sample Debug SNTP Adjust Output—Large Offset from Broadcast Server

```
Router# debug sntp adjust

No delay calculated, offset 11.18
Clock stepped.
```

debug sntp packets

Use the **debug sntp packets** EXEC command to display information about SNTP packets sent and received. The **no** form of this command disables debugging output.

[no] debug sntp packets

Sample Displays

Figure 2-205 show sample **debug sntp packets** command output when a message is received.

Figure 2-205 Sample Debug SNTP Packets Output When a Packet Is Received

```
Router# debug sntp packets

Received SNTP packet from 172.16.186.66, length 48
 leap 0, mode 1, version 3, stratum 4, ppoll 1024
 rtdel 00002B00, rtdsp 00003F18, refid AC101801 (172.16.24.1)
 ref B7237786.ABF9CDE5 (23:28:06.671 UTC Tue May 13 1997)
 org 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
 rec 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
 xmt B7237B5C.A7DE94F2 (23:44:28.655 UTC Tue May 13 1997)
 inp AF3BD529.810B66BC (00:19:53.504 UTC Mon Mar 1 1993)
```

Figure 2-206 show sample **debug sntp packets** command output when a message is sent.

Figure 2-206 Sample Debug SNTP Packets Output When a Packet Is Sent

```
Router# debug sntp packets

Sending SNTP packet to 172.16.25.1
 xmt AF3BD455.FBBE3E64 (00:16:21.983 UTC Mon Mar 1 1993)
```

Table 2-108 describes the significant fields shown in these displays.

Table 2-108 Debug SNTP Packets Field Descriptions

| Field | Description |
|---------|--|
| length | Length of the SNTP packet. |
| leap | Indicates if a leap second will be added or subtracted. |
| mode | Indicates the mode of the router relative to the server sending the packet. |
| version | SNTP version number of the packet. |
| stratum | Stratum of the server. |
| ppoll | Peer polling interval. |
| rtdel | Total delay along the path to the root clock. |
| rtdsp | Dispersion of the root path. |
| refid | Address of the server which the router is currently using for synchronization. |
| ref | Reference timestamp. |

Table 2-108 Debug SNTP Packets Field Descriptions (Continued)

| Field | Description |
|--------------|---|
| org | Originate timestamp. This value indicates the time the request was sent by the router. |
| rec | Receive timestamp. This value indicates the time the request was received by the SNTP server. |
| xmt | Transmit timestamp. This value indicates the time the reply was sent by the SNTP server. |
| inp | Destination timestamp. This value indicate the time the reply was received by the router. |

debug sntp select

Use the **debug sntp select** EXEC command to display information about SNTP server selection. The **no** form of this command disables debugging output.

[no] debug sntp select

Sample Display

Figure 2-207 show sample **debug sntp select** command output. In this example, the router will synchronize its time to server at 172.16.186.66.

Figure 2-207 Sample Debug SNTP Packets Output

```
Router# debug sntp packets  
  
SNTP: Selected 172.16.186.66
```

debug source bridge

Use the **debug source bridge** EXEC command to display information about packets and frames transferred across a source-route bridge. The **no** form of this command disables debugging output.

[no] debug source bridge

Sample Display

Figure 2-208 shows sample **debug source bridge** output for peer bridges using TCP as a transport mechanism. The remote source-route bridging (RSRB) network configuration has ring 2 and ring 1 bridged together through remote peer bridges. The remote peer bridges are connected via a serial line and use TCP as the transport mechanism.

Figure 2-208 Sample Debug Source Bridge Output—TCP Environment

```
router# debug source bridge

RSRB: remote explorer to 5/131.108.250.1/1996 srn 2 [C840.0021.0050.0000]
RSRB: Version/Ring XReq sent to peer 5/131.108.250.1/1996
RSRB: Received version reply from 5/131.108.250.1/1996 (version 2)
RSRB: DATA: 5/131.108.250.1/1996 Ring Xchg Rep, trn 2, vrn 5, off 18, len 10
RSRB: added bridge 1, ring 1 for 5/131.108.240.1/1996
RSRB: DATA: 5/131.108.250.1/1996 Explorer trn 2, vrn 5, off 18, len 69
RSRB: DATA: 5/131.108.250.1/1996 Forward trn 2, vrn 5, off 0, len 92
RSRB: DATA: forward Forward srn 2, br 1, vrn 5 to peer 5/131.108.250.1/1996
```

Explanations for individual lines of output in Figure 2-208 follow.

The following line indicates that a remote explorer frame has been sent to IP address 131.108.250.1 and like all RSRB TCP connections, has been assigned port 1996. The bridge belongs to ring group 5. The explorer frame originated from ring number 2. The routing information field (RIF) descriptor has been generated by the local station and indicates that the frame was sent out via bridge 1 onto virtual ring 5.

```
RSRB: remote explorer to 5/131.108.250.1/1996 srn 2 [C840.0021.0050.0000]
```

The following line indicates that a request for remote peer information has been sent to IP address 131.108.250.1, TCP port 1996. The bridge belongs to ring group 5.

```
RSRB: Version/Ring XReq sent to peer 5/131.108.250.1/1996
```

The following line is the response to the version request previously sent. The response is sent from IP address 131.108.250.1, TCP port 1996. The bridge belongs to ring group 5.

```
RSRB: Received version reply from 5/131.108.250.1/1996 (version 2)
```

The following line is the response to the ring request previously sent. The response is sent from IP address 131.108.250.1, TCP port 1996. The target ring number is 2, virtual ring number is 5, the offset is 18, and the length of the frame is 10 bytes.

```
RSRB: DATA: 5/131.108.250.1/1996 Ring Xchg Rep, trn 2, vrn 5, off 0, len 10
```

The following line indicates that bridge 1 and ring 1 were added to the source-bridge table for IP address 131.108.250.1, TCP port 1996.

```
RSRB: added bridge 1, ring 1 for 5/131.108.250.1/1996
```

The following line indicates that a packet containing an explorer frame came across virtual ring 5 from IP address 131.108.250.1, TCP port 1996. The packet is 69 bytes in length. This packet is received after the Ring Exchange information was received and updated on both sides.

```
RSRB: DATA: 5/131.108.250.1/1996 Explorer trn 2, vrn 5, off 18, len 69
```

The following line indicates that a packet containing data came across virtual ring 5 from IP address 131.108.250.1 over TCP port 1996. The packet is being placed on the local target ring 2. The packet is 92 bytes in length.

```
RSRB: DATA: 5/131.108.250.1/1996 Forward trn 2, vrn 5, off 0, len 92
```

The following line indicates that a packet containing data is being forwarded to the peer that has IP 131.108.250.1 address belonging to local ring 2 and bridge 1. The packet is forwarded via virtual ring 5. This packet is sent after the Ring Exchange information was received and updated on both sides.

```
RSRB: DATA: forward Forward srn 2, br 1, vrn 5 to peer 5/131.108.250.1/1996
```

Figure 2-209 shows sample **debug source bridge** output for peer bridges using direct encapsulation as a transport mechanism. The RSRB network configuration has ring 1 and ring 2 bridged together through peer bridges. The peer bridges are connected via a serial line and use TCP as the transport mechanism.

Figure 2-209 Sample Debug Source Bridge Output—Direct Encapsulation Environment

```
router# debug source bridge

RSRB: remote explorer to 5/Serial1 srn 1 [C840.0011.0050.0000]
RSRB: Version/Ring XReq sent to peer 5/Serial1
RSRB: Received version reply from 5/Serial1 (version 2)
RSRB: IFin: 5/Serial1 Ring Xchg, Rep trn 0, vrn 5, off 0, len 10
RSRB: added bridge 1, ring 1 for 5/Serial1
```

Explanations for individual lines of output in Figure 2-209 follow.

The following line indicates that a remote explorer frame was sent to remote peer Serial1, which belongs to ring group 5. The explorer frame originated from ring number 1. The routing information field (RIF) descriptor 0011.0050 was generated by the local station and indicates that the frame was sent out via bridge 1 onto virtual ring 5.

```
RSRB: remote explorer to 5/Serial1 srn 1 [C840.0011.0050.0000]
```

The following line indicates that a request for remote peer information was sent to Serial1. The bridge belongs to ring group 5.

```
RSRB: Version/Ring XReq sent to peer 5/Serial1
```

The following line is the response to the version request previously sent. The response is sent from Serial 1. The bridge belongs to ring group 5 and the version is 2.

```
RSRB: Received version reply from 5/Serial1 (version 2)
```

The following line is the response to the ring request previously sent. The response is sent from Serial1. The target ring number is 2, virtual ring number is 5, the offset is 0, and the length of the frame is 39 bytes.

```
RSRB: IFin: 5/Serial1 Ring Xchg Rep, trn 2, vrn 5, off 0, len 39
```

The following line indicates that bridge 1 and ring 1 were added to the source-bridge table for Serial1.

```
RSRB: added bridge 1, ring 1 for 5/Serial1
```

debug source error

Use the **debug source error** EXEC command to display source-route bridging errors. The **no** form of this command disables debugging output.

[no] debug source error

Usage Guidelines

The debug source error command displays some output also found in the **debug source bridge** output. Refer to the **debug source bridge** command for other possible output.

Sample Displays

In all of the following examples of **debug source error** command messages, the variable *number* is the Token Ring interface. For example, if the line of output starts with SRB1, the output relates to the Token Ring 1 interface. SRB indicates a source-route bridging message. RSRB indicates a remote source-route bridging message. SRTL B indicates a source-route translational bridging message.

In the following example, a packet of protocol *protocol-type* was dropped:

```
SRBnumber drop: Routed protocol protocol-type
```

In the following example, an Address Resolution Protocol (ARP) packet was dropped. ARP is defined in RFC 826.

```
SRBnumber drop:TYPE_RFC826_ARP
```

In the following example, the current Cisco IOS version does not support Qualified Logical Link Control (QLLC). Reconfigure the router with an image that has the IBM feature set.

```
RSRB: QLLC not supported in version version  
Please reconfigure.
```

In the following example, the packet was dropped because the outgoing interface of the router was down:

```
RSRB IF: outgoing interface not up, dropping packet
```

In the following example, the router received an out-of-sequence IP sequence number in a Fast Sequenced Transport (FST) packet. FST has no recovery for this problem like TCP encapsulation does.

```
RSRB FST: bad sequence number dropping.
```

In the following example, the router was unable to locate the virtual interface:

```
RSRB: couldn't find virtual interface
```

In the following example, the peer router's TCP queue is full. TCPD indicates that this is a TCP debug.

```
RSRB TCPD: tcp queue full for peer
```

In the following example, the router was unable to send data to the *peer* router. A *result* of 1 indicates that the TCP queue is full. A *result* of -1 indicates that the RSRB peer is closed.

```
RSRB TCPD: tcp send failed for peer result
```

In the following example, the Routing Information Identifier was not set in the explorer packet going forward. The packet will not support SRB, so it is dropped.

```
vrforward_explorer - RII not set
```

In the following example, a packet sent to a virtual bridge in the router did not include a routing information field (RIF) to tell the router which route to use:

```
RSRB: no RIF on packet sent to virtual bridge
```

The following example indicates that the RIF did not contain any information or the length field was set to zero:

```
RSRB: RIF length of zero sent to virtual bridge
```

The following message occurs when the local service access point (LSAP) is out of range. The variable *lsap-out* is the value, *type* is the type of RSRB peer, and *state* is the state of the RSRB peer.

```
VRP: rsrb_lsap_out = lsap-out, type = type, state = state
```

In the following message, the router is unable to find another router with which to exchange bridge protocol data units (BPDU's). BPDU's are exchanged to set up the spanning tree and determine the forwarding path.

```
RSRB(span): BPDU's peer not found
```

Related Commands

debug source bridge

debug source event

debug source event

Use the **debug source event EXEC** command to display information on source-route bridging activity. The **no** form of this command disables debugging output.

[no] debug source event

Usage Guidelines

Some of the output from the **debug source bridge** and **debug source error** commands is identical to the output of this command.

Note In order to use the **debug source event** command to display traffic source-routed through an interface, you first must disable fast switching of SRB frames with the **no source bridge route-cache** interface configuration command.

Sample Display

Figure 2-210 shows sample **debug source event** output.

Figure 2-210 Sample Debug Source Event Output

```
router# debug source event

RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
```

Table 2-109 describes significant fields shown in Figure 2-210.

Table 2-109 Debug Source Event Field Descriptions

| Field | Description |
|---------------------|--|
| RSRB0: | Indication that this RIF cache entry is for the Token Ring 0 interface, which has been configured for remote source-route bridging. (SRB1, in contrast, would indicate that this RIF cache entry is for Token Ring 1, configured for source-route bridging.) |
| forward | Forward (normal data) packet, in contrast to a control packet containing proprietary Cisco bridging information. |
| srn 5 | Ring number of the packet's source ring. |
| bn 1 | Bridge number of the bridge this packet traverses. |
| trn 10 | Ring number of the packet's target ring. |
| src: 8110.2222.33c1 | Source address of the route in this RIF cache entry. |
| dst: 1000.5a59.04f9 | Destination address of the route in this RIF cache entry. |

Table 2-109 Debug Source Event Field Descriptions (Continued)

| Field | Description |
|-----------------------|-------------------------------------|
| [0800.3201.00A1.0050] | RIF string in this RIF cache entry. |

Examples of other **debug source event** messages follow.

In the following example messages, *SRBnumber* or *RSRBnumber* denotes a message associated with interface Token Ring *number*. A *number* of 99 denotes the remote side of the network.

```
SRBnumber: no path, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

In the preceding example, a bridgeable packet came in on interface Token Ring *number* but there was nowhere to send it. This is most likely a configuration error. For example, an interface has source bridging turned on, but it is not connected to another source bridging interface or a ring group.

In the following example, a bridgeable packet has been forwarded from Token Ring *number* to the target ring. The two interfaces are directly linked.

```
SRBnumber: direct forward (srn ring bn bridge trn ring)
```

In the following examples, a proxy explorer reply was not generated because there was no way to get to the address from this interface. The packet came from the node with the first *address*.

```
SRBnumber: br dropped proxy XID, address for address, wrong vring (rem)
SRBnumber: br dropped proxy TEST, address for address, wrong vring (rem)
SRBnumber: br dropped proxy XID, address for address, wrong vring (local)
SRBnumber: br dropped proxy TEST, address for address, wrong vring (local)
SRBnumber: br dropped proxy XID, address for address, no path
SRBnumber: br dropped proxy TEST, address for address, no path
```

In the following example, an appropriate proxy explorer reply was generated on behalf of the second *address*. It is sent to the first *address*.

```
SRBnumber: br sent proxy XID, address for address[rif]
SRBnumber: br sent proxy TEST, address for address[rif]
```

The following example indicates that the broadcast bits were not set, or that the routing information indicator on the packet was not set:

```
SRBnumber: illegal explorer, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that the direction bit in the RIF field was set, or that an odd packet length was encountered. Such packets are dropped.

```
SRBnumber: bad explorer control, D set or odd
```

The following example indicates that a spanning explorer was dropped because the spanning option was not configured on the interface:

```
SRBnumber: span dropped, input off, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that a spanning explorer was dropped because it had traversed the ring previously:

```
SRBnumber: span violation, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that an explorer was dropped because the maximum hop count limit was reached on that interface:

```
SRBnumber: max hops reached - hop-cnt, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that the ring exchange request was sent to the indicated peer. This request tells the remote side which rings this node has and asks for a reply indicating which rings that side has.

```
RSRB: sent RingXreq to ring-group/ip-addr
```

The following example indicates that a message was sent to the remote peer. The *label* variable can be AHDR (active header), PHDR (passive header), HDR (normal header), or DATA (data exchange), and *op* can be Forward, Explorer, Ring Xchg, Req, Ring Xchg, Rep, Unknown Ring Group, Unknown Peer, or Unknown Target Ring.

```
RSRB: label: sent op to ring-group/ip-addr
```

The following example indicates that the remote bridge and ring pair were removed from or added to the local ring group table because the remote peer changed:

```
RSRB: removing bn bridge rn ring from ring-group/ip-addr  
RSRB: added bridge bridge, ring ring for ring-group/ip-addr
```

The following example shows miscellaneous remote peer connection establishment messages:

```
RSRB: peer ring-group/ip-addr closed [last state n]  
RSRB: passive open ip-addr(remote port) -> local port  
RSRB: CONN: opening peer ring-group/ip-addr, attempt n  
RSRB: CONN: Remote closed ring-group/ip-addr on open  
RSRB: CONN: peer ring-group/ip-addr open failed, reason[code]
```

The following example shows that an explorer packet was propagated onto the local ring from the remote ring group:

```
RSRBn: sent local explorer, bridge bridge trn ring, [rif]
```

The following messages indicate that the remote source-route bridging code found the packet was in error:

```
RSRBn: ring group ring-group not found  
RSRBn: explorer rif [rif] not long enough
```

The following example indicates that a buffer could not be obtained for a ring exchange packet; this is an internal error.

```
RSRB: couldn't get pak for ringXchg
```

The following example indicates that a ring exchange packet was received that had an incorrect length; this is an internal error.

```
RSRB: XCHG: req/reply badly formed, length pak-length, peer peer-id
```

The following example indicates that a ring entry was removed for the peer; the ring was possibly disconnected from the network, causing the remote router to send an update to all its peers.

```
RSRB: removing bridge bridge ring ring from peer-id ring-type
```

The following example indicates that a ring entry was added for the specified peer; the ring was possibly added to the network, causing the other router to send an update to all its peers.

```
RSRB: added bridge bridge, ring ring for peer-id
```

The following example indicates that no memory was available to add a ring number to the ring group specified; this is an internal error.

```
RSRB: no memory for ring element ring-group
```

The following example indicates that memory was corrupted for a connection block; this is an internal error.

```
RSRB: CONN: corrupt connection block
```

The following example indicates that a connector process started, but that there was no packet to process; this is an internal error.

```
RSRB: CONN: warning, no initial packet, peer: ip-addr peer-pointer
```

The following example indicates that a packet was received with a version number different from the one present on the router:

```
RSRB: IF New version. local=local-version, remote=remote-version,pak-op-code peer-id
```

The following example indicates that a packet with a bad op code was received for a direct encapsulation peer; this is an internal error.

```
RSRB: IFin: bad op op-code (op code string) from peer-id
```

The following example indicates that the virtual ring header will not fit on the packet to be sent to the peer; this is an internal error:

```
RSRB: vrif_sender, hdr won't fit
```

The following example indicates that the specified peer is being opened. The retry count specifies the number of times the opening operation is attempted.

```
RSRB: CONN: opening peer peer-id retry-count
```

The following example indicates that the router, configured for FST encapsulation, received a version reply to the version request packet it had sent previously:

```
RSRB: FST Rcvd version reply from peer-id (version version-number)
```

The following example indicates that the router, configured for FST encapsulation, sent a version request packet to the specified peer:

```
RSRB: FST Version Request. op = opcode, peer-id
```

The following example indicates that the router received a packet with a bad op code from the specified peer; this is an internal error.

```
RSRB: FSTin: bad op opcode (op code string) from peer-id
```

The following example indicates that the TCP connection between the router and the specified peer is being aborted:

```
RSRB: aborting ring-group/peer-id (vrtcpd_abort called)
```

The following example indicates that an attempt to establish a TCP connection to a remote peer timed out:

```
RSRB: CONN: attempt timed out
```

The following example indicates that a packet was dropped because the ring group number in the packet did not correlate with the ring groups configured on the router:

```
RSRBnumber: ring group ring-group not found
```

debug span

Use the **debug span EXEC** command to display information on changes in the spanning-tree topology when debugging a transparent bridge. The **no** form of this command disables debugging output.

[no] debug span

Usage Guidelines

This command is useful for tracking and verifying that the spanning-tree protocol is operating correctly.

Sample Display—IEEE Spanning Tree

Sample **debug span** output for an IEEE BPDU packet follows:

```
ST: Ether4 0000000000000A080002A02D6700000000000A080002A02D6780010000140002000F00
```

Figure 2-211 shows the preceding **debug span** output broken up by fields and labeled to aid documentation.

Figure 2-211 Sample Debug Span Output—IEEE BPDU Packet

```
ST: Ether4 0000 00 00 00 000A 080002A02D67 00000000 000A 080002A02D67 80 01 0000 1400 0200 0F00
      A  B C D E  F          G      H  I          J K L  M  N  O      S2575
```

Table 2-110 describes significant fields shown in Figure 2-211.

Table 2-110 Debug Span Field Descriptions—IEEE BPDU Packet

| Field | Description |
|------------------|---|
| ST: | Indication that this is a spanning tree packet. |
| Ether4 | Interface receiving the packet. |
| (A) 0000 | Indication that this is an IEEE BPDU packet. |
| (B) 00 | Version. |
| (C) 00 | Command mode: 00 indicates config BPDU. 80 indicates the Topology Change Notification (TCN) BPDU. |
| (D) 00 | Topology change acknowledgment: 00 indicates no change. 80 indicates a change notification. |
| (E) 000A | Root priority. |
| (F) 080002A02D67 | Root ID. |
| (G) 00000000 | Root path cost (0 means the sender of this BPDU packet is the root bridge). |
| (H) 000A | Bridge priority. |
| (I) 080002A02D67 | Bridge ID. |
| (J) 80 | Port priority. |
| (K) 01 | Port No. 1. |

Table 2-110 Debug Span Field Descriptions—IEEE BPDU Packet (Continued)

| Field | Description |
|----------|---|
| (L) 0000 | Message age in 256ths of a second (0 seconds, in this case). |
| (M) 1400 | Maximum age in 256ths of a second (20 seconds, in this case). |
| (N) 0200 | Hello time in 256ths of a second (2 seconds, in this case). |
| (O) 0F00 | Forward delay in 256ths of a second (15 seconds, in this case). |

Sample Display—DEC Spanning Tree

Sample **debug span** output for a DEC BPDU packet follows:

```
ST: Ethernet4 E1190100000200000C01A2C90064008000000C0106CE0A01050F1E6A
```

Figure 2-212 shows the preceding **debug span** output broken up by fields and labeled to aid documentation.

Figure 2-212 Sample Debug Span Output

```
E1 19 01 00 0002 00000C01A2C9 0064 0080 00000C0106CE 0A 01 05 0F 1E 6A
A B C D E F G H I J K L M N O S3576
```

Table 2-111 describes significant fields shown in Figure 2-212.

Table 2-111 Debug Span Field Descriptions for a DEC BPDU Packet

| Field | Description |
|------------------|---|
| ST: | Indication that this is a spanning tree packet. |
| Ethernet4 | Interface receiving the packet. |
| (A) E1 | Indication that this is a DEC BPDU packet. |
| (B) 19 | Indication that this is a DEC Hello packet. Possible values are as follows: <ul style="list-style-type: none"> • 0x19—DEC Hello • 0x02—Topology change notification (TCN) |
| (C) 01 | DEC version. |
| (D) 00 | Flag that is a bit field with the following mapping: <ul style="list-style-type: none"> • 1—TCN • 2—TCN acknowledgment • 8—Use short timers |
| (E) 0002 | Root priority. |
| (F) 00000C01A2C9 | Root ID (MAC address). |
| (G) 0064 | Root path cost (translated as 100 in decimal notation). |
| (H) 0080 | Bridge priority. |
| (I) 00000C0106CE | Bridge ID. |
| (J) 0A | Port ID (in contrast to interface number). |
| (K) 01 | Message age (in seconds). |
| (L) 05 | Hello time (in seconds). |

Table 2-111 Debug Span Field Descriptions for a DEC BPDU Packet (Continued)

| Field | Description |
|--------------|-----------------------------|
| (M) 0F | Maximum age (in seconds). |
| (N) 1E | Forward delay (in seconds). |
| (O) 6A | Not applicable. |

debug sse

Use the **debug sse** EXEC command to display information for the silicon switching engine (SSE) processor. The **no** form of this command disables debugging output.

[no] debug sse

Usage Guidelines

By using the **debug sse** command, you can observe statistics and counters maintained by the SSE.

Sample Display

Figure 2-213 shows sample **debug sse** output.

Figure 2-213 Sample Debug SSE Output

```
router# debug sse
SSE: IP number of cache entries changed 273 274
SSE: IP number of cache entries changed 273 274
SSE: bridging enabled
SSE: interface Ethernet0/0 icb 0x30 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/1 icb 0x33 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/2 icb 0x36 addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/3 icb 0x39 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/4 icb 0x3C addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/5 icb 0x3F addr 0x29 status 0x21A040 protos 0x11
SSE: interface Hssi1/0 icb 0x48 addr 0x122 status 0x421E080 protos 0x11
SSE: cache update took 316ms, elapsed 320ms
```

Explanations for representative lines of output in Figure 2-213 follow.

The following line indicates that the SSE cache is being updated due to a change in the IP fast switching cache:

```
SSE: IP number of cache entries changed 273 274
```

The following line indicates that bridging functions were enabled on the SSE:

```
SSE: bridging enabled
```

The following lines indicate that the SSE is now loaded with information about the interfaces:

```
SSE: interface Ethernet0/0 icb 0x30 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/1 icb 0x33 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/2 icb 0x36 addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/3 icb 0x39 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/4 icb 0x3C addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/5 icb 0x3F addr 0x29 status 0x21A040 protos 0x11
SSE: interface Hssi1/0 icb 0x48 addr 0x122 status 0x421E080 protos 0x11
```

The following line indicates that the SSE took 316 ms of processor time to update the SSE cache. The value of 320 ms represents the total time elapsed while the cache updates were performed.

```
SSE: cache update took 316ms, elapsed 320ms
```

debug standby

Use the **debug standby EXEC** command to display Hot Standby Protocol state changes. The **no** form of this command disables debugging output.

[no] debug standby

Usage Guidelines

The **debug standby** command displays Hot Standby Protocol state changes and debugging information regarding transmission and receipt of Hot Standby Protocol packets. Use this command to determine whether hot standby routers recognize one another and take the proper actions.

Sample Display

Figure 2-214 shows sample **debug standby** output.

Figure 2-214 Sample Debug Standby Output

```
router# debug standby

SB: Ethernet0 state Virgin -> Listen
SB: Starting up hot standby process
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB: Ethernet0 state Listen -> Speak
SB:Ethernet0 Hello out 192.168.72.20 Speak pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello out 192.168.72.20 Speak pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello out 192.168.72.20 Speak pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB: Ethernet0 state Speak -> Standby
SB:Ethernet0 Hello out 192.168.72.20 Standby pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello out 192.168.72.20 Standby pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello out 192.168.72.20 Standby pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Active pri 90 hel 3 hol 10 ip 192.168.72.29
SB: Ethernet0 Coup out 192.168.72.20 Standby pri 100 hel 3 hol 10 ip 192.168.72.29
SB: Ethernet0 state Standby -> Active
SB:Ethernet0 Hello out 192.168.72.20 Active pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Speak pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello out 192.168.72.20 Active pri 100 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello in 192.168.72.21 Speak pri 90 hel 3 hol 10 ip 192.168.72.29
SB:Ethernet0 Hello out 192.168.72.20 Active pri 100 hel 3 hol 10 ip 192.168.72.29
```

Table 2-112 describes significant fields shown in Figure 2-214.

Table 2-112 Debug Standby Field Descriptions

| Field | Description |
|-----------|---|
| SB | An abbreviation for “standby.” |
| Ethernet0 | The interface on which a hot standby packet was sent or received. |
| Hello in | Hello packet received from the specified IP address. |

Table 2-112 Debug Standby Field Descriptions (Continued)

| Field | Description |
|-------------------------|---|
| Hello out | Hello packet sent from the specified IP address. |
| pri | Priority advertised in the hello packet. |
| hel | Hello interval advertised in the hello packet. |
| hol | Holddown interval advertised in the hello packet. |
| ip <i>address</i> | Hot standby group IP address advertised in the hello packet. |
| state | Transition from one state to another. |
| Coup out <i>address</i> | Coup packet sent by the router from the specified IP address. |

Explanations for representative lines of output in Figure 2-214 follow.

The following line indicates that the router is initiating the Hot Standby Protocol. The **standby ip** interface configuration command enables hot standby.

```
SB: Starting up hot standby process
```

The following line indicates that a state transition occurred on the interface:

```
SB: Ethernet0 state Listen -> Speak
```

debug stun packet

Use the **debug stun packet** EXEC command to display information on packets traveling through the serial tunnel (STUN) links. Use the **no** form of this command to disable debugging output.

[no] debug stun packet [*group*] [*address*]

Syntax Description

- group* (Optional) Decimal integer assigned to a group. Using this option limits output to packets associated with the specified STUN group.
- address* (Optional) Output is further limited to only those packets containing the specified STUN address. The *address* argument is in the appropriate format for the STUN protocol running for the specified group.

Usage Guidelines

Because using this command is processor intensive, it is best to use it after hours, rather than in a production environment. It is also best to turn this command on by itself, rather than use it in conjunction with other debug commands.

Sample Display

Figure 2-215 shows sample **debug stun packet** output.

Figure 2-215 Sample Debug STUN Packet Output

```

router# debug stun packet
X1 type of packet — STUN sdlc: 0:00:04 Serial3      NDI: (0C2/008) U: SNRM    PF:1
                    STUN sdlc: 0:00:04 Serial3      NDI: (0C2/008) U: SNRM    PF:1
X2 type of packet — STUN sdlc: 0:00:01 Serial3      SDI: (0C2/008) U: UA      PF:1
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:000
X3 type of packet — STUN sdlc: 0:00:00 Serial3      NDI: (0C2/008) I:         PF:1 NR:000 NS:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) I:         PF:1 NR:001 NS:000
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:001
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:001
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:001
                    STUN sdlc: 0:00:00 Serial3      SDI: (0C2/008) S: RR      PF:1 NR:001

```

Explanations for individual lines of output from Figure 2-215 follow.

The following line describes an X1 type of packet:

```
STUN sdlc: 0:00:04 Serial3          NDI: (0C2/008) U: SNRM    PF:1
```

Table 2-113 describes significant fields shown in this line of **debug stun packet** output.

Table 2-113 Debug STUN Packet Field Descriptions

| Field | Description |
|------------|---|
| STUN sdlc: | Indication that the STUN feature is providing the information. |
| 0:00:04 | Time elapsed since receipt of previous packet. |
| Serial3 | Interface type and unit number reporting the event. |
| NDI: | The type of cloud separating the SDLC end nodes. Possible values follow: <ul style="list-style-type: none"> • NDI—Network input • SDI—Serial link |
| 0C2 | SDLC address of the SDLC connection. |
| 008 | A modulo value of 8. |
| U:SNRM | The frame type followed by the command or response type. In this case it is an Unnumbered frame that contains an SNRM (Set Normal Response Mode) command. The possible frame types are as follows: <ul style="list-style-type: none"> • I—Information frame • S—Supervisory frame. The possible commands and responses are: RR (Receive Ready), RNR (Receive Not Ready), and REJ (Reject). • U—Unnumbered frame. The possible commands are: UI (Unnumbered Information), SNRM, DISC/RD (Disconnect/Request Disconnect), SIM/RIM, XID Exchange Identification), TEST. The possible responses are UA (unnumbered acknowledgment), DM (Disconnected Mode), and FRMR (Frame Reject Mode) |
| PF:1 | Poll/Final bit. The possible values are as follows: <ul style="list-style-type: none"> • 0—Off • 1—On |

The following line of output describes an X2 type of packet:

```
STUN sdlc: 0:00:00 Serial3          SDI: (0C2/008) S: RR    PF:1 NR:000
```

All the fields in the previous line of output match those for an X1 type of packet, except the last field, which is additional. NR:000 indicates a receive count of 0; the range for the receive count is 0 to 7.

The following line of output describes an X3 type of packet:

```
STUN sdlc: 0:00:00 Serial3          SDI: (0C2/008) S:I PF:1 NR:000 NS:000
```

All fields in the previous line of output match those for an X2 type of packet, except the last field, which is additional. NS:000 indicates a send count of 0; the range for the send count is 0 to 7.

debug tacacs

Use the **debug tacacs EXEC** command to display information associated with the Terminal Access Controller Access Control System (TACACS). The **no** form of this command disables debugging output.

[no] debug tacacs

Usage Guidelines

TACACS is a distributed security system that secures networks against unauthorized access. Cisco supports TACACS under the Authentication, Authorization, and Accounting (AAA) security system.

Use the **debug aaa authentication** command to get a high-level view of login activity. When TACACS is used on the router, you can use the **debug tacacs** command for more detailed debugging information.

Sample Display

Figure 2-216 shows part of the **debug aaa authentication** command output for a TACACS login attempt that was successful. The information indicates that TACACS+ is the authentication method used.

Figure 2-216 Sample Debug AAA Authentication Output—TACACS Login Attempt

```
14:01:17: AAA/AUTHEN (567936829): Method=TACACS+
14:01:17: TAC+: send AUTHEN/CONT packet
14:01:17: TAC+ (567936829): received authen response status = PASS
14:01:17: AAA/AUTHEN (567936829): status = PASS
```

Figure 2-217 shows part of the **debug tacacs** command output for a TACACS login attempt that was successful as indicated by the status PASS.

Figure 2-217 Sample Debug TACACS Output—Successful Login Attempt

```
14:00:09: TAC+: Opening TCP/IP connection to 192.168.60.15 using source 10.116.0.79
14:00:09: TAC+: Sending TCP/IP packet number 383258052-1 to 192.168.60.15 (AUTHEN/START)
14:00:09: TAC+: Receiving TCP/IP packet number 383258052-2 from 192.168.60.15
14:00:09: TAC+ (383258052): received authen response status = GETUSER
14:00:10: TAC+: send AUTHEN/CONT packet
14:00:10: TAC+: Sending TCP/IP packet number 383258052-3 to 192.168.60.15 (AUTHEN/CONT)
14:00:10: TAC+: Receiving TCP/IP packet number 383258052-4 from 192.168.60.15
14:00:10: TAC+ (383258052): received authen response status = GETPASS
14:00:14: TAC+: send AUTHEN/CONT packet
14:00:14: TAC+: Sending TCP/IP packet number 383258052-5 to 192.168.60.15 (AUTHEN/CONT)
14:00:14: TAC+: Receiving TCP/IP packet number 383258052-6 from 192.168.60.15
14:00:14: TAC+ (383258052): received authen response status = PASS
14:00:14: TAC+: Closing TCP/IP connection to 192.168.60.15
```

Figure 2-218 shows part of the **debug tacacs** command output for a TACACS login attempt that was unsuccessful as indicated by the status FAIL.

Figure 2-218 Sample Debug TACACS Output—Failed Login Attempt

```
13:53:35: TAC+: Opening TCP/IP connection to 192.168.60.15 using source
192.48.0.79
13:53:35: TAC+: Sending TCP/IP packet number 416942312-1 to 192.168.60.15
(AUTHEN/START)
13:53:35: TAC+: Receiving TCP/IP packet number 416942312-2 from 192.168.60.15
13:53:35: TAC+ (416942312): received authen response status = GETUSER
13:53:37: TAC+: send AUTHEN/CONT packet
13:53:37: TAC+: Sending TCP/IP packet number 416942312-3 to 192.168.60.15
(AUTHEN/CONT)
13:53:37: TAC+: Receiving TCP/IP packet number 416942312-4 from 192.168.60.15
13:53:37: TAC+ (416942312): received authen response status = GETPASS
13:53:38: TAC+: send AUTHEN/CONT packet
13:53:38: TAC+: Sending TCP/IP packet number 416942312-5 to 192.168.60.15
(AUTHEN/CONT)
13:53:38: TAC+: Receiving TCP/IP packet number 416942312-6 from 192.168.60.15
13:53:38: TAC+ (416942312): received authen response status = FAIL
13:53:40: TAC+: Closing TCP/IP connection to 192.168.60.15
```

Related Commands

debug aaa accounting

debug aaa authentication

debug tacacs events

Use the **debug tacacs events** EXEC command to display information from the TACACS+ helper process. The **no** form of this command disables debugging output.

[no] debug tacacs events

Usage Guidelines

Use the **debug tacacs events** command only in response to a request from service personnel to collect data when a problem has been reported.



Caution Use the **debug tacacs events** command with caution because it can generate a significant amount of output.

The TACACS protocol is used on routers to assist in managing user accounts. TACACS+ enhances the TACACS functionality by adding security features and cleanly separating out the authentication, authorization, and accounting (AAA) functionality.

Sample Display

Figure 2-219 shows sample **debug tacacs events** output. In this example, the opening and closing of a TCP connection to a TACACS+ server are shown, as well as the bytes read and written over the connection and the connection's TCP status.

Figure 2-219 Sample Debug TACACS Events Output

```
router# debug tacacs events
%LINK-3-UPDOWN: Interface Async2, changed state to up
00:03:16: TAC+: Opening TCP/IP to 192.168.58.104/1049 timeout=15
00:03:16: TAC+: Opened TCP/IP handle 0x48A87C to 192.168.58.104/1049
00:03:16: TAC+: periodic timer started
00:03:16: TAC+: 192.168.58.104 req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (ESTAB)
expire=14 AUTHEN/START/SENDAUTH/CHAP queued
00:03:17: TAC+: 192.168.58.104 ESTAB 3BD868 wrote 46 of 46 bytes
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=12 wanted=12 alloc=12 got=12
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=61 wanted=61 alloc=61 got=49
00:03:22: TAC+: 192.168.58.104 received 61 byte reply for 3BD868
00:03:22: TAC+: req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (CLOSEWAIT) expire=9
AUTHEN/START/SENDAUTH/CHAP processed
00:03:22: TAC+: periodic timer stopped (queue empty)
00:03:22: TAC+: Closing TCP/IP 0x48A87C connection to 192.168.58.104/1049
00:03:22: TAC+: Opening TCP/IP to 192.168.58.104/1049 timeout=15
00:03:22: TAC+: Opened TCP/IP handle 0x489F08 to 192.168.58.104/1049
00:03:22: TAC+: periodic timer started
00:03:22: TAC+: 192.168.58.104 req=3BD868 id=299214410 ver=192 handle=0x489F08 (ESTAB)
expire=14 AUTHEN/START/SENDPASS/CHAP queued
00:03:23: TAC+: 192.168.58.104 ESTAB 3BD868 wrote 41 of 41 bytes
00:03:23: TAC+: 192.168.58.104 CLOSEWAIT read=12 wanted=12 alloc=12 got=12
00:03:23: TAC+: 192.168.58.104 CLOSEWAIT read=21 wanted=21 alloc=21 got=9
00:03:23: TAC+: 192.168.58.104 received 21 byte reply for 3BD868
00:03:23: TAC+: req=3BD868 id=299214410 ver=192 handle=0x489F08 (CLOSEWAIT) expire=13
AUTHEN/START/SENDPASS/CHAP processed
00:03:23: TAC+: periodic timer stopped (queue empty)
```

The TACACAS messages are intended to be self-explanatory or for consumption by service personnel only. However, the following messages shown in Figure 2-219 are briefly explained in the following text.

The following message indicates that a TCP open request to host 192.168.58.104 on port 1049 will time out in 15 seconds if it gets no response:

```
00:03:16: TAC+: Opening TCP/IP to 192.168.58.104/1049 timeout=15
```

The following message indicates a successful open operation and provides the address of the internal TCP “handle” for this connection:

```
00:03:16: TAC+: Opened TCP/IP handle 0x48A87C to 192.168.58.104/1049
```

The following message indicates that a TACACS+ request has been queued. The message identifies:

- Server that the request is destined for
- Internal address of the request
- TACACS+ ID of the request
- TACACS+ version number of the request
- Internal TCP handle the request uses (which will be zero for a single-connection server)
- TCP status of the connection—which is one of the following:
 - CLOSED
 - LISTEN
 - SYNSENT
 - SYNRCVD
 - ESTAB
 - FINWAIT1
 - FINWAIT2
 - CLOSEWAIT
 - LASTACK
 - CLOSING
 - TIMEWAIT
- Number of seconds until the request times out
- Request type

```
00:03:16: TAC+: 192.168.58.104 req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (ESTAB)
expire=14 AUTHEN/START/SENDAUTH/CHAP queued
```

The following message indicates that all 46 bytes were written to address 192.168.58.104 for request 3BD868:

```
00:03:17: TAC+: 192.168.58.104 ESTAB 3BD868 wrote 46 of 46 bytes
```

The following message indicates that 12 bytes were read in reply to the request.:

```
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=12 wanted=12 alloc=12 got=12
```

The following message indicates that 49 more bytes were read, making a total of 61 bytes in all, which is all that was expected:

```
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=61 wanted=61 alloc=61 got=49
```

The following message indicates that a complete 61-byte reply has been read and processed for request 3BD868:

```
00:03:22: TAC+: 192.168.58.104 received 61 byte reply for 3BD868 00:03:22: TAC+:  
req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (CLOSEWAIT) expire=9  
AUTHEN/START/SENDAUTH/CHAP processed
```

The following message indicates that the TACACS+ server helper process switched itself off when it had no more work to do:

```
00:03:22: TAC+: periodic timer stopped (queue empty)
```

Related Commands

- debug aaa accounting**
- debug aaa authentication**
- debug aaa authorization**
- debug tacacs**

debug tarp events

Use the **debug tarp events** EXEC command to display information on Target Identifier Address Resolution Protocol (TARP) activity. The **no** form of this command disables debugging output.

[no] debug tarp events

Usage Guidelines

For complete information on the TARP process, use the **debug tarp packets** command along with the **debug tarp events** command. Events are usually related to error conditions.

Sample Display

Figure 2-220 shows sample **debug tarp events** and **debug tarp packets** output after the **tarp resolve** command was used to determine the NSAP address for the TARP target identifier (TID) *artemis*.

Figure 2-220 Sample Debug TARP Events Output

```
router# debug tarp events
router# debug tarp packets
router# tarp resolve artemis

Type escape sequence to abort.
Sending TARP type 1 PDU, timeout 15 seconds...

NET corresponding to TID artemis is 49.0001.1111.1111.1111.00

*Mar 1 00:43:59: TARP-PA: Propagated TARP packet, type 1, out on Ethernet0
*Mar 1 00:43:59:      Lft = 100, Seq = 11, Prot type = 0xFE, URC = TRUE
*Mar 1 00:43:59:      Ttid len = 7, Stid len = 8, Prot addr len = 10
*Mar 1 00:43:59:      Destination NSAP: 49.0001.1111.1111.1111.00
*Mar 1 00:43:59:      Originator's NSAP: 49.0001.3333.3333.3333.00
*Mar 1 00:43:59:      Target TID: artemis
*Mar 1 00:43:59:      Originator's TID: cerd
*Mar 1 00:43:59: TARP-EV: Packet not propagated to 49.0001.4444.4444.4444.00 on
      interface Ethernet0 (adjacency is not in UP state)
*Mar 1 00:43:59: TARP-EV: No route found for TARP static adjacency
      55.0001.0001.1111.1111.1111.1111.1111.1111.1111.00 - packet not sent
*Mar 1 00:43:59: TARP-PA: Received TARP type 3 PDU on interface Ethernet0
*Mar 1 00:43:59:      Lft = 100, Seq = 5, Prot type = 0xFE, URC = TRUE
*Mar 1 00:43:59:      Ttid len = 0, Stid len = 7, Prot addr len = 10
*Mar 1 00:43:59:      Packet sent/propagated by 49.0001.1111.1111.1111.af
*Mar 1 00:43:59:      Originator's NSAP: 49.0001.1111.1111.1111.00
*Mar 1 00:43:59:      Originator's TID: artemis
*Mar 1 00:43:59: TARP-PA: Created new DYNAMIC cache entry for artemis
```

Table 2-114 describes the fields shown in Figure 2-220.

Table 2-114 Debug TARP Field Descriptions—TARP Resolve Command

| Field | Descriptions |
|-------------------------|---|
| Sending TARP type 1 PDU | PDU requesting the NSAP of the specified TID. |
| Timeout | Number of seconds the router will wait for a response from the Type 1 PDU. The timeout is set by the tarp t1-response-timer command. |

Table 2-114 Debug TARP Field Descriptions—TARP Resolve Command (Continued)

| Field | Descriptions |
|--|--|
| NET corresponding to | NSAP address (in this case, 49.0001.1111.1111.1111.00) for the specified TID. |
| *Mar 1 00:43:59 | Debug timestamp. |
| TARP-PA: Propagated | TARP packet: A Type 1 PDU was sent out on interface Ethernet 0. |
| Lft | Lifetime of the PDU (in hops). |
| Seq | Sequence number of the PDU. |
| Prot type | Protocol type of the PDU. |
| URC | The update remote cache bit. |
| Ttid len | Destination TID length. |
| Stid len | Source TID length. |
| Prot addr len | Protocol address length (bytes). |
| Destination NSAP | NSAP address that the PDU is being sent to. |
| Originator's NSAP | NSAP address that the PDU was sent from. |
| Target TID | TID that the PDU is being sent to. |
| Originator's TID | TID that the PDU was sent from. |
| TARP-EV: Packet not propagated | TARP event: The Type 1 PDU was not propagated on interface Ethernet 0 because the adjacency is not up. |
| TARP-EV: No route found | TARP event: The Type 1 PDU was not sent because no route was available. |
| TARP-PA: Received TARP | TARP packet: A Type 3 PDU was received on interface Ethernet 0. |
| Packet sent/propagated by | NSAP address of the router that sent or propagated the PDU. |
| TARP-PA: Created new DYNAMIC cache entry | TARP packet: A dynamic entry was made to the local TID cache. |

Related Command
debug tarp packets

debug tarp packets

Use the **debug tarp packets** EXEC command to display general information on TARP packets received, generated, and propagated on the router. The **no** form of this command disables debugging output.

[no] debug tarp packets

Usage Guidelines

For complete information on the TARP process, use the **debug tarp events** command along with the **debug tarp packet** command. Events are usually related to error conditions.

Sample Display

Figure 2-221 shows sample **debug tarp packet** output after the **tarp query** command was used to determine the TID for the NSAP address 49.0001.3333.3333.3333.00.

Figure 2-221 Sample Debug TARP Packets Output

```
router# debug tarp packets
router# debug tarp events
router# tarp query 49.0001.3333.3333.3333.00

Type escape sequence to abort.
Sending TARP type 5 PDU, timeout 40 seconds...

TID corresponding to NET 49.0001.3333.3333.3333.00 is cerdiwen

*Mar 2 03:10:11: TARP-PA: Originated TARP packet, type 5, to destination
49.0001.3333.3333.3333.00
*Mar 2 03:10:11: TARP-PA: Received TARP type 3 PDU on interface Ethernet0
*Mar 2 03:10:11:      Lft = 100, Seq = 2, Prot type = 0xFE, URC = TRUE
*Mar 2 03:10:11:      Ttid len = 0, Stid len = 8, Prot addr len = 10
*Mar 2 03:10:11:      Packet sent/propagated by 49.0001.3333.3333.3333.af
*Mar 2 03:10:11:      Originator's NSAP: 49.0001.3333.3333.3333.00
*Mar 2 03:10:11:      Originator's TID: cerdiwen
*Mar 2 03:10:11: TARP-PA: Created new DYNAMIC cache entry for cerdiwen
```

Table 2-115 describes the fields shown in the display.

Table 2-115 Debug TARP Field Descriptions—TARP Query Command

| Field | Descriptions |
|---------------------------------|---|
| Sending TARP type 5 PDU | PDU requesting the TID of the specified NSAP. |
| Timeout | Number of seconds the router will wait for a response from the Type 5 PDU. The timeout is set by the tarp arp-request-timer command. |
| TID corresponding to NET | TID (in this case <i>cerdiwen</i>) for the specified NSAP address. |
| *Mar 2 03:10:11 | Debug timestamp. |
| TARP-PA: Originated TARP packet | TARP packet: A Type 5 PDU was sent. |
| TARP P-A: Received TARP | TARP packet: A Type 3 PDU was received. |
| Lft | Lifetime of the PDU (in hops). |

Table 2-115 Debug TARP Field Descriptions—TARP Query Command (Continued)

| Field | Descriptions |
|--|---|
| Seq | Sequence number of the PDU. |
| Prot type | Protocol type of the PDU. |
| URC | The update remote cache bit. |
| Ttid len | Destination TID length. |
| Stid len | Source TID length. |
| Prot addr len | Protocol address length (bytes). |
| Packet sent/propagated | NSAP address of the router that sent or propagated the PDU. |
| Originator's NSAP | NSAP address that the PDU was sent from. |
| Originator's TID | TID that the PDU was sent from. |
| TARP-PA: Created new DYNAMIC cache entry | TARP packet: A dynamic entry was made to the local TID cache. |

Related Command

debug tarp events

debug tdm

Use the **debug tdm** EXEC command to display time division multiplexer (TDM) bus connection information each time a connection is made on the Cisco AS5200 access server. Use the **no** form of this command to disable debug output.

[no] debug tdm

Usage Guidelines

Use the **debug tdm** command if you are losing channel data between the dual T1 Primary Rate Interfaces (PRI) and any termination points, such as an Ethernet or modem point.

This command displays the TDM bus connection information for each TDM device installed in the access server. One TDM device exists on the PRI card, on the motherboard, and on each modem card. Expect up to 256 TDM connections to be displayed on your terminal when this command is enabled.

Sample Display

Figure 2-222 shows sample **debug tdm** output.

Figure 2-222 Sample Debug TDM Output

```
AS5200# debug tdm
dialtone connection requested.
TDM(reg: 0x2138100): Close connection to ST07, channel 1
TDM(reg: 0x2138100): Connect STi3, channel 1 to ST07, channel 1
```

debug tftp

Use the **debug tftp** EXEC command to display Trivial File Transfer Protocol (TFTP) debugging information when encountering problems netbooting or using the **copy tftp running-config** or **copy running-config tftp** commands. The **no** form of this command disables debugging output.

[no] debug tftp

Sample Display

Figure 2-223 shows sample **debug tftp** output from the EXEC command **write network**.

Figure 2-223 Sample Debug TFTP Output

```
router# debug tftp

TFTP: msclock 0x292B4; Sending write request (retry 0), socket_id 0x301DA8
TFTP: msclock 0x2A63C; Sending write request (retry 1), socket_id 0x301DA8
TFTP: msclock 0x2A6DC; Received ACK for block 0, socket_id 0x301DA8
TFTP: msclock 0x2A6DC; Received ACK for block 0, socket_id 0x301DA8
TFTP: msclock 0x2A6DC; Sending block 1 (retry 0), socket_id 0x301DA8
TFTP: msclock 0x2A6E4; Received ACK for block 1, socket_id 0x301DA8
```

Table 2-116 describes significant fields shown in the first line of output from Figure 2-223.

Table 2-116 Debug TFTP Field Descriptions

| Message | Description |
|---------------------------------|---|
| TFTP: | This entry describes a TFTP packet. |
| msclock 0x292B4; | Internal timekeeping clock (in milliseconds). |
| Sending write request (retry 0) | The TFTP operation. |
| socket_id 0x301DA8 | Unique memory address for the socket for the TFTP connection. |

debug token ring

Use the **debug token ring EXEC** command to display messages about Token Ring interface activity. The **no** form of this command disables debugging output.

[no] debug token ring

Usage Guidelines

This command reports several lines of information for each packet sent or received and is intended for low traffic, detailed debugging.

The Token Ring interface records provide information regarding the current state of the ring. These messages are only displayed when the **debug token events** command is enabled.

The **debug token ring** command invokes verbose Token Ring hardware debugging. This includes detailed displays as traffic arrives and departs the unit.

Note It is best to use this command only on router/bridges with light loads.

Sample Display

Figure 2-224 shows sample **debug token ring** output.

Figure 2-224 Sample Debug Token Ring Output

```
router# debug token ring

TR0: Interface is alive, phys. addr 5000.1234.5678
TR0: in: MAC: acfc: 0x1105 Dst: c000.ffff.ffff Src: 5000.1234.5678 bf: 0x45
TR0: in:      riflcn 0, rd_offset 0, llc_offset 40
TR0: out: MAC: acfc: 0x0040 Dst: 5000.1234.5678 Src: 5000.1234.5678 bf: 0x00
TR0: out: LLC: AAAA0300 00009000 00000100 AAC00000 00000802 50001234 ln: 28
TR0: in: MAC: acfc: 0x1140 Dst: 5000.1234.5678 Src: 5000.1234.5678 bf: 0x09
TR0: in: LLC: AAAA0300 00009000 00000100 AAC0B24A 4B4A6768 74732072 ln: 28
TR0: in:      riflcn 0, rd_offset 0, llc_offset 14
TR0: out: MAC: acfc: 0x0040 Dst: 5000.1234.5678 Src: 5000.1234.5678 bf: 0x00
TR0: out: LLC: AAAA0300 00009000 00000100 D1D00000 FE11E636 96884006 ln: 28
TR0: in: MAC: acfc: 0x1140 Dst: 5000.1234.5678 Src: 5000.1234.5678 bf: 0x09
TR0: in: LLC: AAAA0300 00009000 00000100 D1D0774C 4DC2078B 3D000160 ln: 28
TR0: in:      riflcn 0, rd_offset 0, llc_offset 14
TR0: out: MAC: acfc: 0x0040 Dst: 5000.1234.5678 Src: 5000.1234.5678 bf: 0x00
TR0: out: LLC: AAAA0300 00009000 00000100 F8E00000 FE11E636 96884006 ln: 28
```

Table 2-117 describes significant fields shown in the second line of output from Figure 2-224.

Table 2-117 Debug Token Ring Field Descriptions—Part 1

| Message | Description |
|---------|--|
| TR0: | Name of the interface associated with the Token Ring event. |
| in: | Indication of whether the packet was input to the interface (in) or output from the interface (out). |

Table 2-117 Debug Token Ring Field Descriptions—Part 1 (Continued)

| Message | Description |
|---------------------|--|
| MAC: | The type of packet, as follows: <ul style="list-style-type: none"> • MAC—Media Access Control • LLC—Link Level Control |
| acfc: 0x1105 | Access Control, Frame Control bytes, as defined by the IEEE 802.5 standard. |
| Dst: c000.ffff.ffff | Destination address of the frame. |
| Src: 5000.1234.5678 | Source address of the frame. |
| bf: 0x45 | Bridge flags for internal use by technical support staff. |

Table 2-118 describes significant fields shown in the third line of output from Figure 2-224.

Table 2-118 Debug Token Ring Field Descriptions—Part 2

| Message | Description |
|---------------|--|
| TR0: | Name of the interface associated with the Token Ring event. |
| in: | Indication of whether the packet was input to the interface (in) or output from the interface (out). |
| riflen 0 | Length of the RIF field (in bytes). |
| rd_offset 0 | Offset (in bytes) of the frame pointing to the start of the RIF field. |
| llc_offset 40 | Offset in the frame pointing to the start of the LLC field. |

Table 2-119 describes significant fields shown in the fifth line of output from Figure 2-224.

Table 2-119 Debug Token Ring Field Descriptions—Part 3

| Message | Description |
|----------|---|
| TR0: | Name of the interface associated with the Token Ring event. |
| out: | Indication of whether the packet was input to the interface (in) or output from the interface (out). |
| LLC: | The type of frame, as follows: <ul style="list-style-type: none"> • MAC—Media Access Control • LLC—Link Level Control |
| AAAA0300 | This and the octets that follow it indicate the contents (hex) of the frame. |
| ln: 28 | The length of the information field (in bytes). |

debug v120 event

Use the **debug v120 event** EXEC command to display information on V.120 activity. The **no** form of this command disables debugging output.

[no] debug v120 event

Usage Guidelines

V.120 is an ITU specification that allows for reliable transport of synchronous, asynchronous, or bit transparent data over ISDN bearer channels. For information on Cisco's implementation, refer to the "Configuring ISDN" chapter in the *Wide-Area Networking Configuration Guide*.

For complete information on the V.120 process, use the **debug v120 packet** command along with the **debug v120 event** command. V.120 events are activity events rather than error conditions.

Sample Display

Figure 2-225 shows sample **debug v120 event** output of V.120 starting up and stopping. Also included is the interface that V.120 is running on (BR 0) and where the V.120 configuration parameters are obtained from (default).

Figure 2-225 Sample Debug V.120 Event Output

```
router# debug v120 event  
  
0:01:47: BR0:1-v120 started - Setting default V.120 parameters  
0:02:00: BR0:1:removing v120
```

Related Command

debug v120 packet

debug v120 packet

Use the **debug v120 packet** EXEC command to display general information on all incoming and outgoing V.120 packets. The **no** form of this command disables debugging output.

[no] debug v120 packet

Usage Guidelines

The **debug v120 packet** command shows every packet on the V.120 session. You can use this information to determine whether incompatibilities exist between Cisco's V.120 implementation and other vendors' V.120 implementations.

V.120 is an ITU specification that allows for reliable transport of synchronous, asynchronous, or bit transparent data over ISDN bearer channels. For information on Cisco's implementation, refer to the "Configuring ISDN" chapter in the *Wide-Area Networking Configuration Guide*.

For complete information on the V.120 process, use the **debug v120 events** command along with the **debug v120 packet** command.

Sample Display

Figure 2-226 shows sample **debug v120 packet** output for a typical session startup.

Figure 2-226 Sample Debug V.120 Packet Output

```
router# debug v120 packet

0:03:27: BR0:1: I SABME:11i 256 C/R 0 P/F=1
0:03:27: BR0:1: O UA:11i 256 C/R 1 P/F=1
0:03:27: BR0:1: O IFRAME:11i 256 C/R 0 N(R)=0 N(S)=0 P/F=0 len 43
0x83 0xD 0xA 0xD 0xA 0x55 0x73 0x65
0x72 0x20 0x41 0x63 0x63 0x65 0x73 0x73
0:03:27: BR0:1: I RR:11i 256 C/R 1 N(R)=1 P/F=0
0:03:28: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=0 P/F=0 len 2
0x83 0x63
0:03:28: BR0:1: O RR:11i 256 C/R 1 N(R)=1 P/F=0
0:03:29: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=1 P/F=0 len 2
0x83 0x31
0:03:29: BR0:1: O RR:11i 256 C/R 1 N(R)=2 P/F=0
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to up
0:03:31: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=2 P/F=0 len 2
0x83 0x55
0:03:32: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=3 P/F=0 len 3
0x83 0x31 0x6F
0:03:32: BR0:1: O RR:11i 256 C/R 1 N(R)=3 P/F=0
0:03:32: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=4 P/F=0 len 2
0x83 0x73
0:03:32: BR0:1: O RR:11i 256 C/R 1 N(R)=5 P/F=0
0:03:32: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=5 P/F=0 len 2
0x83 0xA
0:03:32: BR0:1: O IFRAME:11i 256 C/R 0 N(R)=6 N(S)=1 P/F=0 len 9
0x83 0xD 0xA 0x68 0x65 0x66 0x65 0x72 0x3E
```

Table 2-120 describes the fields shown in the display.

Table 2-120 Debug V.120 Packet Field Descriptions

| Field | Descriptions |
|-----------------------|--|
| BR0:1 | Interface number associated with this debugging information. |
| I/O | Packet going into or out of the interface. |
| SABME, UA, IFRAME, RR | V.120 packet type. In this case: <ul style="list-style-type: none">• SABME—set asynchronous balanced mode, extended• US—unnumbered acknowledgment• IFRAME—information frame• RR—receive ready |
| lli 256 | Logical link identifier number. |
| C/R 0 | Command or response. |
| P/F=1 | Poll final. |
| N(R)=0 | Number received. |
| N(S)=0 | Number sent. |
| len 43 | Number of data bytes in the packet. |
| 0x83 | Up to 16 bytes of data. |

Related Command**debug v120 event**

debug vpdn

To display debug traces for the Virtual Private Dialup Network (VPDN) feature, which provides PPP tunnels using the Layer 2 Forwarding (L2F) protocol, use the **debug vpdn EXEC** command. The **no** of this command disables debugging output.

```
[no] debug vpdn { errors | events | packets | l2f-errors | l2f-events | l2f-packets }
```

Syntax Description

| | |
|--------------------|--|
| errors | Displays errors that prevent a tunnel from being established or errors that cause an established tunnel to be closed. |
| events | Displays messages about events that are part of normal tunnel establishment or shutdown. |
| packets | Displays each protocol packet exchanged. This option may result in a large number of debug messages and should generally only be used on a debug chassis with a single active session. |
| l2f-errors | Displays L2F protocol errors that prevent L2F establishment or prevent its normal operation. |
| l2f-events | Displays messages about events that are part of normal tunnels establishment or shutdown for L2F. |
| l2f-packets | Displays messages about L2F protocol headers and status. |

Debug VPDN Events on a Network Access Server—Normal Operations

The network access server has the following VPDN configuration:

```
vpdn outgoing cisco.com stella ip 172.21.9.26
username stella password stella
```

Figure 2-227 shows sample output of the **debug vpdn events** command on a network access server when the L2F tunnel is brought up and CHAP authentication of the tunnel succeeds.

Figure 2-227 Debug VPDN Events on the Network Access Server—Tunnel Coming Up

```
Router# debug vpdn events

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar  2 00:26:05.537: looking for tunnel -- cisco.com --
*Mar  2 00:26:05.545: Async6 VPN Forwarding...
*Mar  2 00:26:05.545: Async6 VPN Bind interface direction=1
*Mar  2 00:26:05.553: Async6 VPN vpn_forward_user bum6@cisco.com is forwarded
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar  2 00:26:06.289: L2F:  Chap authentication succeeded for stella.
```

Figure 2-228 shows sample output of the **debug vpdn events** command on a network access server when the L2F tunnel is brought down normally:

Figure 2-228 Debug VPDN Events on the Network Access Server—Tunnel Coming Down Normally

```
Router# debug vpdn events

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:27:18.865: Async6 VPN cleanup
*Mar 2 00:27:18.869: Async6 VPN reset
*Mar 2 00:27:18.873: Async6 VPN Unbind interface
%LINK-3-UPDOWN: Interface Async6, changed state to down
```

Table 2-121 describes the fields in the **debug vpdn events** command output. The output describes normal operations when a tunnel is brought up or down on a network access server.

Table 2-121 Debug VPDN Events Field Descriptions for the Network Access Server

| Field | Description |
|---|---|
| Asynchronous interface coming up | |
| %LINK-3-UPDOWN: Interface Async6, changed state to up | Asynchronous interface 6 came up. |
| looking for tunnel -- cisco.com -- Async6 VPN Forwarding... | Domain name is identified. |
| Async6 VPN Bind interface direction=1 | Tunnel is bound to the interface. These are the direction values: 1—From the network access server to the home gateway 2—From the home gateway to the network access server |
| Async6 VPN vpn_forward_user bum6@cisco.com is forwarded | Tunnel for the specified user and domain name (bum6@cisco.com) is forwarded. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up | Line protocol is up. |
| L2F: Chap authentication succeeded for stella. | Tunnel was authenticated with the tunnel password <i>stella</i> . |
| Virtual access interface coming down | |
| Async6 VPN cleanup Async6 VPN reset Async6 VPN Unbind interface Async6 VPN reset | Normal cleanup operations performed when the line or virtual access interface goes down. |
| %LINEPROTO-5-UPDOWN: Line protocol on interface Async6, changed state to down | Normal operation when the virtual access interface is taken down. |

Debug VPDN Events on the Home Gateway—Normal Operations

The home gateway has the following VPDN configuration, which uses *stella* as the tunnel name and the tunnel authentication name. The tunnel authentication name might be entered in a users file on an AAA server and used to define authentication requirements for the tunnel.

```
vpdn incoming stella stella virtual-template 1
```

Figure 2-229 shows sample output of the **debug vpdn events** command on the home gateway when the tunnel is brought up successfully.

Figure 2-229 Debug VPDN Events on the Home Gateway—Tunnel Coming Up

```
Router# debug vpdn events

L2F: Chap authentication succeeded for stella.
Virtual-Access3 VPN Virtual interface created for bum6@cisco.com
Virtual-Access3 VPN Set to Async interface
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up
Virtual-Access3 VPN Bind interface direction=2
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up
```

Figure 2-230 shows sample output of the **debug vpdn events** command on a home gateway when the tunnel is brought down normally.

Figure 2-230 Debug VPDN Events on the Home Gateway—Tunnel Coming Down Normally

```
Router# debug vpdn events

%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down
Virtual-Access3 VPN cleanup
Virtual-Access3 VPN reset
Virtual-Access3 VPN Unbind interface
Virtual-Access3 VPN reset
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down
```

Table 2-122 describes the fields in the **debug vpdn events** command output. The output describes normal operations when a tunnel is brought up or down on a network access server.

Table 2-122 Debug VPDN Events Field Descriptions for a Home Gateway

| Field | Description |
|--|---|
| Tunnel Coming Up | |
| L2F: Chap authentication succeeded for stella. | PPP CHAP authentication status for the tunnel named <i>stella</i> . |
| Virtual-Access3 VPN Virtual interface created for bum6@cisco.com | Virtual access interface was set up on the home gateway for the user bum6@cisco.com. |
| Virtual-Access3 VPN Set to Async interface | Virtual access interface 3 was set to asynchronous for character-by-character transmission. |
| Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0 | Virtual template 1 was applied to virtual access interface 3. |

Table 2-122 Debug VPDN Events Field Descriptions for a Home Gateway (Continued)

| Field | Description |
|---|---|
| %LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up | Link status is set to up. |
| Virtual-Access3 VPN Bind interface direction=2 | Tunnel is bound to the interface. These are the direction values: 1—From the network access server to the home gateway 2—From the home gateway to the network access server |
| Virtual-Access3 VPN PPP LCP accepted sent & rev CONFACK | PPP LCP configuration settings (negotiated between the remote client and the network access server) were copied to the home gateway and acknowledged. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up | Line protocol is up; the line can be used. |
| Tunnel Coming Down | |
| %LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down | Virtual access interface is coming down. |
| Virtual-Access3 VPN cleanup Virtual-Access3 VPN reset Virtual-Access3 VPN Unbind interface Virtual-Access3 VPN reset | Router is performing normal cleanup operations when a virtual access interface used for an L2F tunnel comes down. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down | Line protocol is down for virtual access interface 3; the line cannot be used. |

Debug VPDN L2F-Events on the Network Access Server—Normal Operations

Figure 2-231 shows sample output of the **debug vpdn l2f-events** command on the network access server when the L2F tunnel is brought up successfully.

Figure 2-231 Debug VPDN L2F-Events on the Network Access Server—Tunnel Coming Up

```
Router# debug vpdn l2f-events

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:41:17.365: L2F Open UDP socket to 172.21.9.26
*Mar 2 00:41:17.385: L2F_CONF received
*Mar 2 00:41:17.389: L2F Removing resend packet (type 1)
*Mar 2 00:41:17.477: L2F_OPEN received
*Mar 2 00:41:17.489: L2F Removing resend packet (type 2)
*Mar 2 00:41:17.493: L2F building nas2gw_mid0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:41:18.613: L2F_OPEN received
*Mar 2 00:41:18.625: L2F Got a MID management packet
*Mar 2 00:41:18.625: L2F Removing resend packet (type 2)
*Mar 2 00:41:18.629: L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6
```

Figure 2-232 shows sample output of the **debug vpdn l2f-events** command on a network access server when the tunnel is brought down normally.

Figure 2-232 Debug VPDN L2F-Events on the Network Access Server—Tunnel Coming Down

```
Router# debug vpdn l2f-events

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:42:29.213: L2F_CLOSE received
*Mar 2 00:42:29.217: L2F Destroying mid
*Mar 2 00:42:29.217: L2F Removing resend packet (type 3)
*Mar 2 00:42:29.221: L2F Tunnel is going down!
*Mar 2 00:42:29.221: L2F Initiating tunnel shutdown.
*Mar 2 00:42:29.225: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F Got closing for tunnel
*Mar 2 00:42:29.233: L2F Removing resend packet
*Mar 2 00:42:29.233: L2F Closed tunnel structure
%LINK-3-UPDOWN: Interface Async6, changed state to down
*Mar 2 00:42:31.793: L2F Closed tunnel structure
*Mar 2 00:42:31.793: L2F Deleted inactive tunnel
```

Table 2-123 describes the fields in the **debug vpdn l2f-events** command output displays.

Table 2-123 Debug VPDN L2F-Events Field Descriptions for the Network Access Server

| Field | Descriptions |
|---|--|
| Tunnel Coming Up | |
| %LINK-3-UPDOWN: Interface Async6, changed state to up | Asynchronous interface came up normally. |
| L2F Open UDP socket to 172.21.9.26 | L2F opened a UDP socket to the home gateway IP address. |
| L2F_CONF received | The L2F_CONF signal was received. When sent from the home gateway to the network access server, an L2F_CONF indicates the home gateway's recognition of the tunnel creation request. |
| L2F Removing resend packet (type ...) | Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel. |
| L2F_OPEN received | The L2F_OPEN management message was received, indicating that home gateway accepted the network access server configuration of an L2F tunnel. |
| L2F building nas2gw_mid0 | L2F is building a tunnel between the network access server and the home gateway, using MID 0. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up | The line protocol came up. Indicates whether the software processes that handle the line protocol regard the interface as usable. |
| L2F_OPEN received | The L2F_OPEN management message was received, indicating that home gateway accepted the network access server configuration of an L2F tunnel. |

Table 2-123 Debug VPDN L2F-Events Field Descriptions for the Network Access Server

| Field | Descriptions |
|---|--|
| L2F Got a MID management packet | Multiplex ID (MID) management packets are used to communicate between the network access server and the home gateway. |
| L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6 | L2F synchronized the Client IDs on the network access server and the home gateway, respectively. A multiplex ID is assigned to identify this connection in the tunnel. |
| Tunnel Coming Down | |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down | The line protocol came down. Indicates whether the software processes that handle the line protocol regard the interface as usable. |
| %LINK-5-CHANGED: Interface Async6, changed state to reset | Interface was marked as reset. |
| L2F_CLOSE received | The network access server received a request to close the tunnel. |
| L2F Destroying mid | The connection identified by the MID is begin taken down. |
| L2F Tunnel is going down! | Advisory message about impending tunnel shutdown. |
| L2F Initiating tunnel shutdown. | Tunnel shutdown has started. |
| L2F_CLOSE received | The network access server received a request to close the tunnel. |
| L2F Got closing for tunnel | The network access server began tunnel closing operations. |
| %LINK-3-UPDOWN: Interface Async6, changed state to down | The asynchronous interface was taken down. |
| L2F Closed tunnel structure | The network access server closed the tunnel. |
| L2F Deleted inactive tunnel | The now-inactivated tunnel was deleted. |

Debug VPDN L2F-Events on the Home Gateway—Normal Operations

Figure 2-233 shows sample output of the `debug vpdn l2f-events` command on a home gateway when the L2F tunnel is created.

Figure 2-233 Debug VPDN L2F-Events on the Home Gateway—Tunnel is Created

```
Router# debug vpdn l2f-events

L2F_CONF received
L2F Creating new tunnel for stella
L2F Got a tunnel named stella, responding
L2F Open UDP socket to 172.21.9.25
L2F_OPEN received
L2F Removing resend packet (type 1)
L2F_OPEN received
L2F Got a MID management packet
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
```

Figure 2-234 shows sample output of the **debug vpdn l2f-events** command on a home gateway when the L2F tunnel is brought down normally.

Figure 2-234 Debug VPDNA L2F-Events on a Home Gateway—Tunnel Coming Down Normally

```
Router# debug vpdn l2f-events

L2F_CLOSE received
L2F Destroying mid
L2F Removing resend packet (type 3)
L2F Tunnel is going down!
L2F Initiating tunnel shutdown.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
L2F_CLOSE received
L2F Got closing for tunnel
L2F Removing resend packet
L2F Removing resend packet
L2F Closed tunnel structure
L2F Closed tunnel structure
L2F Deleted inactive tunnel
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down
```

Table 2-124 describes the fields in the output of the **debug vpdn l2f-events** command for the home gateway when the tunnel is being created and when the tunnel is being brought down.

Table 2-124 Debug VPDN L2F-Events Field Descriptions for the Home Gateway

| Field | Description |
|--|---|
| Tunnel Coming Up | |
| L2F_CONF received | L2F configuration is received from the network access server. When sent from a network access server to a home gateway, the L2F_CONF is the initial packet in the conversation. |
| L2F Creating new tunnel for stella | The tunnel named <i>stella</i> is being created. |
| L2F Got a tunnel named stella, responding | Home gateway is responding. |
| L2F Open UDP socket to 172.21.9.25 | Opening a socket to the network access server IP address. |
| L2F_OPEN received | The L2F_OPEN management message was received, indicating the network access server is opening an L2F tunnel. |
| L2F Removing resend packet (type ...) | Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel. |
| L2F Got a MID management packet | L2F MID management packets are used to communicate between the network access server and the home gateway. |
| %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up | The home gateway is bringing up virtual access interface 1 for the L2F tunnel. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up | The line protocol is up. The line can be used. |

Table 2-124 Debug VPDN L2F-Events Field Descriptions for the Home Gateway

| Field | Description |
|---|---|
| Tunnel Coming Down | |
| L2F_CLOSE received | The network access server or home gateway received a request to close the tunnel. |
| L2F Destroying mid | The connection identified by the MID is begin taken down. |
| L2F Removing resend packet (type ...) | Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel. |
| L2F Tunnel is going down! L2F Initiating tunnel shutdown. | Router is performing normal operations when a tunnel is coming down. |
| %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down | The virtual access interface is coming down. |
| L2F_CLOSE received L2F Got closing for tunnel L2F Removing resend packet L2F Removing resend packet L2F Closed tunnel structure L2F Closed tunnel structure L2F Deleted inactive tunnel | Router is performing normal cleanup operations when the tunnel is being brought down. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down | Line protocol is down; virtual access interface 1 cannot be used. |

Debug VPDN on the Network Access Server—Error Conditions

Figure 2-235 shows sample output of the **debug vpdn errors** command on a network access server when the tunnel is not set up.

Figure 2-235 Debug VPDN Errors on the Network Access Server—Error Conditions

```
Router# debug vpdn errors

VPN errors debugging is on

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down
%LINK-5-CHANGED: Interface Async1, changed state to reset
%LINK-3-UPDOWN: Interface Async1, changed state to down
%LINK-3-UPDOWN: Interface Async1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up
VPDN tunnel management packet failed to authenticate
VPDN tunnel management packet failed to authenticate
```

Table 2-125 describes the fields in the **debug vpdn errors** command output for the network access server.

Table 2-125 Debug VPDN Errors Fields Descriptions for the Network Access Server

| Field | Description |
|---|---|
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down | The line protocol on the asynchronous interface went down. |
| %LINK-5-CHANGED: Interface Async1, changed state to reset | Asynchronous interface 1 was reset. |
| %LINK-3-UPDOWN: Interface Async1, changed state to down | The link from asynchronous interface 1 link went down and then came back up. |
| %LINK-3-UPDOWN: Interface Async1, changed state to up | |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up | The line protocol on the asynchronous interface came back up. |
| VPDN tunnel management packet failed to authenticate | Tunnel authentication failed. This the most common VPDN error. Note: Check the password for the network access server and the home gateway name. If you store the password on an AAA server, you can use the debug aaa authentication command. |

Figure 2-236 shows sample output of the **debug vpdn l2f-errors** command.

Figure 2-236 Debug VPDN L2F-Errors on the Network Access Server—Error Conditions

```
router# debug vpdn l2f-errors

L2F protocol errors debugging is on

%LINK-3-UPDOWN: Interface Async1, changed state to up
L2F Out of sequence packet 0 (expecting 0)
L2F Tunnel authentication succeeded for home.com
L2F Received a close request for a non-existent mid
L2F Out of sequence packet 0 (expecting 0)
L2F packet has bogus1 key 1020868 D248BA0F
L2F packet has bogus1 key 1020868 D248BA0F
```

Table 2-126 describes the fields in the **debug vpdn l2f-errors** command output.

Table 2-126 Debug VPDN L2F-Errors Field Descriptions

| Field | Description |
|---|---|
| %LINK-3-UPDOWN: Interface Async1, changed state to up | The line protocol on the asynchronous interface came up. |
| L2F Out of sequence packet 0 (expecting 0) | Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received. |

Table 2-126 Debug VPDN L2F-Errors Field Descriptions (Continued)

| Field | Description |
|---|--|
| L2F Tunnel authentication succeeded for home.com | Tunnel was established from the network access server to the home gateway, home.com. |
| L2F Received a close request for a non-existent mid | Multiplex ID was not used previously; cannot close the tunnel. |
| L2F Out of sequence packet 0 (expecting 0) | Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received. |
| L2F packet has bogus1 key 1020868 D248BA0F | Value based on the authentication response given to the peer during tunnel creation. This packet, in which the key does not match the expected value, must be discarded. |
| L2F packet has bogus1 key 1020868 D248BA0F | Another packet was received with an invalid key value. The packet must be discarded. |

Debugging VPDN Packets for Complete Information

Figure 2-237 shows sample output of the **debug vpdn l2f-packets** command on a network access server. This example displays a trace for a **ping** command.

Figure 2-237 Debug VPDN L2F-Packets on a Network Access Server

```

Router# debug vpdn l2f-packets

L2F protocol packets debugging is on

L2F SENDING (17): D0 1 1 10 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 16 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 10 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F SENDING (17): D0 1 1 11 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 17 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 11 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F header flags: 57345 version 57345 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key 1701976070
L2F-IN Output to Async1 (16): FF 3 C0 21 9 F 0 C 0 1D 41 AD FF 11 46 87
L2F-OUT (16): FF 3 C0 21 A F 0 C 0 1A C9 BD FF 11 46 87
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key -2120949344
L2F-OUT (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 3 1 0 0 1 8 0 62 B1
0 0 C A8 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key -2120949344
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key 1701976070
L2F-IN Output to Async1 (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 1 1 0
0 3 0 0 6A B1 0 0 C A8 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD

```

Table 2-127 describes the fields in the **debug vpdn l2f-packets** display.

Table 2-127 Debug VPDN L2F-Packets Field Descriptions

| Field | Description |
|-----------------------------|---|
| L2F SENDING (17) | Number of bytes being sent. The first set of "SENDING"... "RECEIVED" lines displays L2F keepalive traffic. The second set displays L2F management data. |
| L2F header flags: | Version and flags, in decimal. |
| version 53249 | Version. |
| protocol 1 | The protocol for negotiation of the point-to-point link between the network access server and the home gateway is always 1, indicating L2F management. |
| sequence 16 | Sequence numbers start at 0. Each subsequent packet is sent with the next increment of the sequence number. The sequence number is thus a free running counter represented modulo 256. There is distinct sequence counter for each distinct MID value. |
| mid 0 | Multiplex ID, which identifies a particular connection within the tunnel. Each new connection is assigned a MID currently unused within the tunnel. |
| cid 4 | Client ID used to assist endpoints in demultiplexing tunnels. |
| length 17 | Size in octets of the entire packet, including header, all fields present, and payload. Length does not reflect the addition of the checksum, if present. |
| offset 0 | Number of bytes past the L2F header at which the payload data is expected to start. If it is 0, the first byte following the last byte of the L2F header is the first byte of payload data. |
| key 1701976070 | Value based on the authentication response given to the peer during tunnel creation. During the life of a session, the key value serves to resist attacks based on spoofing. If a packet is received in which the key does not match the expected value, the packet must be silently discarded. |
| L2F RECEIVED (17) | Number of bytes received. |
| L2F-IN Otput to Async1 (16) | Payload datagram. The data came in to the VPDN code. |
| L2F-OUT (16): | Payload datagram sent out from the VPDN code to the tunnel. |
| L2F-OUT (101) | Ping payload datagram. The value 62 in this line is the ping packet size in hexadecimal (98 in decimal). The three lines that follow this line show ping packet data. |

Related Commands

debug aaa authentication
debug ppp negotiations
debug ppp error
debug ppp authentication

debug vines arp

Use the **debug vines arp** EXEC command to display debugging information on all Virtual Integrated Network Service (VINES) Address Resolution Protocol (ARP) packets that the router sends or receives. The **no** form of this command disables debugging output.

[no] debug vines arp

Sample Display

Figure 2-238 shows sample **debug vines arp** output.

Figure 2-238 Sample Debug VINES ARP Output

```
router# debug vines arp
VNSARP: received ARP type 0 from 0260.8c43.a7e4
VNSARP: sending ARP type 1 to 0260.8c43.a7e4
VNSARP: received ARP type 2 from 0260.8c43.a7e4
VNSARP: sending ARP type 3 to 0260.8c43.a7e4 assigning address 3001153C:8004
VSARP: received ARP type 0 from 0260.8342.1501
VSARP: sending ARP type 1 to 0260.8342.1501
VSARP: received ARP type 2 from 0260.8342.1501
VSARP: sending ARP type 3 to 0260.8342.1501 assigning address 3001153C:8005,
sequence 143C, metric 2
```

In Figure 2-238, the first four lines show a non-sequenced ARP transaction and the second four lines show a sequenced ARP transaction. Within the first group of four lines, the first line shows that the router received an ARP request (type 0) from indicated station address 0260.8c43.a7e4. The second line shows that the router is sending back the ARP service response (type 1), indicating that it is willing to assign VINES Internet addresses. The third line shows that the router received a VINES Internet address assignment request (type 2) from address 0260.8c43.a7e4. The fourth line shows that the router is responding (type 3) to the address assignment request from the client and assigning it the address 3001153C:8004.

Within the second group of four lines, the sequenced ARP packet also includes the router' current sequence number and the metric value between the router and the client.

Table 2-128 describes significant fields shown in Figure 2-238.

Table 2-128 Debug VINES ARP Field Descriptions

| Field | Description |
|---------------------|---|
| VNSARP: | Banyan VINES nonsequenced ARP message. |
| VSARP: | Banyan VINES sequenced ARP message. |
| received ARP type 0 | ARP request of type 0 was received. Possible type values follow: <ul style="list-style-type: none"> • 0—Query request. The ARP client broadcasts a type 0 message to request an ARP service to respond. • 1—Service response. The ARP service responds with a type 1 message to an ARP client's query request. • 2—Assignment request. The ARP client responds to a service response with a type 2 message to request a VINES Internet address. • 3—Assignment response. The ARP service responds to an assignment request with a type 3 message that includes the assigned VINES Internet address. |
| from 0260.8c43.a7e4 | Indicates the source address of the packet. |

debug vines echo

Use the **debug vines echo** EXEC command to display information on all MAC-level echo packets that the router sends or receives. Banyan VINES interface testing programs make use of these echo packets. The **no** form of this command disables debugging output.

[no] debug vines echo

Note These echo packets do not include network layer addresses.

Sample Display

Figure 2-239 shows sample **debug vines echo** output.

Figure 2-239 Sample Debug VINES Echo Output

```
router# debug vines echo  
  
VINESECHO: 100 byte packet from 0260.8c43.a7e4
```

Table 2-129 describes the fields shown in Figure 2-239.

Table 2-129 Debug VINES Echo Field Descriptions

| Field | Description |
|---------------------|--|
| VINESECHO | Indication that this is a debug vines echo message. |
| 100 byte packet | Packet size in bytes. |
| from 0260.8c43.a7e4 | Source address of the echo packet. |

debug vines ipc

Use the **debug vines ipc** EXEC command to display information on all transactions that occur at the VINES IPC layer, which is one of the two VINES transport layers. The **no** form of this command disables debugging output.

[no] debug vines ipc

Usage Guidelines

You can use the **debug vines ipc** command to discover why an IPC layer process on the router is not communicating with another IPC layer process on another router or Banyan VINES server.

Sample Display

Figure 2-240 shows sample **debug vines ipc** output for three pairs of transactions. For more information about these fields or their values, refer to Banyan VINES documentation.

Figure 2-240 Sample Debug VINES IPC Output

```
router# debug vines ipc

VIPC: sending IPC Data to Townsaver port 7 from port 7
      r_cid 0, l_cid 1, seq 1, ack 0, length 12
VIPC: received IPC Data from Townsaver port 7 to port 7
      r_cid 51, l_cid 1, seq 1, ack 1, length 32
VIPC: sending IPC Ack to Townsaver port 0 from port 0
      r_cid 51, l_cid 1, seq 1, ack 1, length 0
```

Table 2-130 describes the fields shown in Figure 2-240.

Table 2-130 Debug VINES IPC Field Descriptions

| Field | Description |
|---|---|
| VIPC: | Indicates that this is output from the debug vines ipc command. |
| sending | Indicates that the router is either sending an IPC packet to another router or has received an IPC packet from another router. |
| IPC Data to | Indicates the type of IPC frame: <ul style="list-style-type: none"> • Acknowledgment • Data • Datagram • Disconnect • Error • Probe |
| Townsaver port 7 | Indicates the machine name as assigned using the VINES host command, or IP address of the other router. Also indicates the port on that machine through which the packet has been transmitted. |
| from port 7 | Indicates the port on the router through which the packet has been transmitted. |
| r_cid 0, l_cid 1, seq 1, ack 0, length 12 | Indicates the values for various fields in the IPC layer header of this packet. Refer to Banyan VINES documentation for more information. |

debug vines netrpc

Use the **debug vines netrpc** EXEC command to display information on all transactions that occur at the VINES NetRPC layer, which is the VINES Session/Presentation layer. The **no** form of this command disables debugging output.

[no] debug vines netrpc

Usage Guidelines

You can use the **debug vines netrpc** command to discover why a NetRPC layer process on the router is not communicating with another NetRPC layer process on another router or Banyan server.

Sample Display

Figure 2-241 shows sample **debug vines netrpc** output. For more information about these fields or their values, refer to Banyan VINES documentation.

Figure 2-241 Sample Debug VINES NetRPC Output

```
router# debug vines netrpc

VRPC: sending RPC call to Townsaver
VRPC: received RPC return from Townsaver
```

Table 2-131 describes the fields shown in the first line of output in Figure 2-241.

Table 2-131 Debug VINES NetRPC Field Descriptions

| Field | Description |
|---------------------|---|
| VRPC: | Indicates that this is output from the debug vines netrpc command. |
| sending RPC call | Indicates that the router is either sending a NetRPC packet to another router or has received a NetRPC packet from another router. |
| Townsaver | Indicates the transaction type: <ul style="list-style-type: none"> • abort • call • reject • return • return address • search • search all |
| | Indicates the machine name as assigned using the VINES host command or IP address of the other router. |

debug vines packet

Use the **debug vines packet** EXEC command to display general VINES debugging information. This information includes packets received, generated, and forwarded, as well as failed access checks and other operations. The **no** form of this command disables debugging output.

[no] debug vines packet

Sample Display

Figure 2-242 shows sample **debug vines packet** output.

Figure 2-242 Sample Debug VINES Packet Output

```
router# debug vines packet

VINES: s=30028CF9:1 (Ether2), d=FFFFFFFF:FFFF, rcvd w/ hops 0
VINES: s=3000CBD4:1 (Ether1), d=3002ABEA:1 (Ether2), g=3002ABEA:1, sent
VINES: s=3000CBD4:1 (Ether1), d=3000B959:1, rcvd by gw
VINES: s=3000B959:1 (local), d=3000CBD4:1 (Ether1), g=3000CBD4:1, sent
```

The following information describes selected lines of output from Figure 2-242.

Table 2-132 describes the fields shown in the first line of output.

Table 2-132 Debug VINES Packet Field Descriptions

| Field | Description |
|-------------------|--|
| VINES: | Indicates that this is a Banyan VINES packet. |
| s = 30028CF9:1 | Indicates source address of the packet. |
| (Ether2) | Indicates the interface through which the packet was received. |
| d = FFFFFFFF:FFFF | Indicates that the destination is a broadcast address. |
| rcvd w/ hops 0 | Indicates that the packet was received because it was a local broadcast packet. The remaining hop count in the packet was zero (0). |

In the following line, the destination is the address 3002ABEA:1 associated with interface Ether2. Source address 3000CBD4:1 sent a packet to this destination through the gateway at address 3000ABEA:1.

```
VINES: s=3000CBD4:1 (Ether1), d=3002ABEA:1 (Ethernet2), g=3002ABEA:1, sent
```

In the following line, the router being debugged is the destination address (3000B959:1):

```
VINES: s=3000CBD4:1 (Ether1), d=3000B959:1, rcvd by gw
```

In the following line, (local) indicates that the router being debugged generated the packet:

```
VINES: s=3000B959:1 (local), d=3000CBD4:1 (Ether1), g=3000CBD4:1, sent
```

debug vines routing

Use the **debug vines routing** EXEC command to display information on all VINES RTP update messages sent or received and all routing table activities that occur in the router. The **no** form of this command disables debugging output.

```
debug vines routing [verbose]
no debug vines routing
```

Syntax Description

verbose (Optional) Provides detailed information about the contents of each update.

Sample Displays

Figure 2-243 shows sample **debug vines routing** output.

Figure 2-243 Sample Debug VINES Routing Output

```
router# debug vines routing
VSRTTP: generating change update, sequence number 0002C791
Update sent VSRTTP: sent update to Broadcast on Hssi0
Update received VSRTTP: received update from LabRouter on Hssi0
VSRTTP: LabRouter-Hs0-HDLC up -> up, change update, onemore
VRTP: sending update to Broadcast on Ethernet0
VSRTTP: generating null update
VSRTTP: Sending update to Aloe on Hssi0
```

S2854

Figure 2-244 shows sample **debug vines routing verbose** output.

Figure 2-244 Sample Debug VINES Routing Verbose Output

```
router# debug vines routing verbose
VRTP: sending update to Broadcast on Ethernet0
network 30011E7E, metric 0020 (0.4000 seconds)
network 30015800, metric 0010 (0.2000 seconds)
network 3003148A, metric 0020 (0.4000 seconds)
VSRTTP: generating change update, sequence number 0002C795
network Router9 metric 0010, seq 00000000, flags 09
network RouterZZ metric 0230, seq 00052194, flags 02
VSRTTP: sent update to Broadcast on Hssi0
VSRTTP: received update from LabRouter on Hssi0
update: type 00, flags 07, id 000E, ofst 0000, seq 15DFC, met 0010
network LabRouter from the server
network Router9 metric 0020, seq 00000000, flags 09
VSRTTP: LabRouter-Hs0-HDLC up -> up, change update, onemore
```

Figure 2-244 describes two VINES routing updates; the first includes two entries and the second includes three entries. The following information describes selected lines of output.

The following line shows that the router sent a periodic routing update to the broadcast address FFFFFFFF:FFFF through the Ethernet0 interface:

```
VRTP: sending update to Broadcast on Ethernet0
```

The following line indicates that the router knows how to reach network 30011E7E, which is a metric of 0020 away from the router. The value that follows the metric (0.4000 seconds) interprets the metric in seconds.

```
network 30011E7E, metric 0020 (0.4000 seconds)
```

The following lines show that the router sent a change routing update to the Broadcast addresses on the Hssi0 interface using the Sequenced Routing Update Protocol (SRTP) routing protocol:

```
VSRTTP: generating change update, sequence number 0002C795  
VSRTTP: Sending update to Broadcast on Hssi0
```

The lines in between the previous two indicate that the router knows how to reach network Router9, which is a metric of 0010 (0.2000 seconds) away from the router. The sequence number for Router9 is zero, and according to the 0x08 bit in the flags field, is invalid. The 0x01 bit of the flags field indicates that Router9 is attached via a LAN interface.

```
network Router9          metric 0010, seq 00000000, flags 09
```

The next lines indicate that the router can reach network RouterZZ, which is a metric of 0230 (7.0000 seconds) away from the router. The sequence number for RouterZZ is 0052194. The 0x02 bit of the flags field indicates that RouterZZ is attached via a WAN interface.

```
network RouterZZ        metric 0230, seq 00052194, flags 02
```

The following line indicates that the router received a routing update from the router LabRouter through the Hssi0 interface:

```
VINESRTTP: received update from LabRouter on Hssi0
```

The following line displays all SRTP values contained in the header of the SRTP packet. This is a type 00 packet, which is a routing update, and the flags field is set to 07, indicating that this is a change update (0x04) and contains both the beginning (0x01) and end (0x02) of the update. This overall update is update number 000E from the router, and this fragment of the update contains the routes beginning at offset 0000 of the update. The sending router's sequence number is currently 00015DFC, and its configured metric for this interface is 0010.

```
update: type 00, flags 07, id 000E, ofst 0000, seq 00015DFC, met 0010
```

The following line implies that the server sending this update is directly accessible to the router (even though VINES servers do not explicitly list themselves in routing updates). Because this is an implicit entry in the table, the other information for this entry is taken from the previous line.

```
network LabRouter from the server
```

As the first actual entry in the routing update from LabRouter, the following line indicates that Router9 can be reached by sending to this server. This network is a metric of 0020 away from the sending server.

```
network Router9          metric 0020, seq 00000000, flags 09
```

debug vines service

Use the **debug vines service EXEC** command to display information on all transactions that occur at the VINES Service (or applications) layer. The **no** form of this command disables debugging output.

[no] debug vines service

Usage Guidelines

You can use the **debug vines service** command to discover why a VINES Service layer process on the router is not communicating with another Service layer process on another router or Banyan server.

Note Because the **debug vines service** command provides the highest level overview of VINES traffic through the router, it is best to begin debugging using this command, and then proceed to use lower-level VINES **debug** commands as necessary.

Sample Display

Figure 2-245 shows sample **debug vines service** output.

Figure 2-245 Sample Debug VINES Service Output

```
router# debug vines service
```

| | |
|---------------------------|---|
| Sent/ Response pair | <pre>VSRV: Get Time Info sent to Townsaver VSRV: Get Time Info response from Townsaver, time: 01:47:54 PDT Apr 29 1993 VSRV: epoch SS@Aloe@Servers-10, age: 0:15:15</pre> |
|---------------------------|---|

932565

As Figure 2-245 suggests, **debug vines service** lines of output appear as activity pairs—either a sent/response pair as shown, or as a received/sent pair.

Table 2-133 describes the fields shown in the second line of output in Figure 2-245. For more information about these fields or their values, refer to Banyan VINES documentation.

Table 2-133 Debug VINES Service Field Descriptions—Part 1

| Field | Description |
|--------------------------------|---|
| VSRV: | Indicates that this is output from the debug vines service command. |
| Get Time Info | Indicates one of three packet types: <ul style="list-style-type: none"> • Get Time Info • Time Set • Time Sync |
| response from | Indicates whether the packet was sent to another router, a response from another router, or received from another router. |
| Townsaver | Indicates the machine name as assigned using the VINES host command, or IP address of the other router. |
| time: 01:47:54 PDT Apr 29 1993 | Indicates the current time in hours:minutes:seconds and current date. |

Table 2-134 describes the fields shown in the third line of output in Figure 2-245. This line is an extension of the first two lines of output. For more information about these fields or their values, refer to Banyan VINES documentation.

Table 2-134 Debug VINES Service Field Descriptions—Part 2

| Field | Description |
|--------------------|--|
| VSRV: | Output from the debug vines service command. |
| epoch | Line of output that describes a VINES epoch. |
| SS@Aloe@Servers-10 | Epoch name. |
| age: 0:15:15 | Epoch—elapsed time since the time was last set in the network. |

debug vines state

Use the **debug vines state** EXEC command to display information on the VINES SRTP state machine transactions. The **no** form of this command disables debugging output.

[no] debug vines state

Usage Guidelines

This command provides a subset of the information provided by the **debug vines routing** command, showing only the transactions made by the SRTP state machine. Refer to the **debug vines routing** command for descriptions of output from the **debug vines state** command.

debug vines table

Use the **debug vines table** EXEC command to display information on all modifications to the VINES routing table. The **no** form of this command disables debugging output.

[no] debug vines table

Usage Guidelines

This command provides a subset of the information produced by the **debug vines routing** command, as well as some more detailed information on table additions and deletions.

Sample Display

Figure 2-246 shows sample **debug vines table** output.

Figure 2-246 Sample Debug VINES Table Output

```
router# debug vines table
VINESRTP: create neighbor 3001153C:8004, interface Ethernet0
```

Table 2-135 describes significant fields shown in Figure 2-246.

Table 2-135 Debug VINES Table Field Descriptions

| Field | Description |
|-------------------------------|---|
| VINESRTP: | Indicates that this is a debug vines routing or debug vines table message. |
| create neighbor 3001153C:8004 | Indicates that the client at address 3001153C:8004 has been added to the Banyan VINES neighbor table. |
| interface Ethernet 0 | Indicates that this neighbor can be reached through the router interface named Ethernet0. |

debug vlan packet

Use the **debug vlan packet** EXEC command to display general information on virtual LAN (VLAN) packets that the router received but is not configured to support. The **no** form of this command disables debugging output.

[no] debug vlan packet

Usage Guidelines

The **debug vlan packet** command displays only packets with a VLAN identifier that the router is not configured to support. This command allows you to identify other VLAN traffic on the network. Virtual LAN packets that the router is configured to route or switch are counted and indicated when you use the **show vlans** command.

Sample Display

Figure 2-247 shows sample **debug vlan packet** output. In this example, a VLAN packet with a VLAN ID of 1000 was received on FDDI 0 interface and this interface was not configured to route or switch this VLAN packet.

Figure 2-247 Sample Debug VLAN Packet Output

```
router# debug vlan packet

vLAN: IEEE 802.10 packet bearing vLAN ID 1000 received on interface
      Fddi0 which is not configured to route/switch ID 1000.
```

debug x25 all

Use the **debug x25 all** EXEC command to display information on all X.25 traffic, including data, control messages, and flow control (RR and RNR) packets. The **no** form of this command disables debugging output.

[no] debug x25 all

Usage Guidelines

This command is particularly useful for diagnosing problems encountered when placing calls.

The **debug x25 all** output includes data, control messages, and flow control packets for all of the router's virtual circuits. The **debug x25 events** and **debug x25 vc** commands provide a subset of this output.



Caution Because **debug x25 all** displays all X.25 traffic, it is processor intensive and can render the router useless. Only use **debug x25 all** when the aggregate of all X.25 traffic is fewer than five packets per second. This command is useful only on low-speed, low-usage links running below 19.2 kbps.

Sample Display

Figure 2-248 shows sample **debug x25 all** output.

Figure 2-248 Sample Debug X25 All Output

```
router# debug x25 all

Serial2: X25 O R3 RESTART (5) 8 lci 0 cause 7 diag 0
Serial2: X25 I R3 RESTART (5) 8 lci 0 cause 0 diag 0
Serial2: X25 I P1 CALL REQUEST (11) 8 lci 1024
From (2): 49 To(2): 46
Facilities: (0)
Call User Data (4): 0xCC 00 00 00 (ip)
Serial2: X25 O P4 CALL CONNECTED (3) 8 lci 1024
Serial2: X25 I P4 DATA (103) 8 lci 1024 PS 0 PR 0
Serial2: X25 O D1 DATA (103) 8 lci 1024 PS 0 PR 1
Serial2: X25 I D1 DATA (103) 8 lci 1024 PS 1 PR 0
Serial2: X25 O D1 DATA (103) 8 lci 1024 PS 1 PR 2
Serial2: X25 I D1 RR (3) 8 lci 1024 PR 2
Serial2: X25 I D1 DATA (103) 8 lci 1024 PS 2 PR 2
Serial2: X25 O D1 DATA (103) 8 lci 1024 PS 2 PR 3
Serial2: X25 I D1 CLEAR REQUEST (5) 8 lci 1024 cause 0 diag 122
Serial2: X25 O D1 CLEAR CONFIRMATION (3) 8 lci 1024
XOT: X25 O D1 PVC-SETUP, waiting to connect (29) <Serial2 pvc 3><Serial2 pvc 1> 2/1 128/64
XOT: X25 I D1 PVC-SETUP, connected (29) <Serial2 pvc 3><Serial2 pvc 1> 2/1 128/64
Serial2: X25 O D1 RESET REQUEST (5) 8 lci 3 cause 15 diag 0
Serial2: X25 I D1 RESET CONFIRMATION (3) 8 lci 3
```

Figure 2-248 shows a typical exchange of packets between two X.25 devices on a network. The first line of output in Figure 2-248 describes a RESTART packet. Table 2-136 describes the fields in this line of output.

Table 2-136 Debug X25 All Field Descriptions

| Field | Description |
|---------|--|
| Serial2 | The interface on which the X.25 event occurred. Events that occur on an X.25-over-TCP connection report XOT. |
| X25 | That this message describes an X.25 event. |
| O | Indication of whether the X.25 message was input (I) or output (O) through the interface. |
| R3 | State of the virtual circuit. Possible values follow: <ul style="list-style-type: none">• D1—Flow control ready• D2—DTE reset request• D3—DCE reset indication• P1—Idle• P2—DTE waiting for DCE to connect CALL• P3—DCE waiting for DTE to accept CALL• P4—Data transfer• P5—CALL collision• P6—DTE clear request• P7—DCE clear indication• R1—Packet level ready• R2—DTE restart request• R3—DCE restart indication• X1—Nonstandard state for a virtual circuit in hold-down See Annex B of the 1984 ITU-T X.25 Recommendation for more information on these states. |

Table 2-136 Debug X25 All Field Descriptions (Continued)

| Field | Description |
|---------|---|
| RESTART | The type of X.25 packet. Possible values follow: <ul style="list-style-type: none"> • CALL CONNECTED • CALL REQUEST • CLEAR CONFIRMATION • CLEAR REQUEST • DATA • DIAGNOSTIC • ILLEGAL • INTR CONFIRMATION • INTR (interrupt) • PVC-SETUP • REGISTRATION • REGISTRATION CONFIRMATION • RESET CONFIRMATION • RESET REQUEST • RESTART • RESTART CONFIRMATION • RNR (Receiver Not Ready) • RR (Receiver Ready) |
| (5) | Number of bytes in the packet. |
| 8 | Modulo of the virtual circuit. Possible values are 8 or 128. |
| lci 0 | Virtual circuit number. See Annex A of the 1984 ITU-T X.25 Recommendation for information on VC assignment. |
| cause 7 | Code indicating the event that triggered the packet. The cause field can only appear in entries for CLEAR REQUEST, RESET REQUEST, and RESTART packets. Possible values for the cause field can vary, depending on the type of packet. Refer to the “X.25 Cause and Diagnostic Codes” appendix for explanations of these codes. |
| diag 0 | Code providing an additional hint as to what, if anything, went wrong. The diag field can only appear in entries for CLEAR REQUEST, DIAGNOSTIC (as “error 0”), RESET REQUEST and RESTART packets. Because of the large number of possible values, they are listed in the “X.25 Cause and Diagnostic Codes” appendix. |

Table 2-137 describes the PS and PR fields that can appear in a **debug x25 all** display.

Table 2-137 Debug X25 All PS and PR Field Descriptions

| Field | Description |
|-------|--|
| PS 0 | Packet send sequence number; used for flow control of the outgoing packet stream. Present only in DATA packets. |
| PR 0 | Packet receive sequence number used for flow control of the incoming packet stream by indicating the PS value that the sender next expects to see. |

In Figure 2-248, notice also that the CALL REQUEST packet precedes three other lines of output that have a unique format.

```
Serial2: X25 I P1 CALL REQUEST (11) 8 lci 1024
From (2): 49 To(2): 46
Facilities: (0)
Call User Data (4): 0xCC 00 00 00 (ip)
Serial2: X25 O P4 CALL CONNECTED (3) 8 lci 1024
```

These lines indicate that the CALL REQUEST packet has a two-digit source address, 49, and a two-digit destination address, 46. These are X.121 addresses that can be from 0 to 15 digits in length. The Facilities field is (0) bytes in length, indicating that no X.25 facilities are being requested. The optional call user data field is 4 bytes in length. Any encapsulation protocol identification (PID) in the Call User Data will have the encoding values printed and identified. Multiprotocol Virtual Circuits can also have PID information in Data packets; the debug output for these packets will also describe the PID.

The two lines of output in Figure 2-248 that begin with XOT are shown below.

```
XOT: X25 O D1 PVC-SETUP, waiting to connect (29) <Serial2 pvc 3><Serial2 pvc 1> 2/1 128/64
XOT: X25 I D1 PVC-SETUP, connected (29) <Serial2 pvc 3><Serial2 pvc 1> 2/1 128/64
```

These lines of output do not describe standard X.25 packets. Instead, they describe messages that represent a tunneled PVC setup between two routers. Table 2-138 describes the fields these two lines of output.

Table 2-138 Debug X25 All Field Descriptions for Tunneled PVC Activity

| Field | Description |
|-------|--|
| XOT | This message travels over a TCP connection. |
| X25 | This message describes an X.25 event. |
| O | Indication of whether the X.25 message was input (I) or output (O) through the connection. |
| D1 | State of the permanent virtual circuit. Possible values follow. <ul style="list-style-type: none"> • D1—Flow control ready • D2—DTE reset request • D3—DCE reset indication See Annex B of the 1984 ITU-T X.25 Recommendation for more information on these states. |

Table 2-138 Debug X25 All Field Descriptions for Tunneled PVC Activity (Continued)

| Field | Description |
|-----------------|---|
| wait to connect | <p>State of the PVC. Some of these strings only apply to PVCs that are remotely tunneled over a TCP connection. The %X25-3-PVCBAD system error message (as documented in the <i>System Error Messages</i> publication), and the show x25 vc command (as documented in the <i>Router Products Command Reference</i> publication) also use these PVC state strings. Possible values follow:</p> <ul style="list-style-type: none"> • awaiting PVC-SETUP reply • can't support flow control values • connected • dest. disconnected • dest. interface is not up • dest. PVC configuration mismatch • mismatched flow control values • no such dest. interface • no such dest. PVC • non-X.25 dest. interface • PVC setup protocol error • PVC/TCP connect timed out • PVC/TCP connection refused • PVC/TCP routing error • trying to connect via TCP • waiting to connect |
| (29) | Incoming/outgoing message size (in bytes). |
| <Serial2 pvc 3> | Interface and PVC number that originated the message (originator). |
| <Serial2 pvc 1> | Interface and PVC number that responded to that message (responder). |
| 2/1 | Window sizes (in packets). |
| 128/64 | Maximum packet sizes (in bytes). |

debug x25 events

Use the **debug x25 events** EXEC command to display information on all X.25 traffic except X.25 data or acknowledgment packets. The **no** form of this command disables debugging output.

[no] debug x25 events

Usage Guidelines

The **debug x25 events** command is useful for debugging X.25 problems, because it shows changes that occur in the virtual circuits handled by the router. Because most X.25 connectivity problems stem from errors that CLEAR or RESET virtual circuits, you can use **debug x25 events** to identify these errors.

While **debug x25 all** output includes both data and control messages for all of the router's virtual circuits, **debug x25 events** output includes only control messages for all of the router's VCs. In contrast, **debug x25 vc** output filters the output for a single VC number. Thus, **debug x25 events** output is a subset of **debug x25 all** output, and **debug x25 vc** output modifies either of them to further limit the output.

Note Because **debug x25 events** displays a subset of all X.25 traffic, it is safer to use than **debug x25 all** during production hours.

Sample Display

Figure 2-249 shows sample **debug x25 events** output.

Figure 2-249 Sample Debug X25 Events Output

```
router# debug x25 events

Serial2: X25 I R3 RESTART (5) 8 lci 0 cause 0 diag 0
Serial2: X25 I P1 CALL REQUEST (11) 8 lci 1024
From (2): 49 To(2): 46
Facilities: (0)
Call User Data (4): 0xCC 00 00 00 (ip)
Serial2: X25 O P4 CALL CONNECTED (3) 8 lci 1024
Serial2: X25 I D1 CLEAR REQUEST (5) 8 lci 1024 cause 0 diag 122
Serial2: X25 O D1 CLEAR CONFIRMATION (3) 8 lci 1024
Serial2: X25 O D1 RESET REQUEST (5) 8 lci 1 cause 0 diag 122
Serial2: X25 I D1 RESET CONFIRMATION (3) 8 lci 1
```

See the **debug x25 all** command description for information on the fields in **debug x25 events** output.

debug x25 vc

Use the **debug x25 vc** EXEC command to display information on traffic for a particular virtual circuit in order to solve any connectivity or performance problems it is exhibiting. The **no** form of this command removes the filter for a particular virtual circuit from the **debug x25 all** or **debug x25 events** output.

[no] debug x25 vc *number*

Syntax Description

number VC number associated with the virtual circuit(s) you want to monitor.

Usage Guidelines

Because no interface is specified, traffic on any VC that has the specified *number* is reported.

The **debug x25 vc** command limits the output of **debug x25 all** or **debug x25 events** output to the packets occurring on a particular VC number. This command modifies the operation of the **debug x25 all** or **debug x25 events** commands, so one of those commands must be used with **debug x25 vc** to produce output.

VC 0 cannot be specified. It is used for X.25 service messages, such as RESTART packets, not VC traffic. VC0 can be monitored only when no VC filter is used.

Note Because **debug x25 vc** only displays traffic for a small subset of virtual circuits, it is safe to use even under heavy traffic conditions, as long as events for that virtual circuit are fewer than 25 packets per second.

Sample Display

Figure 2-250 shows sample **debug x25 vc** output.

Figure 2-250 Sample Debug X25 VC Output

```
router# debug x25 vc 1
X25 debugging output restricted to VC1
router# debug x25 events
X25 special event debugging is on
router# show debug
X.25 (debugging restricted to VC number 1):
  X25 special event debugging is on

Serial0: X25 0 P2 CALL REQUEST (19) 8 lci 1
  From(14): 31250000000101 To(14): 31109090096101
  Facilities (0)
Serial0: X25 I P2 CLEAR REQUEST (5) 8 lci 1 cause diag 122
```

See the **debug x25 all** command description for information on the fields in **debug x25 vc** output.

debug xns packet

Use the **debug xns packet** EXEC command to display information on XNS packet traffic, including the addresses for source, destination, and next hop router of each packet. The **no** form of this command disables debugging output.

[no] debug xns packet

Usage Guidelines

To gain the fullest understanding of XNS routing activity, you should enable **debug xns routing** and **debug xns packet** together.

Sample Display

Figure 2-251 shows sample **debug xns packet** output.

Figure 2-251 Sample Debug XNS Packet Output.

```
router# debug xns packet

XNS: src=5.0000.0c02.6d04, dst=5.ffff.ffff.ffff, packet sent
XNS: src=1.0000.0c00.440f, dst=1.ffff.ffff.ffff, rcvd. on Ethernet0
XNS: src=1.0000.0c00.440f, dst=1.ffff.ffff.ffff, local processing
```

Table 2-139 describes significant fields shown in Figure 2-251.

Table 2-139 Debug XNS Packet Field Descriptions

| Field | Description |
|------------------------|--|
| XNS: | Indicates that this is an XNS packet. |
| src = 5.0000.0c02.6d04 | Indicates that the source address for this message is 0000.0c02.6d04 on network 5. |
| dst = 5.ffff.ffff.ffff | Indicates that the destination address for this message is the broadcast address ffff.ffff.ffff on network 5. |
| packet sent | Indicates that the packet to destination address 5.ffff.ffff.ffff in Figure 2-251, as displayed using the debug xns packet command, was queued on the output interface. |
| rcvd. on Ethernet0 | Indicates that the router just received this packet through the Ethernet0 interface. |
| local processing | Indicates that the router has examined the packet and determined that it must process it, rather than forwarding it. |

debug xns routing

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

[no] debug xns routing

Usage Guidelines

To gain the fullest understanding of XNS routing activity, enable **debug xns routing** and **debug xns packet** together.

Sample Display

Figure 2-252 shows sample **debug xns routing** output.

Figure 2-252 Sample Debug XNS Routing Output

```
router# debug xns routing

XNSRIP: sending standard periodic update to 5.ffff.ffff.ffff via Ethernet2
network 1, hop count 1
network 2, hop count 2

XNSRIP: got standard update from 1.0000.0c00.440f socket 1 via Ethernet0
net 2: 1 hops
```

Table 2-140 describes significant fields shown in Figure 2-252.

Table 2-140 Debug XNS Routing Field Descriptions

| Field | Description |
|--|---|
| XNSRIP: | This is an XNS routing packet. |
| sending standard periodic update to 5.ffff.ffff.ffff via Ethernet2 | Router indicates that this is a periodic XNS routing information update. Destination address is ffff.ffff.ffff on network 5. Name of the output interface. |
| network 1, hop count 1 | Network 1 is one hop away from this router. |
| got standard update from 1.0000.0c00.440f socket 1 | Router indicates that it has received an XNS routing information update from address 0000.0c00.440f on network 1. The socket number is a well-known port for XNS. Possible values include |
| | <ul style="list-style-type: none"> • 1—routing information • 2—echo • 3—router error |