debug serial interface

To display information on a serial connection failure, use the **debug serial interface** privileged EXEC command. The **no** form of this command disables debugging output.

>**debug serial interface**

>**no debug serial interface**

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   If the **show interface serial** EXEC 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.

⚠

Caution   Although the **debug serial interface** command typically does not generate a substantial amount 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.

Examples   The following sections show and describe sample **debug serial interface** output for various 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

The following is sample output from the **debug serial interface** command for an HDLC connection when keepalives are enabled. This output 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 mineseen fields exceeds three, the line goes down and the interface is reset.

```
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

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

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
```

1 missed keepalive

3 missed keepalives; line goes down and interface is reset

S2561

Table 168 describes the significant fields.

Table 168   debug serial interface Field Descriptions for HDLC

Field	Description
Serial 1	Interface through which the serial connection is taking place.
HDLC	Serial connection is an HDLC connection.
myseq 636119	Myseq counter increases by one each time the router sends a keepalive packet to the remote router.
mineseen 636119	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	Yourseen counter reflects the value of the myseq sequence number the router has received in a keepalive packet from the remote router.
line up	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.

The previous example shows that after missing three keepalives, the line goes down and the interface is reset. However, the interface is also reset when two keepalives are missed, but the line is not marked as "down." This is done in an attempt to restart traffic on the interface without bringing the line down, as shown in the following output:

```
*Mar 18 08:07:29.057: Serial3/2: HDLC myseq 604562, mineseen 604562, yourseen 259336, line up
*Mar 18 08:07:39.053: Serial3/2: HDLC myseq 604563, mineseen 604563, yourseen 259337, line up
*Mar 18 08:07:49.081: Serial3/2: HDLC myseq 604564, mineseen 604564, yourseen 259338, line up
*Mar 18 08:07:59.057: Serial3/2: HDLC myseq 604565, mineseen 604565, yourseen 259339, line up
*Mar 18 08:08:09.073: Serial3/2: HDLC myseq 604566, mineseen 604565, yourseen 259340, line up
*Mar 18 08:08:19.057: Serial3/2: Reset from PC 0x6DEA0
*Mar 18 08:08:19.061: Serial3/2: HDLC myseq 604567, mineseen 604565, yourseen 259341, line up
*Mar 18 08:08:29.057: Serial3/2: HDLC myseq 604568, mineseen 604568, yourseen 259342, line up
*Mar 18 08:08:39.061: Serial3/2: HDLC myseq 604569, mineseen 604569, yourseen 259343, line up
*Mar 18 08:08:49.065: Serial3/2: HDLC myseq 604570, mineseen 604570, yourseen 259344, line up
*Mar 18 08:08:59.053: Serial3/2: HDLC myseq 604571, mineseen 604571, yourseen 259345, line up
```

Even though the "Reset from PC" message appears to occur when there is only a difference of 1 between myseq and mineseen, this message applies to the condition shown in the immediately following line (notice that the timestamp is only a few milliseconds later) where the difference is 2. After the reset, the line has recovered and the difference between myseq and mineseen is zero.

Table 169 describes additional error messages that the **debug serial interface** command can generate for HDLC.

Table 169   debug serial interface Error Messages for HDLC

Field	Description
Illegal serial link type code *<xxx>*, PC = 0x*nnnnnn*	Router attempted to send a packet containing an unknown packet type.
Illegal HDLC serial type code *<xxx>*, PC = 0x*nnnnn*	Unknown packet type is received.
Serial 0: attempting to restart	Interface is down. The hardware is then reset to correct the problem, if possible
Serial 0: Received bridge packet sent to *<nnnnnnnnn>*	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 0x*nnnnnnnn* 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 170 describes error messages that the **debug serial interface** command can generate for ISDN Basic Rate.

Table 170   debug serial interface Error Messages for ISDN Basic Rate

Message	Description
BRI: D-chan collision	Collision on the ISDN D channel has occurred; the software will retry transmission.
Received SID Loss of Frame Alignment int.	ISDN hardware has lost frame alignment. This usually indicates a problem with the ISDN network.
Unexpected IMP int: ipr = 0xnn	ISDN hardware received an unexpected interrupt. The 0xnn variable indicates the value returned by the interrupt register.
BRI(d): RX Frame Length Violation. Length=n 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 0x$nnnnnnn$	BRI hardware has been reset. The 0x$nnnnnnn$ variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers.
BRI(d): Bad state in SCMs scm1=x scm2=x scm3=x BRI(d): Bad state in SCONs scon1=x scon2 =x scon3=x BRI(d): Bad state ub SCR; SCR=x	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=n	Packet is received, but the encapsulation used for the packet is not recognized. The interface might be misconfigured.

Debug Serial Interface for an MK5025 Device

Table 171 describes the additional error messages that the **debug serial interface** command can generate for an MK5025 device.

Table 171   debug serial interface Error Messages for an MK5025 Device

Message	Description
MK5(d): Reset from 0x$nnnnnnnn$	Hardware has been reset. The 0x$nnnnnnnn$ 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=n	Packet is received, but the encapsulation used for the packet is not recognized. Interface might be misconfigured.
MK5(d): No packet available for packet realignment	Serial driver attempted to get a buffer (memory) and was unable to do so.
MK5(d): Bad state in CSR0=(x)	This message is displayed if the hardware is not in the proper state. The hardware is reset. If this message is displayed constantly, it usually indicates a hardware problem.

Table 171   debug serial interface Error Messages for an MK5025 Device (continued)

Message	Description
MK5(d): New serial state=*n*	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, the **debug serial interface** command displays SMDS packets that are sent and received, and 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: dgsize:x 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

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

debug serial packet

no debug serial packet

Syntax Description
This command has no arguments or keywords.

Usage Guidelines
The **debug serial packet** command generates output that is dependent on the type of serial interface and the encapsulation 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.

Examples
The following is sample output from the **debug serial packet** command when SMDS is enabled on the interface:

```
Router# debug serial packet

Interface Serial2 Sending SMDS L3 packet:
SMDS Header: Id: 00 RSVD: 00 BEtag: EC Basize: 0044
Dest:E18009999999FFFF Src:C12015804721FFFF Xh:0403000003000100000000000000000000
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 BEtag: EC Length: 0044
```

As the output shows, when encapsulation is set to SMDS, the **debug serial packet** command displays the entire SMDS header (in hexadecimal notation), and 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

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, use the **debug service-module** privileged EXEC command. The **no** form of this command disables debugging output.

> **debug service-module**

> **no debug service-module**

Syntax Description This command has no arguments or keywords.

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 pre-sent. 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.

Examples The following is sample output from the **debug service-module** command:

```
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 sgbp dial-bids

To display large-scale dial-out negotiations between the primary network access server (NAS) and alternate NASs, use the **debug sgbp dial-bids** privileged EXEC command. The **no** form of this command disables debugging output.

debug sgbp dial-bids

no debug sgbp dial-bids

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   Use this command only when the **sgbp dial-bids** command has been configured.

Examples   The following is sample output from the **debug sgbp dial-bids** command:

```
Router# debug sgbp dial-bids

*Jan  1 00:25:03.643: SGBP-RES:  New bid add request: 4B0 8 2 1 DAC0 1 1
This indicates a new dialout bid has started.
*Jan  1 00:25:03.643: SGBP-RES: Sent Discover message to ID 7B09B71E  49 bytes
The bid request has been sent.
*Jan  1 00:25:03.647: SGBP-RES: Received Message of 49 length:

*Jan  1 00:25:03.647: SGBP-RES: header 5  30  0  31
2 0  0  2D  0  0  0  0  0  0  0  3  0  0  0  1  1E  AF  3A  41  7B  9  B7  1E  8  15  B
3  2  C  6  0  0  DA  C0  D  4  0  0  E  3  1  F  3  1
*Jan  1 00:25:03.647:
*Jan  1 00:25:03.647: SGBP RES: Scan: Message type: Offer
*Jan  1 00:25:03.647: SGBP RES: Scan: Len is 45
*Jan  1 00:25:03.647: SGBP RES: Scan: Transaction ID: 3
*Jan  1 00:25:03.647: SGBP RES: Scan: Message ID: 1
*Jan  1 00:25:03.647: SGBP RES: Scan: Client ID: 1EAF3A41
*Jan  1 00:25:03.651: SGBP RES: Scan: Server ID: 7B09B71E
*Jan  1 00:25:03.651: SGBP RES: Scan: Resource type 8  length 21
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Media type: ISDN
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Min BW: 56000
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Num Links: 0
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port User class: 1
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Priority: 1
*Jan  1 00:25:03.651: SGBP-RES: received 45 length Offer packet
*Jan  1 00:25:03.651: SGBP-RES: Offer from 7B09B71E for Transaction 3 accepted
*Jan  1 00:25:03.651: SGBP RES: Server is uncongested. Immediate win
An alternate network access server has responded and won the bid.
*Jan  1 00:25:03.651: SGBP-RES: Bid Succeeded  handle 7B09B71E  Server-id 4B0
*Jan  1 00:25:03.651: SGBP-RES: Sent Dial-Req message to ID 7B09B71E  66 bytes
The primary network access server has asked the alternate server to dial.
*Jan  1 00:25:04.651: SGBP-RES: QScan: Purging entry
*Jan  1 00:25:04.651: SGBP-RES: deleting entry 6112E204 1EAF3A41 from list...
```

debug sgbp error

To enable the display of debug messages about routing problems between members of a stack group, use the **debug sgbp error** command in privileged EXEC mode. To disable debug messages about routing problems between members of a stack group, use the **no** form of this command.

> **debug sgbp error**

> **no debug sgbp error**

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Command History

Release	Modification
11.2(9)	This command was introduced.

Usage Guidelines Enable the **debug sgbp error** command to enable the display of debug messages about routing problems between members of a stack group.

Note In unusual cases you may see debug messages not documented on this command reference page. These debug messages are intended for expert diagnostic interpretation by the Cisco Technical Assistance Center (TAC).

Examples One common configuration error is setting a source IP address for a stack member that does not match the locally defined IP address for the same stack member. The following debug output shows the error message that results from this misconfiguration:

```
Systema# debug sgbp error

%SGBP-7-DIFFERENT - systemb's addr 10.1.1.2 is different from hello's addr 10.3.4.5
```

This error means that the source IP address of the Stack Group Bidding Protocol (SGBP) hello message received from systemb does not match the IP address configured locally for systemb (through the **sgbp member** command). Correct this configuaration error by going to systemb and checking for multiple interfaces by which the SGBP hello can send the message.

Another common error message is:

```
Systema# debug sgbp error

%SGBP-7-MISCONF, Possible misconfigured member routerk (10.1.1.6)
```

This error message means that routerk is not defined locally, but is defined on another stack member. Correct this configuration error by defining routerk across all members of the stack group using the **sgbp member** command.

■ **debug sgbp error**

The following error message indicates that an SGBP peer is leaving the stack group:

```
Systema# debug sgbp error

%SGBP-7-LEAVING:Member systemc leaving group stack1
```

This error message indicates that the peer systemc is leaving the stack group. Systemc could be leaving the stack group intentionally, or a connectivity problem may exist.

The following error message indicates that an SGBP event was detected from an unknown peer:

```
Systema# debug sgbp error

%SGBP-7-UNKNOWPEER:Event 0x10 from peer at 172.21.54.3
```

An SGBP event came from a network host that was not recognizable as an SGBP peer. Check to see if a network media error could have corrupted the address, or if peer equipment is malfunctioning to generate corrupted packets. Depending on the network topology and firewalling of your network, SGBP packets from a nonpeer host could indicate probing and attempts to breach security.

⚠
Caution If there is a chance your network is under attack, obtain knowledgeable assistance from TAC.

Related Commands	Command	Description
	debug sgbp hellos	Enables the display of debug messages for authentication between stack members.
	sgbp group	Defines a named stack group and makes this router a member of that stack group.
	sgbp member	Specifies the hostname and IP address of a router or access server that is a peer member of a stack group.
	show sgbp	Displays the status of the stack group members.
	username	Establishes a username-based authentication system.

debug sgbp hellos

To enable the display of debug messages for authentication between stack group members, use the **debug sgbp hellos** command in privileged EXEC mode. To disable debug messages about authentication between stack group members, use the **no** form of this command

debug sgbp hellos

no debug sgbp hellos

Syntax Description   This command has no arguments or keywords.

Command Modes   Privileged EXEC

Command History

Release	Modification
11.2(9)	This command was introduced.

Usage Guidelines   Enable the **debug sgbp hellos** command to enable the display of debug messages for authentication between routers configured as members of a stack group.

✎

Note   In unusual cases you may see debug messages not documented on this command reference page. These debug messages are intended for expert diagnostic interpretation by the Cisco Technical Assistance Center (TAC).

Examples   The following output from the **debug sgbp hellos** command shows systema sending a successful Challenge Handshake Authentication Protocol (CHAP) challenge to and receiving a response from systemb. Similarly, systemb sends out a challenge and receives a response from systema:

```
systema# debug sgbp hellos

%SGBP-7-CHALLENGE: Send Hello Challenge to systemb group stack1
%SGBP-7-CHALLENGED: Hello Challenge message from member systemb (10.1.1.2)
%SGBP-7-RESPONSE: Send Hello Response to systemb group stack1
%SGBP-7-CHALLENGE: Send Hello Challenge to systemb group stack1
%SGBP-7-RESPONDED: Hello Response message from member systemb (10.1.1.2)
%SGBP-7-AUTHOK: Send Hello Authentication OK to member systemb (10.1.1.2)
%SGBP-7-INFO: Addr = 10.1.1.2 Reference = 0xC347DF7
%SGBP-5-ARRIVING: New peer event for member systemb
```

This debug output is self-explanitory.

If authentication fails, you may see one of the following messages in your debug output:

```
%SGBP-7-AUTHFAILED - Member systemb failed authentication
```

This error message means that the remote systemb password for the stack group does not match the password defined on systema. To correct this error, make sure that both systema and systemb have the same password defined using the **username** command.

```
%SGBP-7-NORESP -Fail to respond to systemb group stack1, may not have password.
```

This error message means that systema does not have a username or password defined. To correct this error, define a common group password across all stack members using the **username** command.

Related Commands	Command	Description
	debug sgbp error	Enables the display of debug messages about routing problems between members of a stack group.
	sgbp group	Defines a named stack group and makes this router a member of that stack group.
	sgbp member	Specifies the hostname and IP address of a router or access server that is a peer member of a stack group.
	show sgbp	Displays the status of the stack group members.
	username	Establishes a username-based authentication system.

debug sgcp

To debug the Simple Gateway Control Protocol (SGCP), use the **debug sgcp** privileged EXEC command. To turn off debugging, use the **no** form of the command.

debug sgcp {**errors** | **events** | **packet**}

no debug sgcp {**errors** | **events** | **packet**}

Syntax Description

errors	Displays debug information about SGCP errors.
events	Displays debug information about SGCP events.
packet	Displays debug information about SGCP packets.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable access router.

Examples

See the following examples to enable and disable debugging at the specified level:

```
Router# debug sgcp errors

Simple Gateway Control Protocol errors debugging is on

Router# no debug sgcp errors

Simple Gateway Control Protocol errors debugging is off
Router#

Router# debug sgcp events

Simple Gateway Control Protocol events debugging is on
Router# no debug sgcp events

Simple Gateway Control Protocol events debugging is off
Router#

Router# debug sgcp packet

Simple Gateway Control Protocol packets debugging is on
Router# no debug sgcp packet

Simple Gateway Control Protocol packets debugging is off
Router#
```

Related Commands

Command	Description
sgcp	Starts and allocates resources for the SCGP daemon.

debug sgcp errors

To debug Simple Gateway Control Protocol (SGCP) errors, use the **debug sgcp errors** EXEC command. Use the **no** form of this command to turn off debugging.

debug sgcp errors [**endpoint** *string*]

no debug sgcp errors

Syntax Description	**endpoint** *string*	(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint.
		On the Cisco MC3810 router, the endpoint string syntax takes the following forms:
		• DS1 endpoint: **DS1-***slot*/*port*
		• POTS endpoint: **aaln**/*slot*/*port*
		On the Cisco 3600 router, the endpoint string syntax takes the following forms:
		• DS1 endpoint: *slot*/*subunit*/**DS1-***ds1 number*/*ds0 number*
		• POTS endpoint: **aaln**/*slot*/*subunit*/*port*

Defaults   No default behavior or values.

Command Modes   EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced on the Cisco AS5300 access server in a private release not generally available.
12.0(7)XK	Support for this command was extended to the Cisco MC3810 and the Cisco 3600 series routers (except for the Cisco 3620) in a private release that was not generally available. Also, the **endpoint** keyword was added.

Examples   The following example shows the debugging of SGCP errors being enabled:

```
Router# debug sgcp errors

Simple Gateway Control Protocol errors debugging is on
no errors since call went through successfully.
```
The following example shows a debug trace for SGCP errors on a specific endpoint:

```
Router# debug sgcp errors endpoint DS1-0/1

End point name for error debug:DS1-0/1 (1)
00:08:41:DS1 = 0, DS0 = 1
```

```
00:08:41:Call record found
00:08:41:Enable error end point debug for (DS1-0/1)
```

Related Commands	Command	Description
	debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
	debug rtpspi errors	Debugs RTP SPI errors.
	debug rtpspi inout	Debugs RTP SPI in/out functions.
	debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
	debug sgcp events	Debugs SGCP events.
	debug sgcp packet	Debugs SGCP packets.
	debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug sgcp events

To debug Simple Gateway Control Protocol (SGCP) events, use the **debug sgcp events** EXEC command. Use the **no debug sgcp events** command to turn off debugging.

> **debug sgcp events** [**endpoint** *string*]

> **no debug sgcp events**

Syntax Description

endpoint *string*	(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint.
	On the Cisco MC3810 router, the endpoint string syntax takes the following forms:
	• DS1 endpoint: **DS1-***slot*/*port*
	• POTS endpoint: **aaln**/*slot*/*port*
	On the Cisco 3600 router, the endpoint string syntax takes the following forms:
	• DS1 endpoint: *slot*/*subunit*/**DS1-***ds1 number*/*ds0 number*
	• POTS endpoint: **aaln**/*slot*/*subunit*/*port*

Defaults   No default behavior or values.

Command Modes   EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced on the Cisco AS5300 access server in a private release not generally available.
12.0(7)XK	Support for this command was extended to the Cisco MC3810 and the Cisco 3600 series routers (except for the Cisco 3620 router) in a private release that was not generally available. Also, the **endpoint** keyword was added.

Examples   The following example shows a debug trace for SGCP events on a specific endpoint:

```
Router# debug sgcp events endpoint DS1-0/1

End point name for event debug:DS1-0/1 (1)
00:08:54:DS1 = 0, DS0 = 1
00:08:54:Call record found
00:08:54:Enable event end point debug for (DS1-0/1)
```

The following example shows a debug trace for all SGCP events on a gateway:

```
Router# debug sgcp events

*Mar  1 01:13:31.035:callp :19196BC, state :0, call ID :-1, event :23

*Mar  1 01:13:31.035:voice_if->call_agent_ipaddr used as Notify entityNotify entity
available for Tx SGCP msg
NTFY send to ipaddr=1092E01 port=2427
*Mar  1 01:13:31.039:Push msg into SGCP wait ack queue* (1)[25]
*Mar  1 01:13:31.039:Timed Out interval [1]:(2000)
*Mar  1 01:13:31.039:Timed Out interval [1]:(2000)(0):E[25]
*Mar  1 01:13:31.075:Removing msg :
NTFY 25 ds1-1/13@mc1 SGCP 1.1
X:358258758
O:hd


*Mar  1 01:13:31.075:Unqueue msg from SGCP wait ack q** (0)[25]DS1 = 1, DS0 = 13

*Mar  1 01:13:31.091:callp :19196BC, vdbptr :1964EEC, state :1
*Mar  1 01:13:31.091:Checking ack (trans ID 237740140) :

*Mar  1 01:13:31.091:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:31.091:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                    event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                    event=0x20000004, event2=0xC
*Mar  1 01:13:31.091:Same digit map is download (ds1-1/13@mc1)

*Mar  1 01:13:31.091:R:requested trans_id (237740140)

*Mar  1 01:13:31.091:process_signal_ev:seizure possible=1, signal mask=0x4, mask2=0x0
*Mar  1 01:13:32.405:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.489:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:32.610:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.670:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:32.766:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.810:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:32.931:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.967:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.087:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.132:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.240:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.280:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.389:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.433:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.537:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.581:callp :19196BC, state :1, call ID :16, event :9
```

```
*Mar  1 01:13:33.702:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.742:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.742:voice_if->call_agent_ipaddr used as Notify entityNotify entity
available for Tx SGCP msg
NTFY send to ipaddr=1092E01 port=2427
*Mar  1 01:13:33.742:Push msg into SGCP wait ack queue* (1)[26]
*Mar  1 01:13:33.742:Timed Out interval [1]:(2000)
*Mar  1 01:13:33.742:Timed Out interval [1]:(2000)(0):E[26]
*Mar  1 01:13:33.786:Removing msg :
NTFY 26 ds1-1/13@mc1 SGCP 1.1
X:440842371
O:k0, 4081037, s0


*Mar  1 01:13:33.786:Unqueue msg from SGCP wait ack q** (0)[26]DS1 = 1, DS0 = 13

*Mar  1 01:13:33.802:callp :19196BC, vdbptr :1964EEC, state :1
*Mar  1 01:13:33.802:Checking ack (trans ID 698549528) :

*Mar  1 01:13:33.802:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:33.802:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                     event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                     event=0x4, event2=0x0
*Mar  1 01:13:33.802:R:requested trans_id (698549528)

*Mar  1 01:13:33.802:set_up_voip_call_leg:peer_addr=0, peer_port=0.
*Mar  1 01:13:33.806:call_setting_crcx:Enter CallProceeding state rc = 0, call_id=16

*Mar  1 01:13:33.806:callp :19196BC, state :4, call ID :16, event :31

*Mar  1 01:13:33.810:callp :1AF5798, state :2, call ID :17, event :8
call_pre_bridge!

*Mar  1 01:13:33.810:send_oc_create_ack:seizure_possiblle=1, ack-lready-sent=0, ack_send=0
*Mar  1 01:13:33.814:callp :1AF5798, state :4, call ID :17, event :28

*Mar  1 01:13:33.814:Call Connect:Raw Msg ptr=0x1995360, no-offhook=0; call-id=17
*Mar  1 01:13:33.814:SGCP Session Appl:ignore CCAPI event 37

*Mar  1 01:13:33.947:callp :19196BC, state :5, call ID :16, event :32
process_nse_on_orig
DS1 = 1, DS0 = 13

*Mar  1 01:13:34.007:callp :19196BC, vdbptr :1964EEC, state :5
*Mar  1 01:13:34.007:Checking ack (trans ID 123764791) :

*Mar  1 01:13:34.007:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:34.007:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                     event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                     event=0x4, event2=0x0
*Mar  1 01:13:34.007:R:requested trans_id (123764791)

*Mar  1 01:13:34.007:process_signal_ev:seizure possible=1, signal mask=0x0, mask2=0x0
*Mar  1 01:13:34.007:modify_connection:echo_cancel=1.
*Mar  1 01:13:34.007:modify_connection:vad=0.
*Mar  1 01:13:34.007:modify_connection:peer_addr=6000001, peer_port=0->16500.
*Mar  1 01:13:34.007:modify_connection:conn_mode=2.
*Mar  1 01:13:34.011:callp :19196BC, state :5, call ID :16, event :31
```

```
*Mar  1 01:13:34.011:callp :1AF5798, state :5, call ID :17, event :31
process_nse_event

*Mar  1 01:13:34.051:callp :19196BC, state :5, call ID :16, event :39

*Mar  1 01:13:34.051:call_id=16, ignore_ccapi_ev:ignore 19 for state 5
DS1 = 1, DS0 = 13

*Mar  1 01:13:39.497:callp :19196BC, vdbptr :1964EEC, state :5
*Mar  1 01:13:39.497:Checking ack (trans ID 553892443) :

*Mar  1 01:13:39.497:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:39.497:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                     event=0x6003421F, event2=0x3FD
requested signal=0x8, signal2=0x0,
                     event=0x4, event2=0x0
*Mar  1 01:13:39.497:R:requested trans_id (553892443)

*Mar  1 01:13:39.497:process_signal_ev:seizure possible=1, signal mask=0x0, mask2=0x0
*Mar  1 01:13:39.497:modify_connection:echo_cancel=1.
*Mar  1 01:13:39.497:modify_connection:vad=0.
*Mar  1 01:13:39.497:modify_connection:peer_addr=6000001, peer_port=16500->16500.
*Mar  1 01:13:39.497:modify_connection:conn_mode=3.
*Mar  1 01:13:39.497:callp :19196BC, state :5, call ID :16, event :31

*Mar  1 01:13:39.501:callp :1AF5798, state :5, call ID :17, event :31

*Mar  1 01:14:01.168:Removing ack (trans ID 237740140) :
 200 237740140 OK


*Mar  1 01:14:03.883:Removing ack (trans ID 698549528) :
 200 698549528 OK
I:7

v=0
c=IN IP4 5.0.0.1
m=audio 16400 RTP/AVP 0


*Mar  1 01:14:04.087:Removing ack (trans ID 123764791) :
 200 123764791 OK
I:7

v=0
c=IN IP4 5.0.0.1
m=audio 16400 RTP/AVP 0


*Mar  1 01:14:09.573:Removing ack (trans ID 553892443) :
 200 553892443 OK
I:7

v=0
c=IN IP4 5.0.0.1
m=audio 16400 RTP/AVP 0


*Mar  1 01:14:48.091:callp :19196BC, state :5, call ID :16, event :12

*Mar  1 01:14:48.091:voice_if->call_agent_ipaddr used as Notify entityNotify entity
available for Tx SGCP msg
NTFY send to ipaddr=1092E01 port=2427
*Mar  1 01:14:48.091:Push msg into SGCP wait ack queue* (1)[27]
```

```
*Mar  1 01:14:48.091:Timed Out interval [1]:(2000)
*Mar  1 01:14:48.091:Timed Out interval [1]:(2000)(0):E[27]
*Mar  1 01:14:48.128:Removing msg :
NTFY 27 ds1-1/13@mc1 SGCP 1.1
X:97849341
O:hu


*Mar  1 01:14:48.128:Unqueue msg from SGCP wait ack q** (0)[27]DS1 = 1, DS0 = 13

*Mar  1 01:14:48.212:callp :19196BC, vdbptr :1964EEC, state :5
*Mar  1 01:14:48.212:Checking ack (trans ID 79307869) :

*Mar  1 01:14:48.212:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:14:48.212:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                     event=0x6003421F, event2=0x3FD
requested signal=0x4, signal2=0x0,
                     event=0x0, event2=0x0
*Mar  1 01:14:48.212:delete_call:callp:19196BC, call ID:16
*Mar  1 01:14:48.212:sgcp delete_call:Setting disconnect_by_dlcx to 1
*Mar  1 01:14:48.216:callp :1AF5798, state :6, call ID :17, event :29

*Mar  1 01:14:48.216:Call disconnect:Raw Msg ptr = 0x0, call-id=17
*Mar  1 01:14:48.216:disconnect_call_leg O.K. call_id=17
*Mar  1 01:14:48.216:SGCP:Call disconnect:No need to send onhook
*Mar  1 01:14:48.216:Call disconnect:Raw Msg ptr = 0x19953B0, call-id=16
*Mar  1 01:14:48.216:disconnect_call_leg O.K. call_id=16
*Mar  1 01:14:48.220:callp :1AF5798, state :7, call ID :17, event :13

*Mar  1 01:14:48.220:Processing DLCX signal request :4, 0, 0

*Mar  1 01:14:48.220:call_disconnected:call_id=17, peer 16 is not idle yet.DS1 = 1, DS0 =
13

*Mar  1 01:14:48.272:callp :19196BC, vdbptr :1964EEC, state :7
*Mar  1 01:14:48.272:Checking ack (trans ID 75540355) :

*Mar  1 01:14:48.272:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:14:48.272:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                     event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                     event=0x8, event2=0x0
*Mar  1 01:14:48.272:R:requested trans_id (75540355)

*Mar  1 01:14:48.272:process_signal_ev:seizure possible=1, signal mask=0x4, mask2=0x0
*Mar  1 01:14:49.043:callp :19196BC, state :7, call ID :16, event :27

*Mar  1 01:14:49.043:process_call_feature:Onhook event
*Mar  1 01:14:49.043:callp :19196BC, state :7, call ID :16, event :13

*Mar  1 01:15:18.288:Removing ack (trans ID 79307869) :
 250 79307869 OK


*Mar  1 01:15:18.344:Removing ack (trans ID 75540355) :
 200 75540355 OK
```

Related Commands	Command	Description
	debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
	debug rtpspi errors	Debugs RTP SPI errors.
	debug rtpspi inout	Debugs RTP SPI in/out functions.
	debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
	debug sgcp errors	Debugs SGCP errors.
	debug sgcp packet	Debugs SGCP packets.
	debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug sgcp packet

To debug the Simple Gateway Control Protocol (SGCP), use the **debug sgcp packet** EXEC command. Use the **no debug sgcp packet** command to turn off debugging.

> **debug sgcp packet** [**endpoint** *string*]

> **no debug sgcp packet**

Syntax Description	**endpoint** *string*	(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint.
		On the Cisco MC3810, the endpoint string syntax takes the following forms:
		• DS1 endpoint: **DS1-**slot/port
		• POTS endpoint: **aaln/**slot/port
		On the Cisco 3600, the endpoint string syntax takes the following forms:
		• DS1 endpoint: slot/subunit/**DS1-**ds1 number/ds0 number
		• POTS endpoint: **aaln/**slot/subunit/port

Defaults   No default behavior or values.

Command Modes   EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced on the Cisco AS5300 in a private release not generally available.
12.0(7)XK	Support for this command was extended to the Cisco MC3810 and the Cisco 3600 series routers (except for the Cisco 3620) in a private release that was not generally available. Also, the **endpoint** keyword was added

Examples   The following example shows a debug trace for SGCP packets on a specific endpoint:

```
Router# debug sgcp packet endpoint DS1-0/1

End point name for packet debug:DS1-0/1 (1)
00:08:14:DS1 = 0, DS0 = 1
00:08:14:Enable packet end point debug for (DS1-0/1)
```
The following example shows a debug trace for all SGCP packets on a gateway:

```
Router# debug sgcp packet

*Mar  1 01:07:45.204:SUCCESS:Request ID string building is OK
```

```
*Mar  1 01:07:45.204:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:45.204:SUCCESS:SGCP message building OK
*Mar  1 01:07:45.204:SUCCESS:END of building
*Mar  1 01:07:45.204:SGCP Packet sent --->
NTFY 22 ds1-1/13@mc1 SGCP 1.1
X:550092018
O:hd

<---

*Mar  1 01:07:45.204:NTFY Packet sent successfully.
*Mar  1 01:07:45.240:Packet received -

200 22


*Mar  1 01:07:45.244:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:45.244:SUCCESS:END of Parsing
*Mar  1 01:07:45.256:Packet received -

RQNT 180932866 ds1-1/13@mc1 SGCP 1.1
X:362716780
R:hu,k0(A),s0(N),[0-9T](A) (D)
D:(9xx|xxxxxxx)


*Mar  1 01:07:45.256:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:45.256:SUCCESS:Request ID string(362716780) parsing is OK
*Mar  1 01:07:45.260:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:45.260:SUCCESS:Digit Map parsing is OK
*Mar  1 01:07:45.260:SUCCESS:END of Parsing
*Mar  1 01:07:45.260:SUCCESS:SGCP message building OK
*Mar  1 01:07:45.260:SUCCESS:END of building
*Mar  1 01:07:45.260:SGCP Packet sent --->
200 180932866 OK

<---

*Mar  1 01:07:47.915:SUCCESS:Request ID string building is OK
*Mar  1 01:07:47.915:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:47.919:SUCCESS:SGCP message building OK
*Mar  1 01:07:47.919:SUCCESS:END of building
*Mar  1 01:07:47.919:SGCP Packet sent --->
NTFY 23 ds1-1/13@mc1 SGCP 1.1
X:362716780
O:k0, 4081037, s0

<---

*Mar  1 01:07:47.919:NTFY Packet sent successfully.
*Mar  1 01:07:47.955:Packet received -

200 23


*Mar  1 01:07:47.955:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:47.955:SUCCESS:END of Parsing
*Mar  1 01:07:47.971:Packet received -

CRCX 938694984 ds1-1/13@mc1 SGCP 1.1
M:recvonly
L:p:10,e:on,s:off, a:G.711u
R:hu
C:6
```

```
*Mar  1 01:07:47.971:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:47.971:SUCCESS:Connection Mode parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Packet period parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Echo Cancellation parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Silence Supression parsing is OK
*Mar  1 01:07:47.971:SUCCESS:CODEC strings parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Local Connection option parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:47.975:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:07:47.975:SUCCESS:END of Parsing
*Mar  1 01:07:47.979:SUCCESS:Conn ID string building is OK
*Mar  1 01:07:47.979:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:47.979:SUCCESS:SGCP message building OK
*Mar  1 01:07:47.979:SUCCESS:END of building
*Mar  1 01:07:47.979:SGCP Packet sent --->
200 938694984 OK
I:6

v=0
c=IN IP4 5.0.0.1
m=audio 16538 RTP/AVP 0

<---

*Mar  1 01:07:48.188:Packet received -

MDCX 779665338 ds1-1/13@mc1 SGCP 1.1
I:6
M:recvonly
L:p:10,e:on,s:off,a:G.711u
R:hu
C:6

v=0
c=IN IP4 6.0.0.1
m=audio 16392 RTP/AVP 0


*Mar  1 01:07:48.188:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:48.188:SUCCESS:Conn ID string(6) parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Connection Mode parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Packet period parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Echo Cancellation parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Silence Supression parsing is OK
*Mar  1 01:07:48.192:SUCCESS:CODEC strings parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Local Connection option parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:07:48.192:SUCCESS:SDP Protocol version parsing OK
*Mar  1 01:07:48.192:SUCCESS:SDP Conn Data OK
*Mar  1 01:07:48.192:SUCCESS:END of Parsing
*Mar  1 01:07:48.200:SUCCESS:Conn ID string building is OK
*Mar  1 01:07:48.200:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:48.200:SUCCESS:SGCP message building OK
*Mar  1 01:07:48.200:SUCCESS:END of building
*Mar  1 01:07:48.200:SGCP Packet sent --->
200 779665338 OK
I:6

v=0
c=IN IP4 5.0.0.1
m=audio 16538 RTP/AVP 0
```

```
<---

*Mar  1 01:07:53.674:Packet received -

MDCX 177780432 ds1-1/13@mc1 SGCP 1.1
I:6
M:sendrecv
X:519556004
L:p:10,e:on, s:off,a:G.711u
C:6
R:hu
S:hd

v=0
c=IN IP4 6.0.0.1
m=audio 16392 RTP/AVP 0


*Mar  1 01:07:53.674:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:53.674:SUCCESS:Conn ID string(6) parsing is OK
*Mar  1 01:07:53.674:SUCCESS:Connection Mode parsing is OK
*Mar  1 01:07:53.674:SUCCESS:Request ID string(519556004) parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Packet period parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Echo Cancellation parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Silence Supression parsing is OK
*Mar  1 01:07:53.678:SUCCESS:CODEC strings parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Local Connection option parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Signal Requests parsing is OK
*Mar  1 01:07:53.678:SUCCESS:SDP Protocol version parsing OK
*Mar  1 01:07:53.678:SUCCESS:SDP Conn Data OK
*Mar  1 01:07:53.678:SUCCESS:END of Parsing
*Mar  1 01:07:53.682:SUCCESS:Conn ID string building is OK
*Mar  1 01:07:53.682:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:53.682:SUCCESS:SGCP message building OK
*Mar  1 01:07:53.682:SUCCESS:END of building
*Mar  1 01:07:53.682:SGCP Packet sent --->
200 177780432 OK
I:6

v=0
c=IN IP4 5.0.0.1
m=audio 16538 RTP/AVP 0

<---

*Mar  1 01:09:02.401:SUCCESS:Request ID string building is OK
*Mar  1 01:09:02.401:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:09:02.401:SUCCESS:SGCP message building OK
*Mar  1 01:09:02.401:SUCCESS:END of building
*Mar  1 01:09:02.401:SGCP Packet sent --->
NTFY 24 ds1-1/13@mc1 SGCP 1.1
X:519556004
O:hu

<---

*Mar  1 01:09:02.401:NTFY Packet sent successfully.
*Mar  1 01:09:02.437:Packet received -

200 24
```

```
*Mar  1 01:09:02.441:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:09:02.441:SUCCESS:END of Parsing
*Mar  1 01:09:02.541:Packet received -

DLCX 865375036 ds1-1/13@mc1 SGCP 1.1
C:6
S:hu


*Mar  1 01:09:02.541:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:09:02.541:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:09:02.541:SUCCESS:Signal Requests parsing is OK
*Mar  1 01:09:02.541:SUCCESS:END of Parsing
*Mar  1 01:09:02.545:SUCCESS:SGCP message building OK
*Mar  1 01:09:02.545:SUCCESS:END of building
*Mar  1 01:09:02.545:SGCP Packet sent --->
250 865375036 OK


<---

*Mar  1 01:09:02.577:Packet received -

RQNT 254959796 ds1-1/13@mc1 SGCP 1.1
X:358258758
R:hd


*Mar  1 01:09:02.577:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:09:02.577:SUCCESS:Request ID string(358258758) parsing is OK
*Mar  1 01:09:02.577:SUCCESS:Requested Event parsing is OK
*Mar  1 01:09:02.581:SUCCESS:END of Parsing
*Mar  1 01:09:02.581:SUCCESS:SGCP message building OK
*Mar  1 01:09:02.581:SUCCESS:END of building
*Mar  1 01:09:02.581:SGCP Packet sent --->
200 254959796 OK
```

Command History	Command	Description
	debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
	debug rtpspi errors	Debugs RTP SPI errors.
	debug rtpspi inout	Debugs RTP SPI in/out functions.
	debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
	debug sgcp errors	Debugs SGCP errors.
	debug sgcp events	Debugs SGCP events.
	debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug smrp all

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

debug smrp all

no debug smrp all

Syntax Description   This command has no arguments or keywords.

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 learn if packets are sent and received and which specific routes are affected. The **show smrp traffic** EXEC 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

Command	Description
debug smrp group	Displays information about SMRP group activity.
debug smrp mcache	Displays information about SMRP multicast fast-switching cache entries.
debug smrp neighbor	Displays information about SMRP neighbor activity.
debug smrp port	Displays information about SMRP port activity.
debug smrp route	Displays information about SMRP routing activity.
debug smrp transaction	Displays information about SMRP transactions.

debug smrp group

To display information about SMRP group activity, use the **debug smrp group** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp group

no debug smrp group

Syntax Description   This command has no arguments or keywords.

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 *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Examples   The following is sample output from the **debug smrp group** command showing a port being created and deleted on group AT 20.34. (AT signifies that this is an AppleTalk network group.)

```
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 172 lists the messages that may be generated with the **debug smrp group** command concerning the forwarding table.

Table 172   debug smrp group Message Descriptions

Messages	Descriptions
Group *<address>*, deleted on port *<address>*	Group entry was deleted from the group table for the specified port.
Group *<address>*, forward state changed from *state* to *state*	State of the group changed. States are join, forward, and leave.
Group *<address>*, deleted forward entry	Group was deleted from the forwarding table.
Group *<address>*, created on port *<address>* by *<address>*	Group entry was created in the table for the specified port.
Group *<address>*, added by *<address>* to the group	Secondary router has added this group to its group table.
Group *<address>*, discard join request from *<address>*, 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 *<address>*, join request from *<address>*	Request to join the group was received.

Table 172   debug smrp group Message Descriptions (continued)

Messages	Descriptions
Group *<address>*, forward is found	Forward entry for the group was found in the forwarding table.
Group *<address>*, forward state is already joining, ignored	Request to join the group is in progress, so the second request was discarded.
Group *<address>*, no forward found	Forward entry for the group was not found in the forwarding table.
Group *<address>*, 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 *<address>*, created forward entry - parent *<address>* child *<address>*	Forward entry was created in the forwarding table for the parent and child address.
Group *<address>*, creator no longer up on *<address>*	Group creator has not been heard from for a specified time and is deemed no longer available.
Group *<address>*, pruning duplicate path on *<address>*	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 *<address>*, member no longer up on *<address>*	Group member has not been heard from for a specified time and is deemed no longer available.
Group *<address>*, 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 Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp mcache

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

debug smrp mcache

no debug smrp mcache

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   Use the **show smrp mcache** EXEC command (described in the *Cisco IOS AppleTalk and Novell IPX Command Reference* to display the entries in the SMRP multicast cache, and use the **debug smrp mcache** command to learn whether the cache is being populated and invalidated.

Examples   The following is sample output from the **debug smrp mcache** command. In this example, the cache is created and populated for group AT 11.124. (AT signifies that this is an AppleTalk network group.)

```
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 173 lists all the messages that can be generated with the **debug smrp mcache** command concerning the multicast cache.

Table 173   debug smrp mcache Message Descriptions

Messages	Descriptions
Cache populated for group *<address>*	SMRP packet was received on a parent port that has fast switching enabled. As a result, the cache was created and the MAC and network headers were stored for all child ports that have fast switching enabled. Use the **show smrp port appletalk** EXEC 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 *<address>*	Forward cache entry for the group was added to or deleted from the cache.
Forward cache entry validated for group *<address>*	Forward cache entry is validated and is now ready for fast switching.
Forward cache entry invalidated for group *<address>*	Cache entry is invalidated because some change (such as port was shut down) occurred to one of the ports.

Related Commands	Command	Description
	debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp neighbor

To display information about SMRP neighbor activity, use the **debug smrp neighbor** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp neighbor

no debug smrp neighbor

Syntax Description   This command has no arguments or keywords.

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** EXEC command described in the *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Examples   The following is sample output from the **debug smrp neighbor** command. In this example, the neighbor on port 30.02 has changed state from normal operation to secondary operation.

```
Router# debug smrp neighbor

SMRP: Neighbor 30.2, state changed from "normal op" to "secondary op"
```

Table 174 lists all the messages that can be generated with the **debug smrp neighbor** command concerning the neighbor table.

Table 174   debug smrp neighbor Message Descriptions

Messages	Descriptions
Neighbor *<address>*, state changed from *state* to *state*	State of the neighbor changed. States are primary operation, secondary operation, normal operation, primary negotiation, secondary negotiation, and down.
Neighbor *<address>*, 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 *<address>*, no longer up	Neighbor is unavailable because it has not been heard from for a specified duration.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp port

To display information about SMRP port activity, use the **debug smrp port** privileged EXEC command. The **no** form of this command disables debugging output.

> **debug smrp port**

> **no debug smrp port**

Syntax Description   This command has no arguments or keywords.

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 *Cisco IOS AppleTalk and Novell IPX Command Reference*.

Examples   The following is sample output from the **debug smrp port** command. In this example, port 30.1 has changed state from secondary negative to secondary operation to primary negative:

```
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 175 lists all the messages that can be generated with the **debug smrp port** command concerning the port table.

Table 175   debug smrp port Message Descriptions

Messages	Descriptions
Port *<address>*, port created/deleted	Port entry was added to or removed from the port table.
Port *<address>*, line protocol changed to *state*	Line protocol for the port is up or down.
Port *<address>*, state changed from *state* to *state*	State of the port changed. States are primary operation, secondary operation, normal operation, primary negotiation, secondary negotiation, and down.
Port *<address>*, primary/secondary router changed from *<address>* to *<address>*	Primary or secondary port address of the router changed.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp route

To display information about SMRP routing activity, use the **debug smrp route** privileged EXEC command. The **no** form of this command disables debugging output.

> **debug smrp route**

> **no debug smrp route**

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   For more information, refer to the **show smrp route** EXEC command described in the *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Examples   The following is sample output from the **debug smrp route** command. In this example, poison notification is received from port 30.2. Poison notification is the receipt of a poisoned route on a nonparent port.

```
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 176 lists all the messages that can be generated with the **debug smrp route** command concerning the routing table. In Table 176, the term *route* does not refer to an address but rather to a network range.

Table 176   debug smrp route Message Descriptions

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

Table 176   debug smrp route Message Descriptions (continued)

Messages	Descriptions
Route address, improved via address - number -> number hop, number -> number tunnel	Distance to the route has improved (become lower), received from a neighbor.
Route address, switched to address - higher address than address	Tie condition exists, and because this router had the highest network address, it was used to forward the packet.
Route address, parent port changed address -> address	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 address, has been poisoned	Route has been poisoned. Poisoned routes are purged from the routing table after a specified time.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp transaction

To display information about SMRP transactions, use the **debug smrp transaction** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp transaction

no debug smrp transaction

Syntax Description　This command has no arguments or keywords.

Examples　The following is sample output from the **debug smrp transaction** command. In this example, a secondary node request is sent out to all routers on port 30.1.

```
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 177 lists all the messages that can be generated with the **debug smrp route** command.

Table 177　debug smrp Transaction Message Descriptions

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

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug snasw dlc

To display frame information entering and leaving the SNA switch in real time to the console, use the **debug snasw dlc** privileged EXEC command.

> **debug snasw dlc detail**

Syntax Description

detail	Indicates that in addition to a one-line description of the frame being displayed, an entire hexadecimal dump of the frame will follow.

Defaults

By default, a one-line description of the frame is displayed.

⚠️

Caution The **debug snasw dlc** command displays the same trace information available via the **snasw dlctrace** command. The **snasw dlctrace** command is the preferred method for gathering this trace information because it is written to a capture buffer instead of directly to the console. The **debug snasw dlc** command should only be used when it is certain that the output will not cause excessive data to be output to the console.

Command History

Release	Modification
12.0(6)T	This command was introduced.

Examples

The following is an example of the **debug snasw dlc** command output:

```
Router# debug snasw dlc

Sequence
Number          Size of ISR/
     Link       SNA BTU HPR  Description of frame

343   MVSD    In  sz:134  ISR fmh5 DLUR Rq ActPU NETA.APPNRA29
344   MVSD    Out sz:12   ISR +Rsp IPM     slctd nws:0008
345   @I000002 Out sz:18   ISR Rq ActPU
346   MVSD    Out sz:273  ISR fmh5 TOPOLOGY UPDATE
347   @I000002 In  sz:9    ISR +Rsp Data
348   @I000002 In  sz:12   ISR +Rsp IPM     slctd nws:0002
349   @I000002 In  sz:29   ISR +Rsp ActPU
350   MVSD    Out sz:115  ISR fmh5 DLUR +Rsp ActPU
351   MVSD    In  sz:12   ISR +Rsp IPM     slctd nws:0007
352   MVSD    In  sz:88   ISR fmh5 DLUR Rq ActLU NETA.MARTLU1
353   MVSD    Out sz:108  ISR fmh5 REGISTER
354   @I000002 Out sz:27   ISR Rq ActLU NETA.MARTLU1
```

Related Commands	Command	Description
	snasw dlctrace	Captures trace frames entering and leaving the SNA switching Services feature.
	snasw dlcfilter	Filters frames traced by the **snasw dlctrace** or **debug snasw dlc** command.

debug snasw ips

To display internal signal information between the SNA switch and the console in real time, use the **debug snasw ips** privileged EXEC command.

debug snasw dlc

Syntax Description This command has no arguments or keywords.

Defaults By default, a one-line description of the interprocess signal is displayed.

⚠

Caution The **debug snasw ips** command displays the same trace information available via the **snasw ipstrace** command. Output from this **debug** command can be large. The **snasw ipstrace** command is the preferred method for gathering this trace information because it is written to a capture buffer instead of directly to the console. The **debug snasw ips** command should only be used when it is certain that the output will not cause excessive data to be output to the console. The **debug snasw dlc** command displays the same trace information available via the **snasw dlctrace** command.

Command History

Release	Modification
12.0(6)T	This command was introduced.

Examples The following is an example of the **debug snasw ips** command output:

```
Router# debug snasw ips

Sequence
Number              Sending     Receiving
        Signal Name Process     Process     Queue

11257 : DEALLOCATE_RCB : --(0) -> RM(2130000) Q 4
11258 : RCB_DEALLOCATED : RM(2130000) -> PS(22E0000) Q 2
11259 : RCB_DEALLOCATED : --(0) -> PS(22E0000) Q 2
11260 : VERB_SIGNAL : PS(22E0000) -> DR(20F0000) Q 2
11261 : FREE_SESSION : --(0) -> RM(2130000) Q 2
11262 : BRACKET_FREED : RM(2130000) -> HS(22FB0001) Q 2
11263 : BRACKET_FREED : --(0) -> HS(22FB0001) Q 2
11264 : VERB_SIGNAL : --(0) -> DR(20F0000) Q 2
11265 : DLC_MU : DLC(2340000) -> PC(22DD0001) Q 2
11266 : DLC_MU : --(0) -> PC(22DD0001) Q 2
```

Related Commands	Command	Description
	snasw ipstrace	Captures interprocess signal information between Switching Services components.

debug snmp packet

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

debug snmp packet

no debug snmp packet

Syntax Description   This command has no arguments or keywords.

Examples   The following is 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.

```
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 178 describes the significant fields shown in the display.

Table 178   debug snmp packet Field Descriptions

Field	Description
Get-next request	Indicates what type of SNMP PDU the packet is. Possible types are as follows: • Get request • Get-next request • Response • Set request • V1 Trap • Get-bulk request • Inform request • V2 Trap 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.
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 second varbind in the request PDU, the response PDU will have an erridx equal to 2.
nonrptr	Nonrepeater 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 nonrepeater 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 snmp requests

To display information about every SNMP request made by the SNMP manager, use the **debug snmp requests** privileged EXEC command. The **no** form of this command disables debugging output.

debug snmp requests

no debug snmp requests

Syntax Description   This command has no arguments or keywords.

Examples   The following is sample output from the **debug snmp requests** command:

```
Router# debug snmp requests

SNMP Manager API: request
  dest: 171.69.58.33.161, community: public
  retries: 3, timeout: 30, mult: 2, use session rtt
  userdata: 0x0
```

Table 179 describes the significant fields shown in the display.

Table 179   debug snmp requests field Field Descriptions

Field	Description
SNMP Manager API	Indicates that the router sent an SNMP request.
dest	Destination of the request.
community	Community string sent with the request.
retries	Number of times the request has been re-sent.
timeout	Request timeout, or how long the router will wait before resending the request.
mult	Timeout multiplier. The timeout for a re-sent request will be equal to the previous timeout multiplied by the timeout multiplier.
use session rtt	Indicates that the average round-trip time of the session should be used in calculating the timeout value.
userdata	Internal Cisco IOS software data.

debug sntp adjust

To display information about SNTP clock adjustments, use the **debug sntp adjust** privileged EXEC command. The **no** form of this command disables debugging output.

debug sntp adjust

no debug sntp adjust

Syntax Description   This command has no arguments or keywords.

Examples   The following is sample output from the **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.

```
Router# debug sntp adjust

Delay calculated, offset 3.48
Clock slewed.
```

The following is sample output from the **debug sntp adjust** command when an offset to the time reported by a broadcast server is calculated. Because 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.

```
Router# debug sntp adjust

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

debug sntp packets

To display information about SNTP packets sent and received, use the **debug sntp packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug sntp packets

no debug sntp packets

Syntax Description   This command has no arguments or keywords.

Examples   The following is sample output from the **debug sntp packets** command when a message 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)
```

The following is sample output from the **debug sntp packets** command when a message 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 180 describes the significant fields shown in the display.

Table 180   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 that the router is currently using for synchronization.
ref	Reference time stamp.
org	Originate time stamp. This value indicates the time the request was sent by the router.

Table 180   debug sntp packets Field Descriptions (continued)

Field	Description
rec	Receive time stamp. This value indicates the time the request was received by the SNTP server.
xmt	Transmit time stamp. This value indicates the time the reply was sent by the SNTP server.
inp	Destination time stamp. This value indicates the time the reply was received by the router.

debug sntp select

To display information about SNTP server selection, use the **debug sntp select** privileged EXEC command. The **no** form of this command disables debugging output.

debug sntp select

no debug sntp select

Syntax Description   This command has no arguments or keywords.

Examples   The following is sample output from the show sample **debug sntp select** command. In this example, the router will synchronize its time to the server at 172.16.186.66.

```
Router# debug sntp select

SNTP: Selected 172.16.186.66
```

debug source bridge

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

debug source bridge

no debug source bridge

Syntax Description   This command has no arguments or keywords.

Examples   The following is sample output from the **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.

```
Router# debug source bridge

RSRB: remote explorer to 5/192.108.250.1/1996 srn 2 [C840.0021.0050.0000]
RSRB: Version/Ring XReq sent to peer 5/192.108.250.1/1996
RSRB: Received version reply from 5/192.108.250.1/1996 (version 2)
RSRB: DATA: 5/192.108.250.1/1996 Ring Xchg Rep, trn 2, vrn 5, off 18, len 10
RSRB: added bridge 1, ring 1 for 5/192.108.240.1/1996
RSRB: DATA: 5/192.108.250.1/1996 Explorer trn 2, vrn 5, off 18, len 69
RSRB: DATA: 5/192.108.250.1/1996 Forward trn 2, vrn 5, off 0, len 92
RSRB: DATA: forward Forward srn 2, br 1, vrn 5 to peer 5/192.108.250.1/1996
```

The following line indicates that a remote explorer frame has been sent to IP address 192.108.250.1 and, like all RSRB TCP connections, has been assigned port 1996. The bridge belongs to ring group 5. The explorer frame originated from ring 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 5/192.108.250.1/1996 srn 2 [C840.0021.0050.0000]
```

The following line indicates that a request for remote peer information has been sent to IP address 192.108.250.1, TCP port 1996. The bridge belongs to ring group 5.

```
RSRB: Version/Ring XReq sent to peer 5/192.108.250.1/1996
```

The following line is the response to the version request previously sent. The response is sent from IP address 192.108.250.1, TCP port 1996. The bridge belongs to ring group 5.

```
RSRB: Received version reply from 5/192.108.250.1/1996 (version 2)
```

The following line is the response to the ring request previously sent. The response is sent from IP address 192.108.250.1, 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: 5/192.108.250.1/1996 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 192.108.250.1, TCP port 1996:

```
RSRB: added bridge 1, ring 1 for 5/192.108.250.1/1996
```

The following line indicates that a packet containing an explorer frame came across virtual ring 5 from IP address 192.108.250.1, 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: 5/192.108.250.1/1996 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 192.108.250.1 over TCP port 1996. The packet is being placed on the local target ring 2. The packet is 92 bytes in length.

```
RSRB: DATA: 5/192.108.250.1/1996 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 address 192.108.250.1 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 5/192.108.250.1/1996
```

The following is sample output from the **debug source bridge** command 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.

```
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
```

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 1. The 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

To display source-route bridging (SRB) errors, use the **debug source error** privileged EXEC command. The **no** form of this command disables debugging output.

debug source error

no debug source error

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   The debug source error command displays some output also found in the **debug source bridge** output. See the **debug source bridge** command for other possible output.

Examples   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. SRTLB indicates a source-route translational bridging (SR/TLB) 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 TCP queue of the peer router 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 (RII) 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 (BPDUs). BPDUs are exchanged to set up the spanning tree and determine the forwarding path.

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

Related Commands	Command	Description
	debug source bridge	Displays information about packets and frames transferred across a source-route bridge.
	debug source event	Displays information on SRB activity.

debug source event

To display information on source-route bridging activity, use the **debug source event** privileged EXEC command. The **no** form of this command disables debugging output.

debug source event

no debug source event

Syntax Description This command has no arguments or keywords.

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.

Examples The following is sample output from the **debug source event** command:

```
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 181 describes the significant fields shown in the display.

Table 181   debug source event Field Descriptions

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

Table 181   debug source event Field Descriptions (continued)

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

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 the address could not be reached 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 RSRB code found that 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 pre-sent 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

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

> **debug span**

> **no debug span**

Syntax Description   This command has no arguments or keywords.

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

Examples   The following is sample output from the **debug span** command for an IEEE BPDU packet:

```
Router# debug span

ST: Ether4 0000000000000A080002A02D6700000000000A080002A02D6780010000140002000F00
```

The following is sample output from the **debug span** command:

```
ST: Ether4 0000000000000A080002A02D6700000000000A080002A02D6780010000140002000F00
           A  BCDE  F         G     H  I           JKL  M  N  O
```

Table 182 describes the significant fields shown in the display.

Table 182   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.

Table 182   debug span Field Descriptions—IEEE BPDU Packet (continued)

Field	Description
(I) 080002A02D67	Bridge ID.
(J) 80	Port priority.
(K) 01	Port Number 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).

The following is sample output from the **debug span** command for a DEC BPDU packet:

```
Router# debug span

ST: Ethernet4 E1190100000200000C01A2C90064008000000C0106CE0A01050F1E6A
```

The following is sample output from the **debug span** command:

```
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
```

Table 183 describes the significant fields.

Table 183   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: • 0x19—DEC Hello • 0x02—TCN
(C) 01	DEC version.
(D) 00	Flag that is a bit field with the following mapping: • 1—TCN • 2—TCN acknowledgment • 8—Use short timers
(E) 0002	Root priority.
(F) 00000C01A2C9	Root ID (MAC address).
(G) 0064	Root path cost (translated as 100 in decimal notation).
(H) 0080	Bridge priority.
(I) 00000C0106CE	Bridge ID.
(J) 0A	Port ID (in contrast to interface number).

Table 183   debug span Field Descriptions for a DEC BPDU Packet (continued)

Field	Description
(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

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

debug sse

no debug sse

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   Use the **debug sse** command to display statistics and counters maintained by the SSE.

Examples   The following is sample output from the **debug sse** command:

```
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
```

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

To display Hot Standby Router Protocol (HSRP) state changes, use the **debug standby** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug standby [**terse**]

no debug standby [**terse**]

Syntax Description	terse	(Optional) Displays a limited range of HSRP errors, events, and packets.

Command Modes Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.

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.

Examples The following is sample output from the **debug standby** command:

```
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 184 describes the significant fields shown in the display.

Table 184   debug standby Field Descriptions

Field	Description
SB	Abbreviation for "standby."
Ethernet0	Interface on which a Hot Standby packet was sent or received.
Hello in	Hello packet received from the specified IP address.
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	Hold-down interval advertised in the hello packet.
ip *address*	Hot Standby group IP address advertised in the hello packet.
state	Transition from one state to another.
Coup out *address*	Coup packet sent by the router from the specified IP address.

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
```

Related Commands	Command	Description
	debug condition standby	Filters the output of the **debug standby** command on the basis of HSRP group number.
	debug standby errors	Displays error messages related to HSRP.
	debug standby events	Displays events related to HSRP.
	debug standby events icmp	Displays debugging messages for the HSRP ICMP redirects filter.
	debug standby packets	Displays debugging information for packets related to HSRP.

debug standby errors

To display error messages related to Host Standby Router Protocol (HSRP), use the **debug standby errors** command in privileged EXEC mode. To disable the display of these messages, use the **no** form of this command.

> **debug standby errors**

> **no debug standby errors**

Syntax Description   This command has no arguments or keywords

Defaults   Debugging is not enabled.

Command Modes   Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.

Usage Guidelines   You can filter the **debug** output using interface and HSRP group conditional debugging. To enable interface conditional debugging, use the **debug condition interface** command. To enable HSRP conditional debugging, use the **debug condition standby** command.

Examples   The following example enables the display of HSRP errors:

```
debug standby errors
```

Related Commands

Command	Description
debug standby events icmp	Displays HSRP errors.
debug standby events	Displays HSRP events

debug standby events

To display events related to Hot Standby Router Protocol (HSRP), use the **debug standby events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

> **debug standby events** [**all** | **api** | **ha** | **internal** | **protocol** | **redundancy** | **terse** | **track**] [**detail**]

> **no debug standby events** [**all** | **ha** | **internal** | **protocol** | **redundancy** | **terse** | **track**] [**detail**]

Syntax Description		
	all	(Optional) All HSRP events.
	api	(Optional) HSRP application programming interface (API) events.
	ha	(Optional) High availability (HA) events.
	internal	(Optional) Internal HSRP events.
	protocol	(Optional) HSRP protocol events.
	redundancy	(Optional) HSRP redundancy events.
	terse	(Optional) Specifies all HSRP packets, except hellos and advertisements.
	track	(Optional) HSRP tracking events.
	detail	(Optional) Detailed debugging information.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.1	This command was introduced.
	12.2(8)T	The **api** keyword was added.
	12.4(4)T	The **ha** keyword was added.
	12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
	12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
	12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines You can filter the debug output using interface and HSRP group conditional debugging. To enable interface conditional debugging, use the **debug condition interface** command. To enable HSRP conditional debugging, use the **debug condition standby** command.

Examples The following example shows how to enable the debugging of the active and standby Route Processors (RPs) on an active RP console. The HSRP group is configured on the active RP, and the HSRP state is active.

```
Router# debug standby events ha

!Active RP

*Apr 27 04:13:47.755: HSRP: Et0/0/1 Grp 101 RF Encode state Listen into sync buffer
*Apr 27 04:13:47.855: HSRP: CF Sync send ok
```

```
*Apr 27 04:13:57.755: HSRP: Et0/0/1 Grp 101 RF Encode state Speak into sync buffer
*Apr 27 04:13:57.855: HSRP: CF Sync send ok
*Apr 27 04:14:07.755: HSRP: Et0/0/1 Grp 101 RF Encode state Standby into sync buffer
*Apr 27 04:14:07.755: HSRP: Et0/0/1 Grp 101 RF Encode state Active into sync buffer
*Apr 27 04:14:07.863: HSRP: CF Sync send ok
*Apr 27 04:14:07.867: HSRP: CF Sync send ok
!Standby RP

*Apr 27 04:11:21.011: HSRP: RF CF client 32, entity 0 got msg len 24
*Apr 27 04:11:21.011: HSRP: Et0/0/1 Grp 101 RF sync state Init -> Listen
*Apr 27 04:11:31.011: HSRP: RF CF client 32, entity 0 got msg len 24
*Apr 27 04:11:31.011: HSRP: Et0/0/1 Grp 101 RF sync state Listen -> Speak
*Apr 27 04:11:41.071: HSRP: RF CF client 32, entity 0 got msg len 24
*Apr 27 04:11:41.071: HSRP: RF CF client 32, entity 0 got msg len 24
*Apr 27 04:11:41.071: HSRP: Et0/0/1 Grp 101 RF sync state Speak -> Standby
*Apr 27 04:11:41.071: HSRP: Et0/0/1 Grp 101 RF sync state Standby -> Active
```

Table 185 describes the significant fields shown in the display.

Table 185   debug standby events Field Descriptions

Field	Description
RF	Redundancy facility—Internal mechanism that makes Stateful Switchover (SSO) work.
CF	Checkpoint facility—Internal mechanism that makes SSO work.

Related Commands

Command	Description
debug condition interface	Limits output for some debug commands on the basis of the interface, VC, or VLAN.
debug condition standby	Filters the output of the **debug standby** command on the basis of HSRP group number.
debug standby	Displays HSRP state changes.
debug standby errors	Displays error messages related to HSRP.
debug standby events icmp	Displays debugging messages for the HSRP ICMP redirects filter.
debug standby packets	Displays debugging information for packets related to HSRP.

debug standby events icmp

To display debug messages for the Hot Standby Router Protocol (HSRP) Internet Control Message Protocol (ICMP) redirects filter, use the **debug standby events icmp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

> **debug standby events icmp**

> **no debug standby events icmp**

Syntax Description   This command has no arguments or keywords.

Command Modes   Privileged EXEC

Command History

Release	Modification
12.1(3)T	This command was introduced.

Usage Guidelines   This command helps you determine whether HSRP is filtering an outgoing ICMP redirect message.

Examples   The following is sample output from the **debug standby events icmp** command:

```
Router# debug standby events icmp

10:35:20: SB: changing ICMP redirect sent to 20.0.0.4 for dest 30.0.0.2
10:35:20: SB:   gw 20.0.0.2 -> 20.0.0.12, src 20.0.0.11
10:35:20: SB: Use HSRP virtual address 20.0.0.11 as ICMP src
```

If the router being redirected to is passive (HSRP enabled but no active groups), the following debug message is displayed:

```
10:41:22: SB: ICMP redirect not sent to 20.0.0.4 for dest 40.0.0.3
10:41:22: SB:   20.0.0.3 does not contain an active HSRP group
```

If HSRP could not uniquely determine the gateway used by the host, then the following message is displayed:

```
10:43:08: SB: ICMP redirect not sent to 20.0.0.4 for dest 30.0.0.2
10:43:08: SB: could not uniquely determine IP address for mac 00d0.bbd3.bc22
```

The following messages are also displayed if the **debug ip icmp** command is enabled, in which case the message prefix is changed:

```
10:39:09: ICMP: HSRP changing redirect sent to 20.0.0.4 for dest 30.0.0.2
10:39:09: ICMP:   gw 20.0.0.2 -> 20.0.0.12, src 20.0.0.11
10:39:09: ICMP: Use HSRP virtual address 20.0.0.11 as ICMP src
10:39:09: ICMP: redirect sent to 20.0.0.4 for dest 30.0.0.2, use gw 20.0.0.12
```

Related Commands

Command	Description
debug ip icmp	Displays information on ICMP transactions.

debug standby packets

To display debugging information for packets related to Hot Standby Router Protocol (HSRP), use the **debug standby packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

> **debug standby packets** [**advertise** | **all** | **terse** | **coup** | **hello** | **resign**] [**detail**]

> **no debug standby packet** [**advertise** | **all** | **terse** | **coup** | **hello** | **resign**] [**detail**]

Syntax Description

advertise	(Optional) Specifies HSRP advertisement packets.
all	(Optional) Specifies all HSRP packets.
terse	(Optional) Specifies all HSRP packets, except hellos and advertisements.
coup	(Optional) Specifies HSRP coup packets.
hello	(Optional) Specifies HSRP hello packets.
resign	(Optional) Specifies HSRP resign packets.
detail	(Optional) Specifies HSRP packets in detail.

Defaults

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.
12.2	The **advertise** keyword was added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

You can filter the debug output using interface and HSRP group conditional debugging. To enable interface conditional debugging, use the **debug condition interface** command. To enable HSRP conditional debugging, use the **debug condition standby** command.

Note HSRP advertisement packets are packets that are related to HSRP interfaces. Other packet types, including, hello, coup, and resign packets relate to an HSRP group.

Examples

The following example show how to enable the display of all HSRP packets:

```
Router# debug standby packets all

HSRP Packets debugging is on.
```

Related Commands	Command	Description
	debug condition interface	Limits output for some debugging commands based on the interfaces.
	debug condition standby	Filters the output of the **debug standby** command on the basis of HSRP group number.
	debug standby	Displays HSRP state changes.
	debug standby errors	Displays error messages related to HSRP.
	debug standby events	Displays events related to HSRP.
	debug standby events icmp	Displays debugging messages for the HSRP ICMP redirects filter.

debug stun packet

To display information on packets traveling through the serial tunnel (STUN) links, use the **debug stun packet** privileged EXEC command. The **no** form of this command disables debugging output.

> **debug stun packet** [*group*] [*address*]

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

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

Usage Guidelines   Because using this command is processor intensive, it is best to use it after regular business 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.

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

```
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
STUN sdlc: 0:00:01 Serial3        SDI: (0C2/008) U: UA       PF:1
```
X2 type of packet
```
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

The following line describes an X1 type of packet:

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

Table 186 describes the significant fields in this line of **debug stun packet** output.

Table 186   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 the previous packet.
Serial3	Interface type and unit number reporting the event.
NDI:	Type of cloud separating the SDLC end nodes. Possible values are as follows: • NDI—Network input • SDI—Serial link
0C2	SDLC address of the SDLC connection.
008	Modulo value of 8.
U: SNRM	Frame type followed by the command or response type. In this case it is an Unnumbered frame that contains a Set Normal Response Mode (SNRM) command. The possible frame types are as follows: • I—Information frame • S—Supervisory frame. The possible commands and responses are: RR (Receive Ready), RNR (Receive Not Ready), and REJ (Reject). • U—Unnumbered frame. The possible commands are: UI (Unnumbered Information), SNRM, DISC/RD (Disconnect/Request Disconnect), SIM/RIM, XID Exchange Identification), TEST. The possible responses are UA (unnumbered acknowledgment), DM (Disconnected Mode), and FRMR (Frame Reject Mode)
PF:1	Poll/Final bit. Possible values are as follows: • 0—Off • 1—On

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

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

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

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

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

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

debug sw56

To display debug information for switched 56K services, use the **debug sw56** privileged EXEC command.

debug sw56

Syntax Description   This command has no arguments or keywords.

debug syscon perfdata

To display messages related to performance data collection, use the **debug syscon perfdata** privileged EXEC command. The **no** form of this command disables debugging output.

debug syscon perfdata

no debug syscon perfdata

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is primarily useful to your technical support representative.

Examples

The following is sample output from the **debug syscon perfdata** command. In this example, the CallFail poll group is configured and applied to shelf 1111. The system determines when the next polling cycle should occur and polls the shelf at the appropriate time. The data is stored in the file CallFail.891645120, and an older file is deleted.

```
Router# debug syscon perfdata

PERF: Applying 'CallFail' to shelf 1111
PERF: Setting up objects for SNMP polling: 'CallFail', shelf 1111
PERF: year hours mins secs msecs = 1998 15 11 1 5
PERF: Start 'CallFail' timer, next cycle in 0 mins, 59 secs
PERF: Timer event:  CallFail, 4 minutes
PERF: Polling 'CallFail', shelf 1111, pc 60AEFDF0
PERF: SNMP resp: Type 6, 'CallFail', shelf 1111, error_st 0
PERF: Logged polled data to disk0:/performance/shelf-1111/CallFail.891645120
PERF: Deleted disk0:/performance/shelf-1111/CallFail.891637469
```

debug syscon sdp

To display messages related to the Shelf Discovery Protocol (SDP), use the **debug syscon sdp** privileged EXEC command. The **no** form of this command disables debugging output.

debug syscon sdp

no debug syscon sdp

Syntax Description   This command has no arguments or keywords.

Usage Guidelines   Use this command to display information about SDP packets exchanged between the shelf and the system controller.

Examples   The following sample output from the **debug syscon sdp** command shows the system controller discovering a managed shelf. In the first few lines, the system controller receives a hello packet from shelf 99 at 172.23.66.106. The system controller responds with a hello packet. When the shelf sends another hello packet, the system controller resets the timer and sends another packet.

```
Syscon# debug syscon sdp

SYSCTLR: Hello packet received via UDP from 172.23.66.106
%SYSCTLR-6-SHELF_ADD: Shelf 99 discovered located at address 172.23.66.106
Hello packet sent to the RS located at 172.23.66.106
SYSCTLR: Hello packet received via UDP from 172.23.66.106
Timer for shelf 99 updated, shelf is alive
Hello packet sent to the RS located at 172.23.66.106
```

The following sample output from the **debug syscon sdp** command shows the shelf contacting the system controller. The shelf sends a hello packet to the system controller at 172.23.66.111. The system controller responds with the autoconfiguration commands. The remaining lines show the Hello packets were exchanged between the shelf and the system controller.

```
Shelf# debug syscon sdp

SYSCTLR: Hello packet sent to the SYSCTLR at 172.23.66.111
SYSCTLR: Command packet received from SYSCTLR
Feb 24 17:24:16.713: %SHELF-6-SYSCTLR_ESTABLISHED: Configured via system controller
located at 172.23.66.111
SYSCTLR: Rcvd HELLO from SYSCTLR at 172.23.66.111
SYSCTLR: Hello packet sent to the SYSCTLR at 172.23.66.111
SYSCTLR: Rcvd HELLO from SYSCTLR at 172.23.66.111
```

debug syslog-server

To display information about the syslog server process, use the **debug syslog-server** privileged EXEC command. The **no** form of this command disables debugging output.

> **debug syslog-server**

> **no debug syslog-server**

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command outputs a message every time the syslog server receives a message. It also displays information about subfile creation, removal, and renaming.

Use this command when subfiles are not being created as configured or data is not being written to subfiles. This command is also useful for detecting syslog file size mismatches.

Examples

The following sample display shows output when the following command has been added to the configuration:

```
logging syslog-server 10 3 syslogs
```

This example shows the files being created. Use the **dir disk0:/syslogs.dir** command to display the contents of the newly created directory.

```
Router# debug syslog-server

SYSLOG_SERVER:Syslog file syslogs
SYSLOG_SERVER:Directory disk0:/syslogs.dir created.
SYSLOG_SERVER:Syslog file syslogs created successfully.
```

When a syslog message is received, the router checks to determine if the current file will be too large when the new data is added. In this example, two messages are added to the file.

```
SYSLOG_SERVER: Configured size : 10240 bytes
Current size : 0 bytes
Data size : 68 bytes
New size : 68 bytes
SYSLOG_SERVER: Wrote 68 bytes successfully.
SYSLOG_SERVER: Configured size : 10240 bytes
Current size : 68 bytes
Data size : 61 bytes
New size : 129 bytes
SYSLOG_SERVER: Wrote 61 bytes successfully.
```

Table 187 describes the significant fields shown in the display.

Table 187   debug syslog-server Field Descriptions

Field	Description
Configured size	Maximum subfile size, as set in the **logging syslog-server** command.
Current size	Size of the current subfile before the new message is added.

Table 187   debug syslog-server Field Descriptions (continued)

Field	Description
Data size	Size of the syslog message.
New size	Size of the current subfile after the syslog message is added.

The following output indicates that the current file is too full to fit the next syslog message. The oldest subfile is removed, and the remaining files are renamed. A new file is created and opened for writing syslog messages.

```
SYSLOG_SERVER:Last archive subfile disk0:/syslogs.dir/syslogs.2 removed.
SYSLOG_SERVER: Subfile disk0:/syslogs.dir/syslogs.1 renamed as
disk0:/syslogs.dir/syslogs.2.
SYSLOG_SERVER:subfile disk0:/syslogs.dir/syslogs.cur renamed as
disk0:/syslogs.dir/syslogs.1.
SYSLOG_SERVER:Current subfile disk0:/syslogs.dir/syslogs.cur has been opened.
```

debug tacacs

To display information associated with the TACACS, use the **debug tacacs** privileged EXEC command. The **no** form of this command disables debugging output.

debug tacacs

no debug tacacs

Syntax Description   This command has no arguments or keywords.

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.

Examples   The following is sample output from the **debug aaa authentication** command for a TACACS login attempt that was successful. The information indicates that TACACS+ is the authentication method used.

```
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
```

The following is sample output from the **debug tacacs** command for a TACACS login attempt that was successful, as indicated by the status PASS:

```
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
```

The following is sample output from the **debug tacacs** command for a TACACS login attempt that was unsuccessful, as indicated by the status FAIL:

```
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	Command	Description
	debug aaa accounting	Displays information on accountable events as they occur.
	debug aaa authentication	Displays information on AAA/TACACS+ authentication.

debug tacacs events

To display information from the TACACS+ helper process, use the **debug tacacs events** privileged EXEC command. The **no** form of this command disables debugging output.

debug tacacs events

no debug tacacs events

Syntax Description   This command has no arguments or keywords.

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 substantial 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.

Examples   The following is sample output from the **debug tacacs events** command. In this example, the opening and closing of a TCP connection to a TACACS+ server are shown, and the bytes read and written over the connection and the TCP status of the connection:

```
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 TACACS messages are intended to be self-explanatory or for consumption by service personnel only. However, the messages shown 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	Command	Description
	debug aaa accounting	Displays information on accountable events as they occur.
	debug aaa authentication	Displays information on AAA/TACACS+ authentication.
	debug aaa authorization	Displays information on AAA/TACACS+ authorization.
	debug sw56	Displays debug information for switched 56 K services.