

debug serial interface

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

debug serial interface

no debug serial interface

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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

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

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

Table 223 describes the significant fields.

Table 223 *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.

Table 224 describes additional error messages that the **debug serial interface** command can generate for HDLC.

Table 224 *debug serial interface Error Messages for HDLC*

Field	Description
Illegal serial link type code <xxx>, PC = 0xnnnnnnn	Router attempted to send a packet containing an unknown packet type.
Illegal HDLC serial type code <xxx>, PC = 0xnnnnnn	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 <nnnnnnnnnn>	Bridge packet is received over a serial interface configured for HDLC, and bridging is not configured on that interface.

Debug Serial Interface for HSSI

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

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

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

Debug Serial Interface for ISDN Basic Rate

Table 225 describes error messages that the **debug serial interface** command can generate for ISDN Basic Rate.

Table 225 *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 0xnnnnnnnn	BRI hardware has been reset. The 0xnnnnnnnn variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers.

Table 225 *debug serial interface Error Messages for ISDN Basic Rate (continued)*

Message	Description
BRI(d): Bad state in SCMs scm1= <i>x</i> scm2= <i>x</i> scm3= <i>x</i>	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): Bad state in SCONs scon1= <i>x</i> scon2 = <i>x</i> scon3= <i>x</i>	
BRI(d): Bad state ub SCR; SCR= <i>x</i>	
BRI(d): Illegal packet encapsulation= <i>n</i>	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 226](#) describes the additional error messages that the **debug serial interface** command can generate for an MK5025 device.

Table 226 *debug serial interface Error Messages for an MK5025 Device*

Message	Description
MK5(d): Reset from 0x <i>nnnnnnnn</i>	Hardware has been reset. The 0x <i>nnnnnnnn</i> variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers.
MK5(d): Illegal packet encapsulation= <i>n</i>	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=(<i>x</i>)	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.
MK5(d): New serial state= <i>n</i>	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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug serial packet

no debug serial packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

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

As 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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug service-module

no debug service-module

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug sgbp dial-bids

no debug sgbp dial-bids

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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 configuration 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.

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-explanatory.

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** command in privileged EXEC mode. To disable debugging output, 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 Modes

Privileged EXEC

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. To disable debugging output, use the **no** form of this command.

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

no debug sgcp errors

Syntax Description	<p>endpoint <i>string</i></p> <p>(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint.</p> <p>On the Cisco MC3810 router, the endpoint string syntax takes the following forms:</p> <ul style="list-style-type: none"> DS1 endpoint: DS1-slot/port POTS endpoint: aaln/slot/port <p>On the Cisco 3600 router, the endpoint string syntax takes the following forms:</p> <ul style="list-style-type: none"> DS1 endpoint: <i>slot/subunit/DS1-ds1 number/ds0 number</i> 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. To disable debugging output, use the **no** form of this command.

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

no debug sgcp events

Syntax Description	<p>endpoint <i>string</i></p> <p>(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint.</p> <p>On the Cisco MC3810 router, the endpoint string syntax takes the following forms:</p> <ul style="list-style-type: none"> DS1 endpoint: DS1-<i>slot/port</i> POTS endpoint: aaln/<i>slot/port</i> <p>On the Cisco 3600 router, the endpoint string syntax takes the following forms:</p> <ul style="list-style-type: none"> DS1 endpoint: <i>slot/subunit</i>/DS1-<i>ds1 number/ds0 number</i> POTS endpoint: aaln/<i>slot/subunit/port</i>
---------------------------	--

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=0x800003,
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=0x800003,
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 vtspi 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. To disable debugging output, use the **no** form of this command.

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

no debug sgcp packet

Syntax Description	<p>endpoint <i>string</i></p> <p>(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint.</p> <p>On the Cisco MC3810, the endpoint string syntax takes the following forms:</p> <ul style="list-style-type: none"> DS1 endpoint: DS1-<i>slot/port</i> POTS endpoint: aaln/<i>slot/port</i> <p>On the Cisco 3600, the endpoint string syntax takes the following forms:</p> <ul style="list-style-type: none"> DS1 endpoint: <i>slot/subunit</i>/DS1-<i>ds1 number/ds0 number</i> POTS endpoint: aaln/<i>slot/subunit/port</i>
---------------------------	--

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

<---
```

debug sgcp packet

```

*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

```

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 events	Debugs SGCP events.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug snasw dlc

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

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 Modes

Privileged EXEC

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      Link          Size of ISR/
           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 dlcfiter	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** command in privileged EXEC mode.

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 Modes

Privileged EXEC

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      Signal Name      Sending      Receiving
           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
```

■ debug snasw ips

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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug snmp packet

no debug snmp packet

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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 227 describes the significant fields shown in the display.

Table 227 *debug snmp packet Field Descriptions*

Field	Description
Get-next request	<p>Indicates what type of SNMP PDU the packet is. Possible types are as follows:</p> <ul style="list-style-type: none"> • Get request • Get-next request • Response • Set request • V1 Trap • Get-bulk request • Inform request • V2 Trap <p>Depending on the type of PDU, the rest of this line displays different fields. The indented lines following this line list the MIB object names and corresponding values.</p>
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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug snmp requests

no debug snmp requests

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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 228](#) describes the significant fields shown in the display.

Table 228 *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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug sntp adjust

no debug sntp adjust

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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 snmp packets

To display information about SNMP packets sent and received, use the **debug snmp packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug snmp packets

no debug snmp packets

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

Examples The following is sample output from the **debug snmp packets** command when a message is received:

```
Router# debug snmp packets

Received SNMP 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 snmp packets** command when a message is sent:

```
Router# debug snmp packets

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

[Table 229](#) describes the significant fields shown in the display.

Table 229 *debug snmp packets Field Descriptions*

Field	Description
length	Length of the SNMP 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	SNMP 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.

Table 229 debug snmp packets Field Descriptions (continued)

Field	Description
org	Originate time stamp. This value indicates the time the request was sent by the router.
rec	Receive time stamp. This value indicates the time the request was received by the SNMP server.
xmt	Transmit time stamp. This value indicates the time the reply was sent by the SNMP 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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug sntp select

no debug sntp select

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug source bridge

no debug source bridge

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug source error

no debug source error

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

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. SRTL B 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): BPDUs 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.