

## 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 2-234 shows sample **debug serial interface** output for an HDLC connection when keepalives are enabled. In Figure 2-234, 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 2-234 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 2-115 describes significant fields shown in Figure 2-234.

**Table 2-115 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 2-116 describes additional error messages that the **debug serial interface** command can generate for HDLC.

**Table 2-116 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>0xnnnnn</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>nnnnnnnnn</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 2-117 describes error messages that the **debug serial interface** command can generate for ISDN Basic Rate.

**Table 2-117 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 2-117 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 2-118 describes the additional error messages that the **debug serial interface** command can generate for an MK5025 device.

**Table 2-118 Debug Serial Interface Err or Messages for an MK5025 Device**

Message	Description
MK5(d): Reset from 0xnnnnnnnn	This message indicates that the hardware has been reset. The 0xnnnnnnnn 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.

```
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 serial packet

Use the **debug serial packet EXEC** command to display more detailed serial interface debugging information than you can obtain using **debug serial interface** command. The **no** form of this command disables debugging output.

**[no] debug serial packet**

### Usage Guidelines

The **debug serial packet** command generates output that is dependent on the type of serial interface and the encapsulation that is running on that interface. The hardware platform also can impact **debug serial packet** output.

The **debug serial packet** command displays output for only SMDS encapsulations.

### Sample Display

Figure 2-235 shows sample output when SMDS is enabled on the interface.

**Figure 2-235 Sample Debug Serial Packet Output for SMDS**

```
Router# debug serial packet

Interface Serial2 Sending SMDS L3 packet:
SMDS Header: Id: 00 RSVD: 00 Bntag: EC Basize: 0044
Dest:E18009999999FFFF Src:C12015804721FFFF Xh:04030000030001000000000000000000
SMDS LLC: AA AA 03 00 00 00 80 38
SMDS Data: E1 19 01 00 00 80 00 00 0C 00 38 1F 00 0A 00 80 00 00 0C 01 2B 71
SMDS Data: 06 01 01 0F 1E 24 00 EC 00 44 00 02 00 00 83 6C 7D 00 00 00 00 00
SMDS Trailer: RSVD: 00 Bntag: EC Length: 0044
```

As Figure 2-235 shows, when encapsulation is set to SMDS, **debug serial packet** displays the entire SMDS header (in hex), as well as some payload data on transmit or receive. This information is useful only when you have an understanding of the SMDS protocol. The first line of the output indicates either Sending or Receiving.

## 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.

---

### Sample Display

Figure 2-236 shows sample **debug service-module** output.

**Figure 2-236 Sample Debug Service-Module Output**

```
Router# debug service-module

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 group** command described in the *Network Protocols Command Reference, Part 2*

### Sample Display

Figure 2-237 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 2-237 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 2-119 lists the messages that may be generated with the **debug smrp group** command concerning the forwarding table.

**Table 2-119 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.
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 2-119 Debug SMRP Group Message Descriptions (Continued)**

<b>Messages</b>	<b>Descriptions</b>
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.
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.
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 2-238 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 2-238 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 2-120 lists all the messages that can be generated with the **debug smrp mcache** command concerning the multicast cache.

**Table 2-120 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 <b>show smrp port appletalk</b> 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 2-239 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 2-239 Sample Debug SMRP Neighbor Output**

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

Table 2-121 lists all the messages that can be generated with the **debug smrp neighbor** command concerning the neighbor table.

**Table 2-121 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.
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.

### 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 2-240 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 2-240 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 2-122 lists all the messages that can be generated with the **debug smrp port** command concerning the port table.

**Table 2-122 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.
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 2-241 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 2-241 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 2-123 lists all the messages that can be generated with the **debug smrp route** command concerning the routing table. In Table 2-123, the term *route* does not refer to an address but rather it is a network range.

**Table 2-123 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 2-123    Debug SMRP Route Message Descriptions (Continued)**

<b>Messages</b>	<b>Descriptions</b>
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.

**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 2-242 shows sample **debug smrp transaction** output. In this example, a secondary node request is sent out to all routers on port 30.1.

**Figure 2-242 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 2-124 lists all the messages that can be generated with the **debug smrp route** command.

**Table 2-124 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 2-243 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 2-243 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 2-125 describes significant fields in these displays.

**Table 2-125 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"> <li>• Get request</li> <li>• Get-next request</li> <li>• Response</li> <li>• Set request</li> <li>• V1 Trap</li> <li>• Get-bulk request</li> <li>• Inform request</li> <li>• V2 Trap</li> </ul> <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 2-125 Debug SNMP Packet Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
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 Displays

Figure 2-244 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 2-244 Sample Debug SNTP Adjust Output—Small Offset from Configured Server**

```
Router# debug sntp adjust  
  
Delay calculated, offset 3.48  
Clock slewed.
```

Figure 2-245 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 2-245 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 2-246 show sample **debug sntp packets** command output when a message is received.

**Figure 2-246 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 2-247 show sample **debug sntp packets** command output when a message is sent.

**Figure 2-247 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 2-126 describes the significant fields shown in Figure 2-246 and Figure 2-247.

**Table 2-126 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 2-126     Debug SNTP Packets Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
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 2-248 show sample **debug sntp select** command output. In this example, the router will synchronize its time to server at 172.16.186.66.

**Figure 2-248 Sample Debug SNTP Select Output**

```
Router# debug sntp select  
  
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 Displays

Figure 2-249 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 2-249 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 2-249 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 2-250 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 2-250 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 2-250 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 2-251 shows sample **debug source event** output.

**Figure 2-251 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 2-127 describes significant fields shown in Figure 2-251.

**Table 2-127 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 2-127 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:

```
Router# debug span
ST: Ether4 0000000000000A080002A02D6700000000000A080002A02D6780010000140002000F00
```

Figure 2-252 shows the preceding **debug span** output broken up by fields and labeled to aid documentation.

**Figure 2-252 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 2-128 describes significant fields shown in Figure 2-252.

**Table 2-128 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.

**Table 2-128 Debug Span Field Descriptions—IEEE BPDU Packet (Continued)**

Field	Description
(K) 01	Port No. 1.
(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:

```
Router# debug span

ST: Ethernet4 E119010000200000C01A2C90064008000000C0106CE0A01050F1E6A
```

Figure 2-253 shows the preceding **debug span** output broken up by fields and labeled to aid documentation.

**Figure 2-253 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 S2576
```

Table 2-129 describes significant fields shown in Figure 2-253.

**Table 2-129 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"> <li>• 0x19—DEC Hello</li> <li>• 0x02—Topology change notification (TCN)</li> </ul>
(C) 01	DEC version.
(D) 00	Flag that is a bit field with the following mapping: <ul style="list-style-type: none"> <li>• 1—TCN</li> <li>• 2—TCN acknowledgment</li> <li>• 8—Use short timers</li> </ul>
(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).

**Table 2-129 Debug Span Field Descriptions for a DEC BPDU Packet (Continued)**

<b>Field</b>	<b>Description</b>
(K) 01	Message age (in seconds).
(L) 05	Hello time (in seconds).
(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 2-254 shows sample **debug sse** output.

**Figure 2-254 Sample Debug SSE Output**

```
Router# debug sse

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 2-254 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 2-255 shows sample **debug standby** output.

**Figure 2-255 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 2-130 describes significant fields shown in Figure 2-255.

**Table 2-130 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 2-130 Debug Standby Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
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 2-255 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

<i>group</i>	(Optional) Decimal integer assigned to a group. Using this option limits output to packets associated with the specified STUN group.
<i>address</i>	(Optional) Output is further limited to only those packets containing the specified STUN address. The <i>address</i> 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 2-256 shows sample **debug stun packet** output.

**Figure 2-256 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
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

```

S2563

Explanations for individual lines of output from Figure 2-256 follow.

The following line describes an X1 type of packet:

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

Table 2-131 describes significant fields shown in this line of **debug stun packet** output.

**Table 2-131 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"> <li>• NDI—Network input</li> <li>• SDI—Serial link</li> </ul>
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"> <li>• I—Information frame</li> <li>• S—Supervisory frame. The possible commands and responses are: RR (Receive Ready), RNR (Receive Not Ready), and REJ (Reject).</li> <li>• 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)</li> </ul>
PF:1	Poll/Final bit. The possible values are as follows: <ul style="list-style-type: none"> <li>• 0—Off</li> <li>• 1—On</li> </ul>

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 Displays

Figure 2-257 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 2-257 Sample Debug AAA Authentication Output—TACACS Login Attempt**

```
Router# debug aaa authentication
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 2-258 shows part of the **debug tacacs** command output for a TACACS login attempt that was successful as indicated by the status PASS.

**Figure 2-258 Sample Debug TACACS Output—Successful Login Attempt**

```
Router# debug tacacs
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 2-259 shows part of the **debug tacacs** command output for a TACACS login attempt that was unsuccessful as indicated by the status FAIL.

**Figure 2-259 Sample Debug TACACS Output—Failed Login Attempt**

```
Router# debug tacacs

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 2-260 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 2-260 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 2-260 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:

```
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 message identifies the following:

- 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

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 2-261 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 2-261 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 2-132 describes the fields shown in Figure 2-261.

**Table 2-132 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 <b>tarp t1-response-timer</b> command.

**Table 2-132 Debug TARP Field Descriptions—TARP Resolve Command (Continued)**

<b>Field</b>	<b>Descriptions</b>
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 2-262 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 2-262 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 2-133 describes the fields shown in the display.

**Table 2-133 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 <b>tarp arp-request-timer</b> 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 2-133 Debug TARP Field Descriptions—TARP Query Command (Continued)**

<b>Field</b>	<b>Descriptions</b>
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 2-263 shows sample **debug tdm** output.

**Figure 2-263 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 2-264 shows sample **debug tftp** output from the EXEC command **copy running-config tftp**.

**Figure 2-264 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 2-134 describes significant fields shown in the first line of output from Figure 2-264.

**Table 2-134 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 2-265 shows sample **debug token ring** output.

**Figure 2-265 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 2-135 describes significant fields shown in the second line of output from Figure 2-265.

**Table 2-135 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 2-135 Debug Token Ring Field Descriptions—Part 1 (Continued)**

Message	Description
MAC:	The type of packet, as follows: <ul style="list-style-type: none"> <li>• MAC—Media Access Control</li> <li>• LLC—Link Level Control</li> </ul>
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 2-136 describes significant fields shown in the third line of output from Figure 2-265.

**Table 2-136 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 2-137 describes significant fields shown in the fifth line of output from Figure 2-265.

**Table 2-137 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"> <li>• MAC—Media Access Control</li> <li>• LLC—Link Level Control</li> </ul>
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).