debug tarp events

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

debug tarp events

no debug tarp events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

The following is sample output from the **debug tarp events** and **debug tarp packets** commands after the **tarp resolve** command was used to determine the NSAP address for the TARP target identifier (TID) named artemis.

```
Router# debug tarp events

Router# debug tarp packets

Router# tarp resolve artemis

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

 NET corresponding to TID artemis is 49.0001.1111.1111.1111.00

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

Table 203 describes the significant fields in this display.

Table 203 debug tarp events Field Descriptions—tarp resolve Command

Field	Descriptions
Sending TARP type 1 PDU	PDU requesting the NSAP of the specified TID.
timeout	Number of seconds the router will wait for a response from the Type 1 PDU. The timeout is set by the **tarp t1-response-timer** command.
NET corresponding to	NSAP address (in this case, 49.0001.1111.1111.1111.00) for the specified TID.
*Mar 1 00:43:59	Debug time stamp.
TARP-PA: Propagated	TARP packet: A Type 1 PDU was sent out on Ethernet interface 0.
Lft	Lifetime of the PDU (in hops).
Seq	Sequence number of the PDU.
Prot type	Protocol type of the PDU.
URC	Update remote cache bit.
Ttid len	Destination TID length.
Stid len	Source TID length.
Prot addr len	Protocol address length (bytes).
Destination NSAP	NSAP address that the PDU is being sent to.
Originator's NSAP	NSAP address that the PDU was sent from.
Target TID	TID that the PDU is being sent to.
Originator's TID	TID that the PDU was sent from.
TARP-EV: Packet not propagated	TARP event: The Type 1 PDU was not propagated on Ethernet interface 0 because the adjacency is not up.
TARP-EV: No route found	TARP event: The Type 1 PDU was not sent because no route was available.
TARP-PA: Received TARP	TARP packet: A Type 3 PDU was received on Ethernet interface 0.
Packet sent/propagated by	NSAP address of the router that sent or propagated the PDU.
TARP-PA: Created new DYNAMIC cache entry	TARP packet: A dynamic entry was made to the local TID cache.

Related Commands

Command	Description
debug tarp packets	Displays general information on TARP packets received, generated, and propagated on the router.

debug tarp packets

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

debug tarp packets

no debug tarp packets

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

The following is sample output from the **debug tarp packet** command after the **tarp query** command was used to determine the TID for the NSAP address 49.0001.3333.3333.3333.00:

```
Router# debug tarp packets

Router# debug tarp events

Router# tarp query 49.0001.3333.3333.3333.00

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

 TID corresponding to NET 49.0001.3333.3333.3333.00 is cerdiwen

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

Table 204 describes the significant fields shown in the display.

Table 204 debug tarp packets Field Descriptions—tarp query Command

Field	Descriptions
Sending TARP type 5 PDU	PDU requesting the TID of the specified NSAP.
timeout	Number of seconds the router will wait for a response from the Type 5 PDU. The timeout is set by the **tarp arp-request-timer** command.
TID corresponding to NET	TID (in this case cerdiwen) for the specified NSAP address.
*Mar 2 03:10:11	Debug time stamp.

Table 204 debug tarp packets Field Descriptions—tarp query Command (continued)

Field	Descriptions
TARP-PA: Originated TARP packet	TARP packet: A Type 5 PDU was sent.
TARP P-A: Received TARP	TARP packet: A Type 3 PDU was received.
Lft	Lifetime of the PDU (in hops).
Seq	Sequence number of the PDU.
Prot type	Protocol type of the PDU.
URC	The update remote cache bit.
Ttid len	Destination TID length.
Stid len	Source TID length.
Prot addr len	Protocol address length (in bytes).
Packet sent/propagated	NSAP address of the router that sent or propagated the PDU.
Originator's NSAP	NSAP address that the PDU was sent from.
Originator's TID	TID that the PDU was sent from.
TARP-PA: Created new DYNAMIC cache entry	TARP packet: A dynamic entry was made to the local TID cache.

Related Commands

Command	Modification
debug tarp events	Displays information on TARP activity.

debug tccs signaling

To see information about the transparent CCS connection, use the **debug tccs signaling** command. Enter the **no** form of this command to disable debugging output.

debug tccs signaling

no debug tccs signaling

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced.
12.1(2)T	This command was integrated into the Cisco IOS 12.1(2)T release.

Usage Guidelines

Use this command with caution, because it displays every packet that the D channel transmits to the packet network and to the PBX. This command is CPU-intensive and should be used only as a last resort.

Use this command to debug a transparent CCS connection in the following cases:

- Observe the results of the **ccs connect** command results when you configure the setup.
- Observe CCS traffic at run time; the output shows the actual CCS packets received at run time and the number of packets received and sent.

Examples

The following example shows output from the command on both the originating and terminating sides:

```
Router# debug tccs signaling

TCCS Domain packet debugging is on
mazurka-4#
01:37:12: 1 tccs packets received from the port.
01:37:12: 1 tccs packets received from the nework.
01:37:12: tx_tccs_fr_pkt:pkt rcvd from network->tx_start
01:37:12: tx_tccs_fr_pkt: dlci=37, cid=100, payld-type =0,
        payld-length=162, cid_type=424
01:37:12: datagramsize=26
01:37:12: [0] A4 40 C0 0
01:37:12: [4] 86 86 86 86
01:37:12: [8] 86 86 86 86
01:37:12: [12] 86 86 86 86
01:37:12: [16] 86 86 86 86
01:37:12: [20] 86 86 86 86
```

```
01:37:12: [24] 86 86 11 48
01:37:12: 2 tccs packets received from the port.
01:37:12: 1 tccs packets received from the nework.
01:37:12: pri_tccs_rx_intr:from port->send_sub_channel
01:37:12: tccs_db->vcd = 37, tccs_db->cid = 100
01:37:12: pak->datagramsize=25
01:37:12: [0] A4 40 C0 0
01:37:12: [4] 42 43 43 43
01:37:12: [8] 43 43 43 43
01:37:12: [12] 43 43 43 43
01:37:12: [16] 43 43 43 43
01:37:12: [20] 43 43 43 43
01:37:12: [24] 43 43 43 0
```

```
Router# debug tccs signaling
00:53:26: 61 tccs packets received from the port.
00:53:26: 53 tccs packets received from the nework.
00:53:26: pri_tccs_rx_intr:from port->send_sub_channel
00:53:26: tccs_db->vcd = 37, tccs_db->cid = 100
00:53:26: pak->datagramsize=7
00:53:26: [0] A4 40 C0 0
00:53:26: [4] 0 1 7F 64
00:53:27: 62 tccs packets received from the port.
00:53:27: 53 tccs packets received from the nework.
00:53:27: pri_tccs_rx_intr:from port->send_sub_channel
00:53:27: tccs_db->vcd = 37, tccs_db->cid = 100
00:53:27: pak->datagramsize=7
00:53:27: [0] A4 40 C0 0
00:53:27: [4] 0 1 7F 64
00:53:28: 63 tccs packets received from the port.
00:53:28: 53 tccs packets received from the nework.
00:53:28: pri_tccs_rx_intr:from port->send_sub_channel
00:53:28: tccs_db->vcd = 37, tccs_db->cid = 100
00:53:28: pak->datagramsize=7
00:53:28: [0] A4 40 C0 0
00:53:28: [4] 0 1 7F 64
00:53:29: 64 tccs packets received from the port.
00:53:29: 53 tccs packets received from the nework.
```

debug tdm

To display time-division multiplexer (TDM) BUS CONNECTION information each time a connection is made on Cisco AS5300 access servers, use the **debug tdm** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug tdm [*api* | *detail* | *dynamic* | *pri* | *test* | *tsi* | *vdev*]

no debug tdm [*api* | *detail* | *dynamic* | *pri* | *test* | *tsi* | *vdev*]

Syntax Description

api	(Optional) Displays a debug message whenever the TDM subsystem API is invoked from another subsystem.
detail	(Optional) Displays detailed messages (i.e., trace messages) whenever the TDM software executes.
dynamic	(Optional) Displays TDM debugging information whenever a backplane timeslot is allocated or deallocated.
pri	(Optional) Routes modem back-to-back connections from the modem-to-PRI board to modem board. By default, the modem back-to-back connections route from modem board to motherboard to modem board.
test	(Optional) Simulates the failure of allocating a TDM timeslot. Verifies that the software and TDM hardware recovers from the failure.
tsi	(Optional) Displays debugging information about the TSI Chip MT8980/MT90820 driver.
vdev	(Optional) TDM per voice device debug <0-2> slot and port number (that is, 0/1). Displays debug information whenever a modem board TDM connection is made.

Usage Guidelines

The **debug tdm** command output is to be used primarily by a Cisco technical support representative. The **debug tdm** command enables display of debugging messages for specific areas of code that execute.

Examples

The following examples show the turning on of the debug option, performing a modem call, and turning off the debug option:

```
Router# debug tdm api

TDM API debugging is on
Router#
23:16:04: TDM(vdev reg: 0x3C500100/PRI reg: 0x3C400100): two way connection requested.
23:16:04: TDM(reg: 0x3C500100): Close connection to STo8, channel 1
23:16:04: TDM(reg: 0x3C500100): Connect STi4, channel 1 to STo8, channel 1
23:16:04: TDM(reg: 0x3C500100): Close connection to STo4, channel 1
23:16:04: TDM(reg: 0x3C500100): Connect STi8, channel 1 to STo4, channel 1
23:16:04: TDM(reg: 0x3C400100): Close connection to STo12, channel 31
23:16:04: TDM(reg: 0x3C400100): Close connection to STo8, channel 31
23:16:04: TDM(reg: 0x3C400100): Connect STi12, channel 31 to STo4, channel 1
23:16:04: TDM(reg: 0x3C400100): Connect STi4, channel 1 to STo12, channel 31
23:18:22: TDM(reg: 0x3C500100): default RX connection requested.
23:18:22: TDM(reg: 0x3C500100): Close connection to STo8, channel 1
23:18:22: TDM(reg: 0x3C500100): default TX connection requested.
23:18:22: TDM(reg: 0x3C500100): Close connection to STo4, channel 1
```

```
23:18:22: TDM(reg: 0x3C500100): Close connection to STo8, channel 1
23:18:22: TDM(reg: 0x3C500100): Close connection to STo4, channel 1
23:18:22: TDM(reg: 0x3C400100): default RX connection requested.
23:18:22: TDM(reg: 0x3C400100): Close connection to STo4, channel 1
23:18:22: TDM(reg: 0x3C400100): Connect STi12, channel 31 to STo8, channel 31
23:18:22: TDM(reg: 0x3C400100): default TX connection requested.
23:18:22: TDM(reg: 0x3C400100): Close connection to STo12, channel 31
23:18:22: TDM(reg: 0x3C400100): Connect STi8, channel 31 to STo12, channel 31
Router# no debug tdm api
TDM API debugging is off

Router# debug tdm detail
TDM Detail Debug debugging is on
router_2#show tdm pool
```

Dynamic Backplane Timeslot Pool:

```
Grp ST Ttl/Free Req(Cur/Ttl/Fail)    Queues(Free/Used)   Pool Ptr
 0 0-3 128 128  0  0    0    0x60CB6B30 0x60CB6B30 0x60CB6B28
 1 4-7 128 128  0  3    0    0x60CB6B40 0x60CB6B40 0x60CB6B2C
Router#
Router# no debug tdm detail
TDM Detail Debug debugging is off


Router# debug tdm dynamic
TDM Dynamic BP Allocation debugging is on
Router#
23:30:16: tdm_allocate_bp_ts(), slot# 1, chan# 3
23:30:16: TDM(reg: 0x3C500100): Open Modem RX ST8, CH3 to BP ST4 CH3
23:30:16: TDM(reg: 0x3C500100): Open Modem TX ST8, CH3 to BP ST4 CH3
23:30:16: TDM Backplane Timeslot Dump @ 0x60E6D244, tdm_free_bptsCount[1] = 127
 vdev_slot : 0x01   bp_stream : 0x04
 vdev_channel : 0x03  bp_channel : 0x03   freeQueue : 0x60CB6B40
23:30:16: TDM(PRI:0x3C400100):Close PRI framer st12 ch31
23:30:16: TDM(PRI:0x3C400100):Close HDLC controller st8 ch31
23:30:43: tdm_deallocate_bp_ts(), slot# 1, chan# 3
23:30:43: TDM(reg: 0x3C500100):Close Modem RX ST8, CH3 to BP ST4 CH3
23:30:43: TDM(reg: 0x3C500100):Close Modem TX ST8, CH3 to BP ST4 CH3
23:30:43: TDM Backplane Timeslot Dump @ 0x60E6D244, tdm_free_bptsCount[1] = 128
 vdev_slot : 0x01   bp_stream : 0x04
 vdev_channel : 0x03  bp_channel : 0x03   freeQueue : 0x60CB6B40
Router#
Router# no debug tdm dynamic
TDM Dynamic BP Allocation debugging is off


Router# debug tdm pri
TDM connectvia PRI feature board debugging is on
Router# no debug tdm pri
TDM connectvia PRI feature board debugging is off


Router# debug tdm test
TDM Unit Test debugging is on
23:52:01: Bad tdm_allocate_bp_ts() call, simulating error condition for vdev in slot 1
port 5
Router# no debug tdm test
TDM Unit Test debugging is off

Router# debug tdm tsi
TDM TSI debugging is on
Router#
23:56:40: MT90820(reg: 0x3C500100): Close connection to STi8, channel 9
```

```
23:56:40: MT90820(reg: 0x3C500100): Connect STi4, channel 10 to STo8, channel 9
23:56:40: MT90820(reg: 0x3C500100): Close connection to STi4, channel 10
23:56:40: MT90820(reg: 0x3C500100): Connect STi8, channel 9 to STo4, channel 10
23:56:40: MT90820(reg: 0x3C400100): Close connection to STi12, channel 31
23:56:40: MT90820(reg: 0x3C400100): Close connection to STi8, channel 31
23:56:40: MT90820(reg: 0x3C400100): Connect STi12, channel 31 to STo4, channel 10
23:56:40: MT90820(reg: 0x3C400100): Connect STi4, channel 10 to STo12, channel 31
23:57:03: MT90820(reg: 0x3C500100): Close connection to STi8, channel 9
23:57:03: MT90820(reg: 0x3C500100): Close connection to STi4, channel 10
23:57:03: MT90820(reg: 0x3C500100): Close connection to STi8, channel 9
23:57:03: MT90820(reg: 0x3C500100): Close connection to STi4, channel 10
23:57:03: MT90820(reg: 0x3C400100): Close connection to STi4, channel 10
23:57:03: MT90820(reg: 0x3C400100): Connect STi12, channel 31 to STo8, channel 31
23:57:03: MT90820(reg: 0x3C400100): Close connection to STi12, channel 31
23:57:03: MT90820(reg: 0x3C400100): Connect STi8, channel 31 to STo12, channel 31
Router#
Router# no debug tdm tsi
TDM TSI debugging is off


Router# debug tdm vdev ?
 <0-2> Slot/port number (i.e. 0/1)
Router# debug tdm vdev 1/8
Enabling TDM debug for voice device in slot 0 port 1
Router#
23:55:00: TDM(vdev reg: 0x3C500100/PRI reg: 0x3C400100): two way connection requested.
23:55:00: tdm_allocate_bp_ts(), slot# 1, chan# 8
23:55:00: TDM(reg: 0x3C500100): Open Modem RX ST8, CH8 to BP ST4 CH9
23:55:00: TDM(reg: 0x3C500100): Open Modem TX ST8, CH8 to BP ST4 CH9
23:55:00: TDM Backplane Timeslot Dump @ 0x60E6D2D4, tdm_free_bptsCount[1] = 127
 vdev_slot : 0x01   bp_stream : 0x04
 vdev_channel : 0x08  bp_channel : 0x09   freeQueue : 0x60CB6B40

23:55:00: TDM(PRI:0x3C400100):Close PRI framer st12 ch31
23:55:00: TDM(PRI:0x3C400100):Close HDLC controller st8 ch31
23:55:31: TDM(reg: 0x3C500100): default RX connection requested.
23:55:31: TDM(reg: 0x3C500100): default TX connection requested.
23:55:31: tdm_deallocate_bp_ts(), slot# 1, chan# 8
23:55:31: TDM(reg: 0x3C500100):Close Modem RX ST8, CH8 to BP ST4 CH9
23:55:31: TDM(reg: 0x3C500100):Close Modem TX ST8, CH8 to BP ST4 CH9
23:55:31: TDM Backplane Timeslot Dump @ 0x60E6D2D4, tdm_free_bptsCount[1] = 128
 vdev_slot : 0x01   bp_stream : 0x04
 vdev_channel : 0x08  bp_channel : 0x09   freeQueue : 0x60CB6B40
Router#
Router# no debug tdm vdev 1/8
Disabling TDM debug for voice device in slot 0 port 1
Router#
```

debug telco-return msg

To display debug messages for telco-return events, use the **debug cable telco-return msg** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cable telco-return msg

no debug cable telco-return msg

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for telco-return messages is not enabled.

Command History

Release	Modification
12.0(4)XI	This command was introduced.

Examples

```
ubr7223#debug cable telco-return msg
CMTS telco-return msg debugging is on
```

Related Commands

Command	Description
debug telco-return msg	Displays debug messages for telco-return events.

debug telnet

To display information about Telnet option negotiation messages for incoming Telnet connections to a Cisco IOS Telnet server, use the **debug telnet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug telnet

no debug telnet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
8.1	This command was introduced.

Examples

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

```
Router# debug telnet

*Oct 28 21:31:12.035:Telnet1/00:1 1 251 1
*Oct 28 21:31:12.035:TCP1/00:Telnet sent WILL ECHO (1)
*Oct 28 21:31:12.035:Telnet1/00:2 2 251 3
*Oct 28 21:31:12.035:TCP1/00:Telnet sent WILL SUPPRESS-GA (3)
*Oct 28 21:31:12.035:Telnet1/00:4 4 251 0
*Oct 28 21:31:12.035:TCP1/00:Telnet sent WILL BINARY (0)
*Oct 28 21:31:12.035:Telnet1/00:40000 40000 253 0
*Oct 28 21:31:12.035:TCP1/00:Telnet sent DO BINARY (0)
*Oct 28 21:31:12.035:Telnet1/00:10000000 10000000 253 31
*Oct 28 21:31:12.035:TCP1/00:Telnet sent DO WINDOW-SIZE (31)
*Oct 28 21:31:12.035:TCP1/00:Telnet received WILL TTY-TYPE (24)
*Oct 28 21:31:12.035:TCP1/00:Telnet sent DO TTY-TYPE (24)
*Oct 28 21:31:12.035:Telnet1/00:Sent SB 24 1
*Oct 28 21:31:12.035:TCP1/00:Telnet received WILL TTY-SPEED (32) (refused)
*Oct 28 21:31:12.035:TCP1/00:Telnet sent DONT TTY-SPEED (32)
*Oct 28 21:31:12.035:TCP1/00:Telnet received DO SUPPRESS-GA (3)
*Oct 28 21:31:12.035:TCP1/00:Telnet received WILL SUPPRESS-GA (3)
*Oct 28 21:31:12.035:TCP1/00:Telnet sent DO SUPPRESS-GA (3)
*Oct 28 21:31:12.035:TCP1/00:Telnet received DO ECHO (1)
*Oct 28 21:31:12.035:TCP1/00:Telnet received DO BINARY (0)
*Oct 28 21:31:12.035:TCP1/00:Telnet received WILL BINARY (0)
*Oct 28 21:31:12.059:TCP1/00:Telnet received WILL COMPORT (44)
*Oct 28 21:31:12.059:TCP1/00:Telnet sent DO COMPORT (44)
*Oct 28 21:31:12.059:TCP1/00:Telnet received DO COMPORT (44)
*Oct 28 21:31:12.059:TCP1/00:Telnet sent WILL COMPORT (44)
*Oct 28 21:31:12.059:TCP1/00:Telnet received WONT WINDOW-SIZE (31)
*Oct 28 21:31:12.059:TCP1/00:Telnet sent DONT WINDOW-SIZE (31)
*Oct 28 21:31:12.059:Telnet1/00:recv SB 24 0
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 10 TTY1/00:Telnet COMPORT rcvd bad
suboption:0xA/0x1E
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 1
```

```
*Oct 28 21:31:12.091:Telnet_CP-1/00 baudrate index 0
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 101 X.dctBXctBXctBX`W`P`>
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 2
*Oct 28 21:31:12.091:Telnet_CP-1/00 datasize index 8  8
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 102X.dctBXctBXctBX`W`P`>
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 3
*Oct 28 21:31:12.091:Telnet_CP-1/00 parity index 1  0
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 103 X.dctBXctBXctBX`W`P`>
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 4
*Oct 28 21:31:12.091:Telnet_CP-1/00 stopbits index 1
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 104 X.dctBXctBXctBX`W`P`>
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 5
*Oct 28 21:31:12.091:Telnet_CP-1/00 HW flow on
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 105 X.dctBXctBXctBX`W`P`>
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 11 nTTY1/00:Telnet COMPORT rcvd ba
d suboption:0xB/0xEE
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 5
*Oct 28 21:31:12.091:Telnet_CP-1/00 unimplemented option 0x10
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 105
*Oct 28 21:31:12.091:Telnet1/00:recv SB 44 5
*Oct 28 21:31:12.091:Telnet_CP-1/00 DTR on
*Oct 28 21:31:12.091:Telnet1/00:Sent SB 44 105X.dctBXctBXctBX`W`P`>
*Oct 28 21:31:12.091:TCP1/00:Telnet received WONT WINDOW-SIZE (31)
*Oct 28 21:31:12.099:Telnet1/00:Sent SB 44 107 3
*Oct 28 21:31:12.099:COMPORT1/00:sending notification 0x33
```

Table 205 describes the significant fields shown in the display.

Table 205 debug telnet Field Descriptions

Field	Description
Telnet1/00: 1 1 251 1	Untranslated decimal option negotiations that are sent. 1/00 denotes the line number that the Telnet server is operating on.
TCP1/00:	Symbolically decoded option negotiations. 1/00 denotes the line number that the Telnet server is operating on. Telnet option negotiations are defined in the following RFCs: • RFC 854—*Telnet Protocol Specification* • RFC 856—*Telnet Binary Transmission* • RFC 858—*Telnet Suppress Go Ahead Option* • RFC 1091—*Telnet Terminal-Type Option* • RFC 1123, sec. 3—*Requirements for Internet Hosts—Application and Support* • RFC 2217—*Telnet Com Port Control Option*

Related Commands

Command	Description
debug ip tcp transactions	Displays information on significant TCP transactions such as state changes, retransmissions, and duplicate packets.
debug modem	Displays modem line activity on an access server.

debug text-to-fax

To the off-ramp text-to-fax conversion, use the **debug text-to-fax** EXEC command to show information relating. Use the **no** form of this command to disable debugging output.

debug text-to-fax

[no] debug text-to-fax

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0(4)T	This command was introduced.

Examples

The following debug output shows the off-ramp text-to-fax conversion.

```
Router# debug text-to-fax
Text to fax debugging is on
Router#6d03h: text2fax_data_handler: START_OF_CONNECTION
6d03h: text2fax_data_handler: new_context
6d03h: text2fax_data_handler: resolution: fine
6d03h: text2fax_data_handler: buffer size: 50
6d03h: text2fax_put_buffer: START_OF_FAX_PAGE
6d03h: text2fax_put_buffer: START_OF_FAX_PAGE
6d03h: text2fax_put_buffer: END_OF_FAX_PAGE. Dial now ...if not in progress

6d03h: text2fax_data_handler: START_OF_DATA
6d03h: text2fax_data_handler: END_OF_DATA
6d03h: text2fax_data_handler: Dispose context
6d03h: text2fax_data_handler: START_OF_CONNECTION
6d03h: text2fax_data_handler: END_OF_CONNECTION
6d03h: %FTSP-6-FAX_CONNECT: Transmission
6d03h: %FTSP-6-FAX_DISCONNECT: Transmission
6d03h: %LINK-3-UPDOWN: Interface Serial1:22, changed state to down
```

debug tftp

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

debug tftp

no debug tftp

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug tftp** command from the **copy system:running-config tftp** EXEC command:

```
Router# debug tftp

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

Table 206 describes the significant fields in the first line of output.

Table 206 debug tftp Field Descriptions

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

debug tgrm

To display debug messages for all trunk groups, use the **debug tgrm** EXEC command. To end the display of debug messages, use the **no** form of this command.

debug tgrm

no debug tgrm

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

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

Examples

The following examples show output of the **debug tgrm** command.

This message indicates which interface was selected for the outgoing voice call:

```
TGRM:tgrm_select_interface() - Interface Serial0:23 selected
```

This message indicates that the outgoing voice call was denied because of trunk group configuration (*Allowed* shows the **max-calls** value):

```
TGRM:tgrm_select_interface() - Outgoing voice call denied. Allowed = 5, Current = 6
```

This message indicates that the trunk group has no interfaces belonging to it:

```
TGRM:tgrm_select_interface() - Trunk group 3 has no members
```

This message indicates that the outgoing voice or modem call was denied because of trunk group configuration (*Allowed* shows the **max-calls** value). For a data call, the message is "Outgoing data call denied."

```
TGRM:Serial0:23:tgrm_accept_call() - Outgoing voice call denied. Allowed = > 5,
Current = 6
```

This message indicates that the incoming data call was denied because of trunk group configuration (*Allowed* shows the **max-calls** value). For a voice call, the message is "Incoming voice call denied."

```
TGRM:Serial0:23:tgrm_accept_call() - Incoming data call denied. Allowed = 5, Current = 6
```

Related Commands

Command	Description
debug cdapi events	Displays information about the CDAPI.
debug isdn events	Displays ISDN events occurring on the user side (on the router) of the ISDN interface.
debug isdn q931	Displays information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network.
trunk group (global)	Defines a trunk group globally.
trunk-group (interface)	Assigns a specified interface to a defined trunk group.

debug tiff reader

To display output about the off-ramp TIFF reader, use the **debug tiff reader** EXEC command. Use the **no** form of this command to disable debugging output.

debug tiff reader

[no] debug tiff-reader

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0(4)T	This command was introduced.

Examples

The following debug example displays information about the off-ramp TIFF reader.

```
Router# debug tiff reader
*Jan 1 18:59:13.683: tiff_reader_data_handler: new context
*Jan 1 18:59:13.683: tiff_reader_data_handler: resolution: standard
*Jan 1 18:59:13.683: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
ENGINE_START/DONE gggg(pl 616E9994)

*Jan 1 18:59:13.691: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.699: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)

*Jan 1 18:59:13.703: tiff_reader_put_buffer: START_OF_FAX_PAGEi>> tiff_reader_engine()
case FAX_EBUFFER gggg

*Jan 1 18:59:13.711: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.719: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.727: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.735: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.743: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.751: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.759: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.767: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.775: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
```

```
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.787: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.795: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.803: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.811: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.819: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.827: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.835: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.843: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.851: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.863: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.871: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.879: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.887: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.895: tiff_reader_data_handler: buffer size: 1524
*Jan 1 18:59:13.903: tiff_reader_data_handler: buffer size: 1524i>> tiff_reader_engine()
case FAX_EBUFFER pppp(pl 616E9994)
i>> tiff_reader_engine() case FAX_EBUFFER gggg

*Jan 1 18:59:13.907: tiff_reader_data_handler: buffer size: 311i>> tiff_r_finish()
END_OF_FAX_PAGE pppp

*Jan 1 18:59:13.907: tiff_reader_put_buffer: END_OF_FAX_PAGE. Dial now ...if not in
progress
*Jan 1 18:59:13.907: tiff_reader_data_handler: END_OF_DATA
*Jan 1 18:59:13.907: tiff_reader_data_handler: BUFF_END_OF_PART
*Jan 1 18:59:13.907: tiff_reader_data_handler: Dispose context
```

Related Commands

Command	Description
debug tiff writer	Displays output about the on-ramp TIFF writer.

debug tiff writer

To display output about the on-ramp TIFF writer, use the **debug tiff writer** EXEC command. Use the **no** form of this command to disable debugging output.

debug tiff writer

[no] debug tiff-writer

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0(4)T	This command was introduced.

Examples

The following debug example shows information about the off-ramp TIFF writer.

```
Router# debug tiff writer
*Jan 1 18:54:59.419: tiff_writer_data_process: START_OF_CONNECTION
18:55:10: %FTSP-6-FAX_CONNECT: Reception
*Jan 1 18:55:14.903: tiff_writer_data_process: START_OF_FAX_PAGE
*Jan 1 18:55:14.903: tiff_writer_data_process: tiff file created = 2000:01:01 18:55:14
18:55:21: %FTSP-6-FAX_DISCONNECT: Reception
*Jan 1 18:55:19.039: tiff_writer_data_process: END_OF_CONNECTION or ABORT_CONNECTION
*Jan 1 18:55:19.039: tiff_writer_put_buffer: END_OF_FAX_PAGE

*Jan 1 18:55:19.039: send TIFF_PAGE_READY
*Jan 1 18:55:19.039: send TIFF_PAGE_READY
18:55:21: %LINK-3-UPDOWN: Interface Serial2:0, changed state to down
```

Related Commands

Command	Description
debug tiff reader	Displays output about the on-ramp TIFF reader.

debug token ring

To display messages about Token Ring interface activity, use the **debug token ring** privileged EXEC command. The **no** form of this command disables debugging output.

debug token ring

no debug token ring

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

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

Caution It is best to use this command only on router and bridges with light loads.

Examples

The following is sample output from the **debug token ring** command:

```
Router# debug token ring

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

Table 207 describes the significant fields in the second line of output.

Table 207 debug token ring Field Descriptions

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

Table 207 debug token ring Field Descriptions (continued)

Message	Description
MAC:	Type of packet, as follows: • MAC—Media Access Control • LLC—Link Level Control
acfc: 0x1105	Access Control, Frame Control bytes, as defined by the IEEE 802.5 standard.
Dst: c000.ffff.ffff	Destination address of the frame.
Src: 5000.1234.5678	Source address of the frame.
bf: 0x45	Bridge flags for internal use by technical support staff.

Table 208 describes the significant fields shown in the third line of output.

Table 208 debug token ring Field Descriptions

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

Table 209 describes the significant fields shown in the fifth line of output.

Table 209 debug token ring Field Descriptions

Message	Description
TR0:	Name of the interface associated with the Token Ring event.
out:	Indication of whether the packet was input to the interface (in) or output from the interface (out).
LLC:	Type of frame, as follows: • MAC—Media Access Control • LLC—Link Level Control
AAAA0300	This and the octets that follow it indicate the contents (hex) of the frame.
ln: 28	The length of the information field (in bytes).

debug tsp

To display information about the telephony service provider (TSP), use the **debug tsp** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug tsp {*all* | *call* | *error* | *port*}

no debug tsp {*all* | *call* | *error* | *port*}

Syntax Description

all	Enables all TSP debugging (except statistics)
call	Enables call debugging.
error	Error debugging.
port	Port debugging.

Defaults

Disabled

Command History

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

Examples

The following example shows output for the **debug tsp** *all* command:

```
01:04:12:CDAPI TSP RX ===> callId=(32 ), Msg=(CDAPI_MSG_CONNECT_IND,1 )
Sub=(CDAPI_MSG_SUBTYPE_NULL,0 )cdapi_tsp_connect_ind
01:04:12:TSP CDAPI:cdapi_free_msg returns 1
01:04:13:tsp_process_event:[0:D, 0.1 , 3] tsp_cdapi_setup_ack tsp_alert
01:04:13:tsp_process_event:[0:D, 0.1 , 5] tsp_alert_ind
01:04:13:tsp_process_event:[0:D, 0.1 , 10]
01:04:14:tsp_process_event:[0:D, 0.1 , 10]
01:04:17:CDAPI TSP RX ===> callId=(32 ), Msg=(CDAPI_MSG_DISCONNECT_IND,7 )
Sub=(CDAPI_MSG_SUBTYPE_NULL,0 )cdapi_tsp_disc_ind
01:04:17:TSP CDAPI:cdapi_free_msg returns 1
01:04:17:tsp_process_event:[0:D, 0.1 , 27] cdapi_tsp_release_indtsp_disconnet_tdm
01:04:17:tsp_process_event:[0:D, 0.4 , 7] cdapi_tsp_release_comp
```

Related Commands

Command	Description
debug tsp	Displays information about the telephony service provider.
debug voip rawmsg	Displays the raw message owner, length, and pointer.

debug txconn all

To turn on all debug flags for CTRC communications with CICS, use the **debug txconn all** privileged EXEC command. Use the **no** form of this command to disable all debugging output.

debug txconn all

no debug txconn all

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example shows the immediate output of the **debug txconn all** command. For examples of specific debugging messages, see the examples provided for the **debug txconn appc**, **debug txconn config**, **debug txconn data**, **debug txconn event**, **debug txconn tcp**, and **debug txconn timer** commands.

```
Router# debug txconn all

All possible TXConn debugging has been turned on
```

Related Commands

Command	Description
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn appc	Displays APPC-related trace or error messages for communications with CICS.
debug txconn config	Displays trace or error messages for CTRC configuration and control blocks for CICS communications.
debug txconn data	Displays CICS client and host data being handled by CTRC, in hexadecimal notation.
debug txconn event	Displays trace or error messages for CTRC events related to CICS communications.
debug txconn tcp	Displays error messages or traces for TCP/IP communications with CICS.
debug txconn timer	Displays performance information related to CICS communications.
show debugging	Displays the state of each debugging option.

debug txconn appc

To display APPC-related trace or error messages for communications with CICS, use the **debug txconn** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug txconn appc

no debug txconn appc

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example shows APPC debugging output from the **debug txconn appc** command:

```
01:18:05: TXCONN-APPC-622ADF38: Verb block =
01:18:05: TXCONN-APPC-622ADF38:  0001 0200 0300 0000 0400 0000 0000 0000
01:18:05: TXCONN-APPC-622ADF38:  0000 00FC 0000 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-622ADF38:  0000 0000 0840 0007 0000 0000 0000 0000
01:18:05: TXCONN-APPC-622ADF38:  7BC9 D5E3 C5D9 4040 07F6 C4C2 4040 4040
01:18:05: TXCONN-APPC-622ADF38:  4040 4040 4040 4040 4040 4040 4040 4040
01:18:05: TXCONN-APPC-622ADF38:  4040 4040 4040 4040 4040 4040 4040 4040
01:18:05: TXCONN-APPC-622ADF38:  4040 4040 4040 4040 4040 4040 4040 4040
01:18:05: TXCONN-APPC-622ADF38:  4040 4040 4040 4040 0000 0000 0000 0000
01:18:05: TXCONN-APPC-622ADF38:  0000 0000 0000 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-622ADF38:  0000 0000 0000 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-622ADF38:  00E2 E3C1 D9E6 4BC7 C1E9 C5D3 D3C5 4040
01:18:05: TXCONN-APPC-622ADF38:  4040 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730: Verb block =
01:18:05: TXCONN-APPC-621E5730:  0001 0200 0300 0000 0400 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730:  0000 00FD 0000 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730:  0000 0000 0840 0007 0000 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730:  C9C2 D4D9 C4C2 4040 07F6 C4C2 4040 4040
01:18:05: TXCONN-APPC-621E5730:  4040 4040 4040 4040 4040 4040 4040 4040
01:18:05: TXCONN-APPC-621E5730:  4040 4040 4040 4040 4040 4040 4040 4040
01:18:05: TXCONN-APPC-621E5730:  4040 4040 4040 4040 4040 4040 4040 4040
01:18:05: TXCONN-APPC-621E5730:  4040 4040 4040 4040 0000 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730:  0000 0000 0000 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730:  0000 0000 0000 0000 0000 0000 0000 0000
01:18:05: TXCONN-APPC-621E5730:  00E2 E3C1 D9E6 4BE2 E3C5 D3D3 C140 4040
01:18:05: TXCONN-APPC-621E5730:  4040 0000 0000 0000 0000 0000
```

Related Commands

Command	Description
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn all	Displays all CTRC debugging information related to communications with CICS.
debug txconn config	Displays trace or error messages for CTRC configuration and control blocks for CICS communications.
debug txconn data	Displays CICS client and host data being handled by CTRC, in hexadecimal notation.
debug txconn event	Displays trace or error messages for CTRC events related to CICS communications.
debug txconn tcp	Displays error messages or traces for TCP/IP communications with CICS.
debug txconn timer	Displays performance information related to CICS communications.
show debugging	Displays the state of each debugging option.

debug txconn config

To display trace or error messages for CTRC configuration and control blocks for CICS communications, use the **debug txconn config** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug txconn config

no debug txconn config

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example shows output for the **debug txconn config** command:

```
Router# debug txconn config

22:11:37: TXCONN-CONFIG: deleting transaction 61FCE414
22:11:37: TXCONN-CONFIG: deleting connection 61FB5CB0
22:11:37: TXCONN-CONFIG: server 62105D6C releases connection 61FB5CB0
22:11:44: TXCONN-CONFIG: new connection 61FB64A0
22:11:44: TXCONN-CONFIG: server 6210CEB4 takes connection 61FB64A0
22:11:44: TXCONN-CONFIG: new transaction 61E44B9C
22:11:48: TXCONN-CONFIG: deleting transaction 61E44B9C
22:11:53: TXCONN-CONFIG: new transaction 61E44B9C
22:11:54: TXCONN-CONFIG: deleting transaction 61E44B9C
```

Related Commands

Command	Description
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn all	Displays all CTRC debugging information related to communications with CICS.
debug txconn appc	Displays APPC-related trace or error messages for communications with CICS.
debug txconn data	Displays CICS client and host data being handled by CTRC, in hexadecimal notation.
debug txconn event	Displays trace or error messages for CTRC events related to CICS communications.
debug txconn tcp	Displays error messages or traces for TCP/IP communications with CICS.
debug txconn timer	Displays performance information related to CICS communications.
show debugging	Displays the state of each debugging option.

debug txconn data

To display a hexadecimal dump of CICS client and host data being handled by CTRC, plus information about certain CTRC internal operations, use the **debug txconn data** privileged EXEC command. Use the no form of this command to disable the debugging output.

debug txconn data

no debug txconn data

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example shows selected output from the **debug txconn data** command when a connection is established, data is received from the client via TCP/IP, data is sent to the client, and then the connection is closed.

```
Router# debug txconn data

TXConn DATA debugging is on

00:04:50: TXConn(62197464) Created
00:04:50: TXConn(62197464) State(0) MsgID(0) -> nextState(1)
00:04:50: TXConn(62197464) Client->0000 003A 0000 0002 000B 90A0
00:04:50: TXConn(62197464) Received LL 58 for session(0 0 2).
00:06:27: TXConn(62197464) Client<-0000 0036 0000 0003 000B 8001 0707 0864
00:06:53: TXConn(62175024) Deleted
```

The following lines show output when data is sent to the host:

```
00:04:50: TXTrans(id:62197910 conn:62197464 addr:2) LL(58) FMH5(0) CEBI(0)
00:04:50: TXTrans(id:62197910 conn:62197464 addr:2) State(0) MsgID(7844) -> nextState(1)
00:04:50: TXTrans(id:62197910 conn:62197464 addr:2) conversationType(mapped) syncLevel(1)
sec(0)
00:04:50: TXTrans(id:62197910 conn:62197464 addr:2) TPName CCIN
00:04:50: TXTrans(id:62197910 conn:62197464 addr:2) apDataLength(32) GDSID(12FF)

00:04:50: TXTrans(id:62197910 conn:62197464 addr:2) ->Host 0000 0008 03F4 F3F7 0000 0008
0401 0000
```

The following lines show output when data is received from the host:

```
00:05:01: TXTrans(id:62197910 conn:62197464 addr:2) <-Host  0092 12FF 0000 000C 0102 0000
0000 0002
```

The following lines show CTRC generating an FMH7 error message indicating that a CICS transaction has failed at the host or has been cleared by a router administrator:

```
00:06:27: TXTrans(id:6219853C conn:62197464 addr:3) Generating FMH7.
```

```
00:06:27: %TXCONN-3-TXEXCEPTION: Error occurred from transaction 3 of client
157.151.241.10 connected to server CICSC, exception type is 9
```

The following line shows CTRC responding to an FMH7 error message sent by the CICS client program:

```
00:07:11: TXTrans(id:62197910 conn:62197464 addr:2) Generating FMH7 +RSP.
```

Related Commands

Command	Description
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn all	Displays all CTRC debugging information related to communications with CICS.
debug txconn appc	Displays APPC-related trace or error messages for communications with CICS.
debug txconn config	Displays trace or error messages for CTRC configuration and control blocks for CICS communications.
debug txconn event	Displays trace or error messages for CTRC events related to CICS communications.
debug txconn tcp	Displays error messages or traces for TCP/IP communications with CICS.
debug txconn timer	Displays performance information related to CICS communications.
show debugging	Displays the state of each debugging option.

debug txconn event

To display trace or error messages for CTRC events related to CICS communications, use the **debug txconn event** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug txconn event

no debug txconn event

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example shows output for the **debug txconn event** command:

```
Router# debug txconn event

TXConn event debugging is on
Router#
22:15:08: TXCONN-EVENT: [*] Post to 62146464(cn), from 6211E744(tc), msg
61FC6170, msgid 0x6372 'cr', buffer 6211289C.
22:15:08: TXCONN-EVENT: Dispatch to 62146464, from 6211E744, msg 61FC6170,
msgid 6372 'cr', buffer 6211289C.
22:15:08: TXCONN-EVENT: [*] Post to 61E44BA0(sn), from 62146464(cn), msg
621164D0, msgid 0x7844 'xD', buffer 0.
22:15:08: TXCONN-EVENT: [*] Post to 6211E744(tc), from 62146464(cn), msg
61FC6170, msgid 0x6347 'cG', buffer 0.
22:15:08: TXCONN-EVENT: Dispatch to 61E44BA0, from 62146464, msg 621164D0,
msgid 7844 'xD', buffer 0.
22:15:08: TXCONN-EVENT: Dispatch to 6211E744, from 62146464, msg 61FC6170,
msgid 6347 'cG', buffer 0.
22:15:08: TXCONN-EVENT: [*] Post to 62146464(cn), from 6211E744(tc), msg
61FC6170, msgid 0x6372 'cr', buffer 6211289C.
22:15:08: TXCONN-EVENT: Dispatch to 62146464, from 6211E744, msg 61FC6170,
msgid 6372 'cr', buffer 6211289C.
22:15:08: TXCONN-EVENT: [*] Post to 61E44BA0(sn), from 62146464(cn), msg
61FBFBF4, msgid 0x7844 'xD', buffer 0.
22:15:08: TXCONN-EVENT: [*] Post to 6211E744(tc), from 62146464(cn), msg
61FC6170, msgid 0x6347 'cG', buffer 0.
22:15:08: TXCONN-EVENT: Dispatch to 61E44BA0, from 62146464, msg 61FBFBF4,
msgid 7844 'xD', buffer 0.
22:15:08: TXCONN-EVENT: [*] Post to 61FC6394(ap), from 61E44BA0(sn), msg
621164D0, msgid 0x634F 'cO', buffer 0.
22:15:08: TXCONN-EVENT: Dispatch to 6211E744, from 62146464, msg 61FC6170,
msgid 6347 'cG', buffer 0.
```

Related Commands

Command	Description
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn all	Displays all CTRC debugging information related to communications with CICS.
debug txconn appc	Displays APPC-related trace or error messages for communications with CICS.
debug txconn config	Displays trace or error messages for CTRC configuration and control blocks for CICS communications.
debug txconn data	Displays CICS client and host data being handled by CTRC, in hexadecimal notation.
debug txconn tcp	Displays error messages or traces for TCP/IP communications with CICS.
debug txconn timer	Displays performance information related to CICS communications.
show debugging	Displays the state of each debugging option.

debug txconn tcp

To display error messages and traces for TCP, use the **debug txconn tcp** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug txconn tcp

no debug txconn tcp

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example displays output from the **debug txconn tcp** command:

```
Router# debug txconn tcp

TXCONN-TCP-63528473: tcpdriver_passive_open returned NULL
TXCONN-TCP-63528473: (no memory) tcp_reset(63829482) returns 4
TXCONN-TCP: tcp_accept(74625348,&error) returns tcb 63829482, error 4
TXCONN-TCP: (no memory) tcp_reset(63829482) returns 4
TXCONN-TCP-63528473: (open) tcp_create returns 63829482, error = 4
TXCONN-TCP-63528473: tcb_connect(63829482,1.2.3.4,2010) returns 4
TXCONN-TCP-63528473: (open error) tcp_reset(63829482) returns 4
TXCONN-TCP-63528473: tcp_create returns 63829482, error = 4
TXCONN-TCP-63528473: tcb_bind(63829482,0.0.0.0,2001) returns 4
TXCONN-TCP-63528473: tcp_listen(63829482,,) returns 4
TXCONN-TCP-63528473: (errors) Calling tcp_close (63829482)
```

Related Commands

Command	Description
debug ip	Displays debugging information related to TCP/IP communications.
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn all	Displays all CTRC debugging information related to communications with CICS.
debug txconn appc	Displays APPC-related trace or error messages for communications with CICS.
debug txconn config	Displays trace or error messages for CTRC configuration and control blocks for CICS communications.
debug txconn data	Displays CICS client and host data being handled by CTRC, in hexadecimal notation.
debug txconn event	Displays trace or error messages for CTRC events related to CICS communications.
debug txconn timer	Displays performance information related to CICS communications.
show debugging	Displays the state of each debugging option.

debug txconn timer

To display performance information regarding CTRC communications with CICS, use the **debug txconn timer** privileged EXEC command. Use the **no** form of this command to disable the debugging output.

debug txconn timer

no debug txconn timer

Syntax Description

This command has no arguments or keywords.

Defaults

By default, debugging is not enabled for the txconn subsystem.

Command History

Release	Modification
12.0(5)XN	This command was introduced.

Examples

The following example shows turnaround time and host response time in milliseconds for a CICS transaction requested through CTRC. Turnaround time is measured from when CTRC receives the first request packet for the transaction until CTRC sends the last response packet of the transaction to the client. Host response time is measured from when CTRC sends the last request packet for a transaction to the host until CTRC receives the first response packet from the host for that transaction.

```
Router# debug txconn timer

TXConn timer debugging is on
00:04:14: TXTrans(id:622F4350 conn:62175024 addr:1) Turnaround Time = 4536(msec)
HostResponseTime = 120(msec)
```

Related Commands

Command	Description
debug snasw	Displays debugging information related to SNA Switching Services.
debug txconn all	Displays all CTRC debugging information related to communications with CICS.
debug txconn appc	Displays APPC-related trace or error messages for communications with CICS.
debug txconn config	Displays trace or error messages for CTRC configuration and control blocks for CICS communications.
debug txconn data	Displays CICS client and host data being handled by CTRC, in hexadecimal notation.
debug txconn event	Displays trace or error messages for CTRC events related to CICS communications.
debug txconn tcp	Displays error messages or traces for TCP/IP communications with CICS.
show debugging	Displays the state of each debugging option.

debug udptn

To display debug messages for UDPTN events, use the **debug udptn** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug udptn

no debug udptn

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0(5)T	This command was introduced.

Examples

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

```
terrapin# debug udptn

terrapin# udptn 172.16.1.1
Trying 172.16.1.1 ... Open

*Mar 1 00:10:15.191:udptn0:adding multicast group.
*Mar 1 00:10:15.195:udptn0:open to 172.16.1.1:57 Loopback0jjaassdd
*Mar 1 00:10:18.083:udptn0:output packet w 1 bytes
*Mar 1 00:10:18.087:udptn0:Input packet w 1 bytes
terrapin# disconnect
Closing connection to 172.16.1.1 [confirm] y
terrapin#
*Mar 1 00:11:03.139:udptn0:removing multicast group.
```

Related Commands

Command	Description
udptn	Enables transmission or reception of UDP packets.
transport output	Defines the protocol that can be used for outgoing connections from a line.

debug v120 event

To display information on V.120 activity, use the **debug v120 event** privileged EXEC command. The **no** form of this command disables debugging output.

debug v120 event

no debug v120 event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

V.120 is an ITU specification that allows for reliable transport of synchronous, asynchronous, or bit transparent data over ISDN bearer channels.

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

Examples

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

```
Router# debug v120 event

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

Related Commands

Command	Description
debug v120 packet	Displays general information on all incoming and outgoing V.120 packets.

debug v120 packet

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

debug v120 packet

no debug v120 packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

V.120 is an ITU specification that allows for reliable transport of synchronous, asynchronous, or bit transparent data over ISDN bearer channels.

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

Examples

The following is sample output from the **debug v120 packet** command for a typical session startup:

```
Router# debug v120 packet

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

Table 210 describes the significant fields in the display.

Table 210 debug v.120 packet Field Descriptions

Field	Descriptions
BR0:1	Interface number associated with this debugging information.
I/O	Packet going into or out of the interface.
SABME, UA, IFRAME, RR	V.120 packet type. In this case: • SABME—Set asynchronous balanced mode, extended • US—Unnumbered acknowledgment • IFRAME—Information frame • RR—Receive ready
lli 256	Logical link identifier number.
C/R 0	Command or response.
P/F=1	Poll final.
N(R)=0	Number received.
N(S)=0	Number sent.
len 43	Number of data bytes in the packet.
0x83	Up to 16 bytes of data.

Related Commands

Command	Description
debug tarp events	Displays information on TARP activity.

debug vg-anylan

To monitor error information and 100VG connection activity, use the **debug vg-anylan** privileged EXEC command. The **no** form of this command disables debugging output.

debug vg-anylan

no debug vg-anylan

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command could create a substantial amount of command output.

Examples

The following is sample output from the **debug vg-anylan** command:

```
Router# debug vg-anylan

%HP100VG-5-LOSTCARR: HP100VG(2/0), lost carrier
```

Table 211 lists the possible messages that could be generated by this command.

Table 211 debug vg-anylan Message Descriptions

Message	Description	Action
%HP100VG-5-LOSTCARR: HP100VG(2/0), lost carrier	Lost carrier debug message. The VG controller detects that the link to the hub is down due to cable, hub, or VG controller problem.	Check, repair, or replace the cable or hub. If you determine that the cable and hub are functioning normally, repair or replace the 100VG-AnyLAN port adapter.
%HP100VG-5-CABLEERR: HP100VG(2/0), cable error, training failed	Bad cable error messages. Cable did not pass training.[1]	Check, repair, or replace the cable or hub. If you determine that the cable and hub are functioning normally, repair or replace the 100VG-AnyLAN port adapter.
%HP100VG-5-NOCABLE: HP100VG(2/0), no tone detected, check cable, hub	No cable attached error message. The VG MAC cannot hear tones from the hub.[1]	Check, repair, or replace the cable or hub. If you determine that the cable and hub are functioning normally, repair or replace the 100VG-AnyLAN port adapter.

Table 211 debug vg-anylan Message Descriptions (continued)

Message	Description	Action
HP100VG-1-FAIL: HP100VG(2/0), Training Fail - unable to login to the hub	Training to the VG network failed. Login to the hub rejected by the hub.[1]	Take action based on the following error messages: • %HP100VG-1-DUPMAC: HP100VG(2/0), A duplicate MAC address has been detected • HP100VG-1-LANCNF: HP100VG(2/0), Configuration is not compatible with the network • %HP100VG-1-ACCESS: HP100VG(2/0), Access to network is not allowed
%HP100VG-1-DUPMAC : HP100VG(2/0), A duplicate MAC address has been detected	Duplicate MAC address on the same VG network. Two VG devices on the same LAN segment have the same MAC address.	Check the router configuration to make sure that no duplicate MAC address is configured.
%HP100VG-1-LANCNF: HP100VG(2/0), Configuration is not compatible with the network	Configuration of the router is not compatible to the network.	Check that the configuration of the hub for Frame Format, Promiscuous, and Repeater bit indicates the proper configuration.
%HP100VG-1-ACCESS: HP100VG(2/0), Access to network is not allowed	Access to the VG network is denied by the hub.	Check the configuration of the hub.
%HP100VG-3-NOTHP100VG: Device reported 0x5101A	Could not find the 100VG PCI device on a 100VG-AnyLAN port adapter.	Make sure the 100VG-AnyLAN port adapter is properly seated in the slot. Otherwise repair or replace the 100VG-AnyLAN port adapter.
%HP100VG-1-DISCOVER: Only found 0 interfaces on bay 2, shutting down bay	No 100VG interface detected on a 100VG-AnyLAN port adapter in a slot.	Make sure the 100VG-AnyLAN port adapter is properly seated in the slot. Otherwise repair or replace the 100VG-AnyLAN port adapter.

1. This message might display when the total load on the cascaded hub is high. Wait at least 20 seconds before checking to determine if the training really failed. Check if the protocol is up after 20 seconds before starting troubleshooting.

debug video vicm

To display debug messages for the Video Call Manager (ViCM) that handles video calls, enter the **debug video vicm** privileged EXEC command. The **no** form of the command disables ViCM debugging.

debug video vicm

no debug video vicm

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for the ViCM is not enabled.

Command History

Release	Modification
12.0(5)XK	This command was introduced.
12.0(6)T	This command was modified.

Examples

The following example shows output when you use the **debug video vicm** command. Comments are enclosed in asterisks (*).

```
Router# debug video vicm

Video ViCM FSM debugging is on

***** Starting Video call *****

Router# SVC HANDLE in rcvd:0x80001B:

00:42:55:ViCM - current state = Idle, Codec Ready
00:42:55:ViCM - current event = SVC Setup
00:42:55:ViCM - new state = Call Connected

00:42:55:ViCM - current state = Call Connected
00:42:55:ViCM - current event = SVC Connect Ack
00:42:55:ViCM - new state = Call Connected

*****Video Call Disconnecting*****

Router#
00:43:54:ViCM - current state = Call Connected
00:43:54:ViCM - current event = SVC Release
00:43:54:ViCM - new state = Remote Hangup

00:43:54:ViCM - current state = Remote Hangup
00:43:54:ViCM - current event = SVC Release Complete
00:43:54:ViCM - new state = Remote Hangup
mc3810_video_lw_periodic:Codec is not ready
mc3810_video_lw_periodic:sending message
00:43:55:ViCM - current state = Remote Hangup
```

```
00:43:55:ViCM - current event = DTR Deasserted
00:43:55:ViCM - new state = Idle
mc3810_video_lw_periodic:Codec is ready

mc3810_video_lw_periodic:sending message
00:43:55:ViCM - current state = Idle
00:43:55:ViCM - current event = DTR Asserted
00:43:55:ViCM - new state = Idle, Codec Ready
```

debug vines arp

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

debug vines arp

no debug vines arp

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug vines arp** command:

```
Router# debug vines arp

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

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

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

Table 212 describes the significant fields shown in the display.

Table 212 debug vines arp Field Descriptions

Field	Description
VNSARP:	Banyan VINES nonsequenced ARP message.
VSARP:	Banyan VINES sequenced ARP message.
received ARP type 0	ARP request of type 0 was received. Type values are as follow: • 0—Query request. The ARP client broadcasts a type 0 message to request an ARP service to respond. • 1—Service response. The ARP service responds with a type 1 message to an ARP client's query request. • 2—Assignment request. The ARP client responds to a service response with a type 2 message to request a VINES Internet address. • 3—Assignment response. The ARP service responds to an assignment request with a type 3 message that includes the assigned VINES Internet address.
from 0260.8c43.a7e4	Indicates the source address of the packet.

debug vines echo

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

debug vines echo

no debug vines echo

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

These echo packets do not include network-layer addresses.

Examples

The following is sample output from the **debug vines echo** command:

```
Router# debug vines echo

VINESECHO: 100 byte packet from 0260.8c43.a7e4
```

Table 213 describes the significant fields shown in the display.

Table 213 debug vines echo Field Descriptions

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

debug vines ipc

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

debug vines ipc

no debug vines ipc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

The following is sample output from the **debug vines ipc** command for three pairs of transactions. For more information about these fields or their values, refer to Banyan VINES documentation.

```
Router# debug vines ipc

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

Table 214 describes the significant fields shown in the display.

Table 214 debug vines ipc Field Descriptions

Field	Description
VIPC:	Indicates that this is output from the **debug vines ipc** command.
sending	Indicates that the router is either sending an IPC packet to another router or has received an IPC packet from another router.
IPC Data to	Indicates the type of IPC frame, as follows: • Acknowledgment • Data • Datagram • Disconnect • Error • Probe

Table 214 debug vines ipc Field Descriptions (continued)

Field	Description
Townsaver port 7	Indicates the machine name as assigned using the VINES **host** command, or IP address of the other router. Also indicates the port on that machine through which the packet has been sent.
from port 7	Indicates the port on the router through which the packet has been sent.
r_cid 0, l_cid 1, seq 1, ack 0, length 12	Indicates the values for various fields in the IPC layer header of this packet. Refer to Banyan VINES documentation for more information.

debug vines netrpc

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

debug vines netrpc

no debug vines netrpc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

The following is sample output from the **debug vines netrpc** command. For more information about these fields or their values, refer to Banyan VINES documentation.

```
Router# debug vines netrpc

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

Table 215 describes the significant fields shown in the display.

Table 215 debug vines netrpc Field Descriptions

Field	Description
VRPC:	Indicates that this is output from the **debug vines netrpc** command.
sending RPC	Indicates that the router is either sending a NetRPC packet to another router or has received a NetRPC packet from another router.
call	Indicates the transaction type as follows: • abort • call • reject • return • return address • search • search all
Townsaver	Indicates the machine name as assigned using the VINES **host** command or IP address of the other router.

debug vines packet

To display general Banyan VINES debugging information, such as packets received, generated, and forwarded, and failed access checks and other operations, use the **debug vines packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug vines packet

no debug vines packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug vines packet

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

Table 216 describes the fields shown in the first line of output.

Table 216 debug vines packet Field Descriptions

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

In the following line, the destination is the address 3002ABEA:1 associated with Ethernet interface 2. Source address 3000CBD4:1 sent a packet to this destination through the gateway at address 3000ABEA:1.

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

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

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

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

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

debug vines routing

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

debug vines routing [**verbose**]

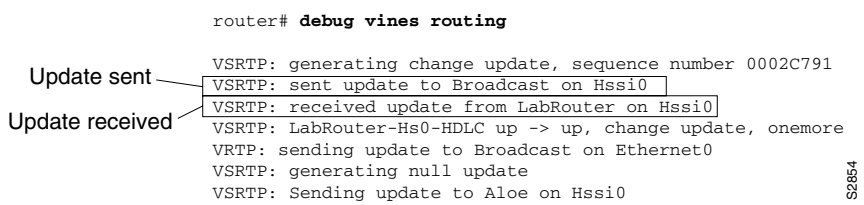
no debug vines routing [**verbose**]

Syntax Description

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

Examples

The following is sample output from the **debug vines routing** command:



The following is sample output from the **debug vines routing verbose** command:

```
Router# debug vines routing verbose

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

The output describes two VINES routing updates; the first includes two entries and the second includes three entries. Explanations for selected lines follow.

The following line shows that the router sent a periodic routing update to the broadcast address FFFFFFFF:FFFF through the Ethernet interface 0:

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

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

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

The following lines show that the router sent a change routing update to the Broadcast addresses on the Hssi interface 0 using the Sequenced Routing Update Protocol (SRTP) routing protocol:

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

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

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

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

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

The following line indicates that the router received a routing update from the router LabRouter through the Hssi interface 0:

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

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

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

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

```
network LabRouter from the server
```

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

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

debug vines service

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

debug vines service

no debug vines service

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

Examples

The following is sample output from the **debug vines service** command:

```
router# debug vines service

VSRV: Get Time Info sent to Townsaver
VSRV: Get Time Info response from Townsaver, time: 01:47:54 PDT Apr 29 1993
VSRV:     epoch SS@Aloe@Servers-10, age: 0:15:15
```

Sent/ Response pair

S2565

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

Table 217 describes the fields shown in the second line of output. For more information about these fields or their values, refer to Banyan VINES documentation.

Table 217 debug vines service Field Descriptions

Field	Description
VSRV:	Indicates that this is output from the **debug vines service** command.
Get Time Info	Indicates one of three packet types, as follows: • Get Time Info • Time Set • Time Sync
response from	Indicates whether the packet was sent to another router, a response from another router, or received from another router.

Table 217 debug vines service Field Descriptions (continued)

Field	Description
Townsaver	Indicates the machine name as assigned using the VINES **host** command, or IP address of the other router.
time: 01:47:54 PDT Apr 29 1993	Indicates the current time (in hours:minutes:seconds) and current date.

Table 218 describes the fields shown in the third line of output. This line is an extension of the first two lines of output. For more information about these fields or their values, refer to Banyan VINES documentation.

Table 218 debug vines service Field Descriptions

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

debug vines state

To display information on the Banyan VINES SRTP state machine transactions, use the **debug vines state** privileged EXEC command. The **no** form of this command disables debugging output.

debug vines state

no debug vines state

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

debug vines table

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

debug vines table

no debug vines table

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

The following is sample output from the **debug vines table** command:

```
Router# debug vines table

VINESRTP: create neighbor 3001153C:8004, interface Ethernet0
```

Table 219 describes the significant fields in the display.

Table 219 debug vines table Field Descriptions

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

debug vlan packet

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

debug vlan packet

no debug vlan packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

The following is sample output from the **debug vlan packet** output. In this example, a VLAN packet with a VLAN ID of 1000 was received on FDDI interface 0 and this interface was not configured to route or switch this VLAN packet:

```
Router# debug vlan packet

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

debug voice all

To display debugging information for all components of the Voice Call Manager, use the **debug voice all** privileged EXEC command. The **no** form of this command disables debugging output.

debug voice all [*slot/port*]

no debug voice all [*slot/port*]

Syntax Description

slot/port	(Optional) The slot and port number of the voice port. If the *slot/port* argument is entered, then only debugging information for that voice port is displayed. If the *slot/port* is not entered, debugging information for all voice ports is displayed.

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

The **debug voice all** command output provides debug output for all the debug commands for the Voice Call Manager compiled into one display. For sample output of the individual commands, see the sample displays for the **debug voice cp, debug voice eecm, debug voice protocol, debug voice signaling**, and **debug voice tdsm** commands.

Related Commands

Command	Description
debug voip ccapi	Debugs the call control API.
debug voice eecm	Displays debugging information for the Voice End-to-End Call Manager.
debug voice protocol	Displays debugging information for the Voice Line Protocol State machine.
debug voice signaling	Displays debugging information for the voice port signalling.
debug voice tdsm	Displays debugging information for the voice tandem switch.

debug voice cp

To display debugging information for the Voice Call Processing State Machine, use the **debug voice cp** privileged EXEC command. The **no** form of this command disables debugging output.

debug voice cp [*slot/port*]

no debug voice cp [*slot/port*]

Syntax Description

slot/port	(Optional) The slot and port number of the voice port. If the *slot/port* argument is entered, then only debugging information for that voice port is displayed.

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

The following is sample output from the **debug voice cp** command:

```
Router# debug voice cp 1/1

Voice Call Processing State Machine debugging is on

1/1: CPD( ), idle gets event seize_ind
1/1: CPD( ), idle gets event dsp_ready
1/1: CPD( ), idle ==> collect
1/1: CPD(in), collect gets event digit
1/1: CPD(in), collect gets event digit
1/1: CPD(in), collect gets event digit
1/1: CPD(in), collect gets event digit
1/1: CPD(in), collect gets event addr_done
1/1: CPD(in), collect ==> request
1/1: CPD(in), request gets event call_proceeding
1/1: CPD(in), request ==> in_wait_answer
1/1: CPD(in), in_wait_answer gets event call_accept
1/1: CPD(in), in_wait_answer gets event call_answered
1/1: CPD(in), in_wait_answer ==> connected
1/1: CPD(in), connected gets event peer_onhook
1/1: CPD(in), connected ==> disconnect_wait
1/1: CPD(in), disconnect_wait gets event idle_ind
1/1: CPD(in), disconnect_wait ==> idle
```

Related Commands

Command	Description
debug voice all	Displays debugging information for all components of the Voice Call Manager.
debug voice eecm	Displays debugging information for the Voice End-to-End Call Manager.
debug voice protocol	Displays debugging information for the Voice Line protocol State machine.
debug voice signaling	Displays debugging information for the voice port signalling.
debug voice tdsm	Displays debugging information for the voice tandem switch.

debug voice eecm

To display debugging information for the Voice End-to-End Call Manager, use the **debug voice eecm** privileged EXEC command. The **no** form of this command disables debugging output.

debug voice eecm [*slot/port*]

no debug voice eecm [*slot/port*]

Syntax Description

slot/port	(Optional) Slot and port number of the voice port. If the *slot/port* is entered, then only debugging information for that voice port is displayed.

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

The following is sample output from the **debug voice eecm** command:

```
Router# debug voice eecm

1/1: EECM(in), ST_NULL        EV_ALLOC_DSP
1/1: EECM(in), ST_DIGIT_COLLECT  EV_PARSE_DIGIT  3
1/1: EECM(in), ST_DIGIT_COLLECT  EV_PARSE_DIGIT  7
1/1: EECM(in), ST_DIGIT_COLLECT  EV_PARSE_DIGIT  0
1/1: EECM(in), ST_DIGIT_COLLECT  EV_PARSE_DIGIT  2
1/1: EECM(in), ST_ADDRESS_DONE   EV_OUT_SETUP
-1/-1: EECM(out), ST_NULL     EV_IN_SETUP
1/1: EECM(in), ST_OUT_REQUEST   EV_IN_PROCEED
1/2: EECM(out), ST_SEIZE      EV_ALLOC_DSP
1/2: EECM(out), ST_SEIZE      EV_OUT_ALERT
1/1: EECM(in), ST_OUT_REQUEST   EV_IN_ALERT
1/1: EECM(in), ST_OUT_REQUEST   EV_OUT_ALERT_ACK
1/2 EECM(out), ST_IN_PENDING   EV_OUT_CONNECT
1/1: EECM(in), ST_WAIT_FOR_ANSWER EV_IN_CONNECT
1/2: EECM(out), ST_ACTIVE     EV_OUT_REL
1/1: EECM(in), ST_ACTIVE      EV_IN_REL
1/1: EECM(in), ST_DISCONN_PENDING EV_OUT_REL_ACK
```

Related Commands

Command	Description
debug voice all	Displays debugging information for all components of the Voice Call Manager.
debug voip ccapi	Debugs the call control API.
debug voice protocol	Displays debugging information for the Voice Line protocol State machine.
debug voice signaling	Displays debugging information for the voice port signalling.
debug voice tdsm	Displays debugging information for the voice tandem switch.

debug voice protocol

To display debugging information for the Voice Line protocol State machine, use the **debug voice protocol** privileged EXEC command. The **no** form of this command disables debugging output.

debug voice protocol [*slot/port*]

no debug voice protocol [*slot/port*]

Syntax Description

slot/port	(Optional) Slot/port number of the voice port. If the *slot/port* is entered, then only debugging information for that voice port is displayed.

Usage Guidelines

In the debugging display, the following abbreviations are used for the different signalling protocols:

LFXS	FXS trunk loop start protocol.
LFXO	FXO trunk loop start protocol.
GFXS	FXS trunk ground start protocol.
GFXO	FXO trunk ground start protocol.
E&M	E&M trunk protocol.

Command History

This command is valid on the Cisco MC3810 device only.

Examples

The following is sample output from the **debug voice protocol** command:

```
Router# debug voice protocol

Voice Line protocol State machine debugging is on

1/1: LFXS( ), idle gets event offhook
1/1: LFXS( ), idle ==> seize
1/1: LFXS(in), seize gets event ready
1/1: LFXS(in), seize ==> dial_tone
1/1: LFXS(in), dial_tone gets event digit
1/1: LFXS(in), dial_tone ==> collect
1/1: LFXS(in), collect gets event digit
1/1: LFXS(in), collect gets event digit
1/1: LFXS(in), collect gets event digit
1/1: LFXS(in), collect gets event addr_done
1/1: LFXS(in), collect ==> call_progress
1/2: LFXS( ), idle gets event seize
1/2: LFXS( ), idle ==> ringing
1/2: LFXS(out), ringing gets event dial_tone
1/2: LFXS(out), ringing gets event offhook
1/2: LFXS(out), ringing ==> connected
1/1: LFXS(in), call_progress gets event answer
1/1: LFXS(in), call_progress ==> connected
1/2: LFXS(out), connected gets event onhook
1/2: LFXS(out), connected ==> disconnect_wait
1/2: LFXS(out), disconnected_wait gets event disconnect
```

```
1/2: LFXS(out), disconnect_wait ==> cpc
1/1: LFXS(in), connected gets event disconnect
1/2: LFXS(out), connected ==> cpc
1/2: LFXS(out), cpc gets event offhook
1/2: LFXS(out), cpc gets event timer1
1/2: LFXS(out), cpc ==> cpc_recover
1/2: LFXS(out), cpc gets event timer1
1/2: LFXS(out), cpc_recover ==> offhook_wait
1/1: LFXS(in), offhook_wait gets event onhook
1/1: LFXS(in), offhook_wait ==> idle
1/2: LFXS(out), offhook_wait gets event onhook
1/2: LFXS(out), offhook_wait ==> idle
```

Related Commands

Command	Description
debug voice all	Displays debugging information for the voice tandem switch.
debug voip ccapi	Debugs the call control API.
debug voice eecm	Displays debugging information for the Voice End-to-End Call Manager.
debug voice signaling	Displays debugging information for the voice port signalling.
debug voice tdsm	Displays debugging information for the voice tandem switch.

debug voice signaling

To display debugging information for the voice port signalling, use the **debug voice signaling** privileged EXEC command. The **no** form of this command disables debugging output.

debug voice signaling [*slot/port*]

no debug voice signaling [*slot/port*]

Syntax Description

slot/port	(Optional) Slot and port number of the voice port. If the *slot/port* argument is entered, then only debugging information for that voice port is displayed.

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

The following is sample output from the **debug voice signaling** command:

```
Router# debug voice signaling

1/1: TIU, report_local_hook=1
1/2: TIU, set ring cadence=1
1/2: TIU, ringer on
1/2: TIU, ringer off
1/2: TIU, ringer on
1/2: TIU, report_local_hook=1
1/2: TIU, turning off ringer due to SW ringtrip
1/2: TIU, ringer off
1/2: TIU, set ring cadence=0
1/2: TIU, ringer off
1/2: TIU, set reverse battery=1
1/2: TIU, set reverse battery=1
1/1: TIU, report_local_hook=0
1/2: TIU, set reverse battery=0
1/2: TIU, set loop disabled=1
1/1: TIU, set reverse battery=0
1/1: TIU, set loop disabled=1
1/2: TIU, report_local_hook=1
1/1: TIU, report_lead_gnd grounded=1
1/1: TIU, report_lead_gnd grounded=0
1/2: TIU, set loop disabled=0
1/1: TIU, set loop disabled=0
1/1: TIU, report_local_hook=0
1/2: TIU, report_local_hook=0
1/1: TIU, report_local_hook=1
1/2: TIU, report_local_hook=1
1/1: TIU, report_local_hook=0
1/2: TIU, report_local_hook=0
1/1: TIU, set reverse battery=0
1/2: TIU, set reverse battery=0
```

Related Commands

Command	Description
debug voice all	Displays debugging information for all components of the Voice Call Manager.
debug voip ccapi	Debugs the call control API.
debug voice eecm	Displays debugging information for the Voice End-to-End Call Manager.
debug voice protocol	Displays debugging information for the Voice Line protocol State machine.
debug voice tdsm	Display debugging information for the voice tandem switch.

debug voice tdsm

To display debugging information for the voice tandem switch, use the **debug voice tdsm** privileged EXEC command. The **no** form of this command disables debugging output.

debug voice tdsm [*slot/port*]

no debug voice tdsm [*slot/port*]

Syntax Description

slot/port	(Optional) Slot and port number of the voice port. If the *slot/port* argument is entered, then only debugging information for that voice port is displayed.

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

The following is sample output from the **debug voice tdsm** command:

```
Router# debug voice tdsm

Voice tandem switch debugging is on

-1/-1: TDSM(out), ref= -1, state NULL gets event OUT_SETUP
1/1: TDSM(in), ref=6, state CALL_INITIATED gets event IN_CALLPROC
1/1: TDSM(in), ref=6, state OUTG_CALLPROC gets event IN_ALERTING
1/1: TDSM(in), ref=6, state CALL_DELIVERED gets event IN_CONNECT
1/1: TDSM(out),ref=6, state CALL_ACTIVE send out conn. ack
1/1: TDSM(out),ref=6, state CALL_ACTIVE send out release, cause LOCAL_ONHOOK
1/1: TDSM(in), ref=6, state RELEASE_REQ gets event IN_REL_COMP, cause REMOTE_ONHOOK
-1/-1: TDSM(in), ref=-1, state NULL gets event IN_SETUP
-1/-1: TDSM(out), ref=6, state INC_CALLPROC gets event OUT_ALERTING
1/1: TDSM(out),ref=6, state CALL_RECEIVED gets event OUT_CONNECT
1/1: TDSM(in), ref-6, state CONNECT_REQ gets event IN_CONN_ACK
1/1: TDSM(out),ref-6, state CALL_ACTIVE send out release, cause LOCAL_ONHOOK
1/1: TDSM(in), ref=6, state RELEASE_REQ gets event IN_REL_COMP, cause REMOTE_ONHOOK
-1/-1:TDSM(out), ref=-1, state NULL gets event OUT_SETUP
1/1: TDSM(in), ref=7, state CALL_INITIATED gets event IN_CALLPROC
1/1: TDSM(in), ref=7, state OUTG_CALLPROC gets event IN_ALERTING
1/1: TDSM(in), ref=7, state CALL_DELIVERED gets event IN_CONNECT
1/1: TDSM(out),ref=7, state CALL_ACTIVE send out conn.ack
1/1: TDSM(out),ref=7, state CALL_ACTIVE send out release, cause LOCAL_ONHOOK
-1/-1: TDSM(in), ref=-1, state NULL gets event IN_SETUP
-1/-1: TDSM(out), ref=7, state INC_CALLPROC gets event OUT_ALERTING
1/1: TDSM(out),ref=7. state CALL_RECEIVED gets event OUT_CONNECT
1/1: TDSM(in), ref=7, state CONNECT_REQ gets event IN_CONN_ACK
1/1: TDSM(in), ref=7, state CALL_ACTIVE send out release, cause LOCAL_ONHOOK
1/1: TDSM(in), ref=7, state RELEASE_REQ gets event IN_REL_COMP, cause REMOTE_ONHOOK
-1/-1: TDSM(out), ref=-1, state NULL gets event OUT_SETUP
1/1: TDSM(in), ref=8, state CALL_INITIATED gets event IN_CALLPROC
1/1: TDSM(in), ref=8, state OUTG_CALLPROC gets event IN_ALERTINGbug all
```

Related Commands

Command	Description
debug voice all	Displays debugging information for all components of the Voice Call Manager.
debug voip ccapi	Debugs the call control API.
debug voice eecm	Displays debugging information for the Voice End-to-End Call Manager.
debug voice protocol	Displays debugging information for the Voice Line protocol State machine.
debug voice signaling	Displays debugging information for the voice port signalling.

debug voice vofr

To show Cisco trunk and FRF.11 trunk call setup attempts and to show which dial peer is used in the call setup, use the **debug voice vofr** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug voice vofr

no debug voice vofr

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced.

Usage Guidelines

This command applies to Cisco trunks and FRF.11 trunks only; it does not apply to switched calls.

This command applies to VoFR, VoATM, and VoHDLC dial peers on the Cisco MC3810 device.

Examples

The following example shows sample output from the **debug voice vofr** command for a Cisco trunk:

```
Router# debug voice vofr

1d05h: 1/1:VOFR, unconf ==> pending_start
1d05h: 1/1:VOFR,create VOFR
1d05h: 1/1:VOFR,search dial-peer 7100 preference 0
1d05h: 1/1:VOFR, pending_start ==> start
1d05h: 1/1:VOFR,
1d05h:voice_configure_perm_svc:
1d05h:dial-peer 7100 codec = G729A payload size = 30 vad = off dtmf relay = on
   seq num = off
1d05h:voice-port 1/1 codec = G729A payload size = 30 vad = off dtmf relay = on
   seq num = off
1d05h: 1/1:VOFR,SIGNAL-TYPE = cept
1d05h:init_frf11 tcid 0 master 0 signaltype 2
1d05h:Going Out Of Service on tcid 0 with sig state 0001
1d05h: 1/1:VOFR, start get event idle
1d05h: 1/1:VOFR, start get event
1d05h: 1/1:VOFR, start get event set up
1d05h: 1/1:VOFR, start ==> pending_connect
1d05h: 1/1:VOFR, pending_connect get event connect
1d05h: 1/1:VOFR, pending_connect ==> connect
1d05h: 1/1:VOFR,SIGNAL-TYPE = cept
1d05h:init_frf11 tcid 0 master 1 signaltype 2
1d05h:start_vofr_polling on port 0 signaltype 2
```

The following example shows sample output from the **debug voice vofr** command for an FRF.11 trunk:

```
Router# debug voice vofr

1d05h: 1/1:VOFR,search dial-peer 7200 preference 2
1d05h: 1/1:VOFR,SIGNAL-TYPE = cept
1d05h:Launch Voice Trunk:signal-type 2
```

```
1d05h:calculated bandwidth = 10, coding = 6, size = 30
1d05h:%Voice-port 1/1 is down.
1d05h: 1/1:VOFR, pending_start get event idle
1d05h:Codec Type = 6 Payload Size = 30 Seq# off
1d05h:%Voice-port 1/1 is up.
1d05h:init_frf11 tcid 0 master 1 signaltype 2
1d05h:status OK :cid = 100
1d05h: 1/1:VOFR,
1d05h:start FRF11
1d05h: 1/1:VOFR, pending_start ==> frf11
1d05h: 1/1:VOFR,SIGNAL-TYPE = cept
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccsip all	Displays the ccswvoice function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug frame-relay fragment	Displays information related to Frame Relay fragmentation on a PVC.
debug vpm error	Displays the behavior of the Holst state machine.
debug vtsp port	Displays the behavior of the VTSP state machine.
debug vtsp vofr subframe	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug voip aaa

To enable debugging messages for gateway aaa to be output to the system console, use the **debug voip aaa** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug voip aaa

no debug voip aaa

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.

debug voip ccapi

To debug the call control API, use the **debug voip ccapi** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug voip ccapi

no debug voip ccapi

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.

Examples

The following is sample output for the **debug voip ccapi** command.

```
Router# show debug

voip:
 voip ccAPI function enter/exit debugging is on
Oct 9 17:39:20.267:cc_api_call_setup_ind (vdbPtr=0x60ED5134,
callInfo={called=3001, calling=4004, fdest=0 peer_tag=1},
callID=0x6104B374)
Oct 9 17:39:20.275:cc_process_call_setup_ind (event=0x60D45CF0) handed
call to app "sess"
Oct 9 17:39:20.279:ccAppInitialize (name=App for callId 3
, appHandle=0x6103DD44)
Oct 9 17:39:20.279:ccCallSetContext (callID=0x3, context=0x6103DD3C)
Oct 9 17:39:20.279:ccCallSetupAck (callID=0x3)
Oct 9 17:39:20.279:ccGenerateTone (callID=0x3 tone=8)
Oct 9 17:39:20.279:ccCallApp (callID=0x3)
Oct 9 17:39:20.279:ccCallSetContext (callID=0x3, context=0x60DC4594)
00:11:31:%RADIUS-6-SERVERALIVE:Radius server 171.69.184.73 is
responding
again (previously dead).
Oct 9 17:39:22.808:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=1, mode=0)
Oct 9 17:39:23.069:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=1, mode=0)
Oct 9 17:39:23.399:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=5, mode=0)
Oct 9 17:39:23.652:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=1, mode=0)
Oct 9 17:39:24.041:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=0, mode=0)
Oct 9 17:39:24.294:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=0, mode=0)
Oct 9 17:39:24.294:ccCallAppReturn (callID=0x3)
Oct 9 17:39:24.294:ccCallApp (callID=0x3)
Oct 9 17:39:24.294:ccCallSetContext (callID=0x3, context=0x6105DC90)
Oct 9 17:39:24.294:ccCallProceeding (callID=0x3, prog_ind=0x0)
Oct 9 17:39:24.294:ccCallSetupRequest (peer=0x60FE4068, dest=,
params=0x6105DB70 mode=0, *callID=0x60D50978)
Oct 9 17:39:24.294:callingNumber=4004, calledNumber=115100,
redirectNumber=
```

```
Oct 9 17:39:24.294:accountNumber=, finalDestFlag=0,
guid=3c85.5d28.2861.0004.0000.0000.000a.8dfc
Oct 9 17:39:24.294:peer_tag=115
Oct 9 17:39:24.294:ccIFCallSetupRequest:(vdbPtr=0x60D4A268, dest=,
callParams={called=115100, calling=4004, fdest=0, voice_peer_tag=115},
mode=0x0)
Oct 9 17:39:24.294:ccCallSetContext (callID=0x4, context=0x6105DD78)
Oct 9 17:39:26.350:cc_api_call_alert(vdbPtr=0x60D4A268, callID=0x4,
prog_ind=0x8, sig_ind=0x0)
Oct 9 17:39:26.350:ccCallAlert (callID=0x3, prog_ind=0x8, sig_ind=0x0)
Oct 9 17:39:26.350:ccConferenceCreate (confID=0x60D509C8, callID1=0x3,
callID2=0x4, tag=0x0)
Oct 9 17:39:26.350:cc_api_bridge_done (confID=0x1, srcIF=0x60D4A268,
srcCallID=0x4, dstCallID=0x3, disposition=0, tag=0x0)
Oct 9 17:39:26.350:cc_api_bridge_done (confID=0x1, srcIF=0x60ED5134,
srcCallID=0x3, dstCallID=0x4, disposition=0, tag=0x0)
Oct 9 17:39:26.350:cc_api_caps_ind (dstVdbPtr=0x60D4A268,
dstCallId=0x4,srcCallId=0x3, caps={codec=0x7, fax_rate=0x7F, vad=0x3})
Oct 9 17:39:26.350:cc_api_caps_ind (dstVdbPtr=0x60ED5134,
dstCallId=0x3,srcCallId=0x4, caps={codec=0x4, fax_rate=0x2, vad=0x2})
Oct 9 17:39:26.350:cc_api_caps_ack (dstVdbPtr=0x60ED5134,
dstCallId=0x3,srcCallId=0x4, caps={codec=0x4, fax_rate=0x2, vad=0x2})
Oct 9 17:39:26.350:cc_api_caps_ack (dstVdbPtr=0x60D4A268,
dstCallId=0x4,srcCallId=0x3, caps={codec=0x4, fax_rate=0x2, vad=0x2})
Oct 9 17:39:26.430:cc_api_call_connected(vdbPtr=0x60D4A268,
callID=0x4)
Oct 9 17:39:26.430:ccCallConnect (callID=0x3)
Oct 9 17:39:26.430:ccCallAppReturn (callID=0x3)
Oct 9 17:39:26.430:ccCallSetContext (callID=0x4, context=0x6103DD3C)
Oct 9 17:39:30.683:cc_api_call_disconnected(vdbPtr=0x60D4A268,
callID=0x4, cause=0x10)
Oct 9 17:39:30.683:ccCallDisconnect (callID=0x4, cause=0x10 tag=0x0)
Oct 9 17:39:30.683:ccConferenceDestroy (confID=0x1, tag=0x0)
Oct 9 17:39:30.687:cc_api_bridge_done (confID=0x1, srcIF=0x60D4A268,
srcCallID=0x4, dstCallID=0x3, disposition=0 tag=0x0)
Oct 9 17:39:30.727:cc_api_call_disconnect_done(vdbPtr=0x60D4A268,
callID=0x4, disp=0, tag=0x0)
Oct 9 17:39:30.727:cc_api_bridge_done (confID=0x1, srcIF=0x60ED5134,
srcCallID=0x3, dstCallID=0x4, disposition=0 tag=0x0)
Oct 9 17:39:30.727:ccCallDisconnect (callID=0x3, cause=0x10 tag=0x0)
Oct 9 17:39:30.779:cc_api_call_disconnect_done(vdbPtr=0x60ED5134,
callID=0x3, disp=0, tag=0x0)
00:11:42:%LINK-3-UPDOWN:Interface Serial0:18, changed state to down
```

debug voip ccapi error

To trace error logs in the call control API, use the **debug voip ccapi error** privileged EXEC command. The **no** form of this command disables debugging output.

debug voip ccapi error

no debug voip ccapi error

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug voip ccapi error** command traces the error logs in the call control API. Error logs are generated during normal call processing, when there are insufficient resources, or when there are problems in the underlying network-specific code, the higher call session application, or the call control API itself.

This debug command shows error events or unexpected behavior in system software. In most cases, no events will be generated.

debug voip ccapi inout

To trace the execution path through the call control API, use the **debug voip ccapi inout** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug voip ccapi inout

no debug voip ccapi inout

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug voip ccapi inout** command traces the execution path through the call control API, which serves as the interface between the call session application and the underlying network-specific software. You can use the output from this command to understand how calls are being handled by the router.

This command shows how a call flows through the system. Using this debug level, you can see the call setup and teardown operations performed on both the telephony and network call legs.

Examples

The following example shows the call setup indicated and accepted by the router:

```
Router# debug voip ccapi inout

cc_api_call_setup_ind (vdbPtr=0x60BFB530, callInfo={called=, calling=, fdest=0},
callID=0x60BFAEB8)
cc_process_call_setup_ind (event=0x60B68478)
sess_appl: ev(14), cid(1), disp(0)
ccCallSetContext (callID=0x1, context=0x60A7B094)
ccCallSetPeer (callID=0x1, peer=0x60C0A868, voice_peer_tag=2, encapType=1,
dest-pat=+14085231001, answer=)
ccCallSetupAck (callID=0x1)
```

The following example shows the caller entering DTMF digits until a dial-peer is matched:

```
cc_api_call_digit (vdbPtr=0x60BFB530, callID=0x1, digit=4, mode=0)
sess_appl: ev(8), cid(1), disp(0)
ssa: cid(1)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0)
cc_api_call_digit (vdbPtr=0x60BFB530, callID=0x1, digit=1, mode=0)
sess_appl: ev(8), cid(1), disp(0)
ssa: cid(1)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0)
cc_api_call_digit (vdbPtr=0x60BFB530, callID=0x1, digit=0, mode=0)
sess_appl: ev(8), cid(1), disp(0)
ssa: cid(1)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0)
cc_api_call_digit (vdbPtr=0x60BFB530, callID=0x1, digit=0, mode=0)
sess_appl: ev(8), cid(1), disp(0)
ssa: cid(1)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0)
cc_api_call_digit (vdbPtr=0x60BFB530, callID=0x1, digit=1, mode=0)
sess_appl: ev(8), cid(1), disp(0)
```

```
ssa: cid(1)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0)
ccCallProceeding (callID=0x1, prog_ind=0x0)
ssaSetupPeer cid(1), destPat(+14085241001), matched(8), prefix(), peer(60C0E710)
```

The following example shows the call setup over the IP network to the remote router:

```
ccCallSetupRequest (peer=0x60C0E710, dest=, params=0x60A7B0A8 mode=0, *callID=0x60B6C110)
ccIFCallSetupRequest: (vdbPtr=0x60B6C5D4, dest=, callParams={called=+14085241001,
calling=+14085231001, fdest=0, voice_peer_tag=104}, mode=0x0)
ccCallSetContext (callID=0x2, context=0x60A7B2A8)
```

The following example shows the called party is alerted, a CODEC is negotiated, and voice path is cut through:

```
cc_api_call_alert(vdbPtr=0x60B6C5D4, callID=0x2, prog_ind=0x8, sig_ind=0x1)
sess_appl: ev(6), cid(2), disp(0)
ssa: cid(2)st(1)oldst(0)cfid(-1)csize(0)in(0)fDest(0)-cid2(1)st2(1)oldst2(0)
ccCallAlert (callID=0x1, prog_ind=0x8, sig_ind=0x1)
ccConferenceCreate (confID=0x60B6C150, callID1=0x1, callID2=0x2, tag=0x0)
cc_api_bridge_done (confID=0x1, srcIF=0x60B6C5D4, srcCallID=0x2, dstCallID=0x1,
disposition=0, tag=0x0)
cc_api_bridge_done (confID=0x1, srcIF=0x60BFB530, srcCallID=0x1, dstCallID=0x2,
disposition=0, tag=0x0)
cc_api_caps_ind (dstVdbPtr=0x60B6C5D4, dstCallId=0x2,srcCallId=0x1, caps={codec=0x7,
fax_rate=0x7F, vad=0x3})
cc_api_caps_ind (dstVdbPtr=0x60BFB530, dstCallId=0x1,srcCallId=0x2, caps={codec=0x4,
fax_rate=0x2, vad=0x2})
cc_api_caps_ack (dstVdbPtr=0x60BFB530, dstCallId=0x1,srcCallId=0x2, caps={codec=0x4,
fax_rate=0x2, vad=0x2})
cc_api_caps_ack (dstVdbPtr=0x60B6C5D4, dstCallId=0x2,srcCallId=0x1, caps={codec=0x4,
fax_rate=0x2, vad=0x2})
sess_appl: ev(17), cid(1), disp(0)
ssa: cid(1)st(3)oldst(0)cfid(1)csize(0)in(1)fDest(0)-cid2(2)st2(3)oldst2(1)
```

The following example shows that the call is connected and voice is active:

```
cc_api_call_connected(vdbPtr=0x60B6C5D4, callID=0x2)
sess_appl: ev(7), cid(2), disp(0)
ssa: cid(2)st(4)oldst(1)cfid(1)csize(0)in(0)fDest(0)-cid2(1)st2(4)oldst2(3)
ccCallConnect (callID=0x1)
```

The following example shows how the system processes voice statistics and monitors voice quality during the call:

```
ccapi_request_rt_packet_stats (requestorIF=0x60B6C5D4, requestorCID=0x2,
   requestedCID=0x1, tag=0x60A7C598)
cc_api_request_rt_packet_stats_done (requestedIF=0x60BFB530, requestedCID=0x1,
   tag=0x60A7A4C4)
ccapi_request_rt_packet_stats (requestorIF=0x60B6C5D4, requestorCID=0x2,
   requestedCID=0x1, tag=0x60A7C598)
cc_api_request_rt_packet_stats_done (requestedIF=0x60BFB530, requestedCID=0x1,
   tag=0x60C1FE54)
ccapi_request_rt_packet_stats (requestorIF=0x60B6C5D4, requestorCID=0x2,
   requestedCID=0x1, tag=0x60A7C598)
cc_api_request_rt_packet_stats_done (requestedIF=0x60BFB530, requestedCID=0x1,
   tag=0x60A7A5F4)
ccapi_request_rt_packet_stats (requestorIF=0x60B6C5D4, requestorCID=0x2,
   requestedCID=0x1, tag=0x60A7C598)
cc_api_request_rt_packet_stats_done (requestedIF=0x60BFB530, requestedCID=0x1,
   tag=0x60A7A6D8)
```

```
ccapi_request_rt_packet_stats (requestorIF=0x60B6C5D4, requestorCID=0x2,
   requestedCID=0x1, tag=0x60A7C598)
cc_api_request_rt_packet_stats_done (requestedIF=0x60BFB530, requestedCID=0x1,
   tag=0x60A7ACBC)
```

The following example shows that disconnection is indicated from the calling party, call legs are torn down and disconnected:

```
cc_api_call_disconnected(vdbPtr=0x60BFB530, callID=0x1, cause=0x10)
sess_appl: ev(9), cid(1), disp(0)
ssa: cid(1)st(5)oldst(3)cfid(1)csize(0)in(1)fDest(0)-cid2(2)st2(5)oldst2(4)
ccConferenceDestroy (confID=0x1, tag=0x0)
cc_api_bridge_done (confID=0x1, srcIF=0x60B6C5D4, srcCallID=0x2, dstCallID=0x1,
disposition=0 tag=0x0)
cc_api_bridge_done (confID=0x1, srcIF=0x60BFB530, srcCallID=0x1, dstCallID=0x2,
disposition=0 tag=0x0)
sess_appl: ev(18), cid(1), disp(0)
ssa: cid(1)st(6)oldst(5)cfid(-1)csize(0)in(1)fDest(0)-cid2(2)st2(6)oldst2(4)
ccCallDisconnect (callID=0x1, cause=0x10 tag=0x0)
ccCallDisconnect (callID=0x2, cause=0x10 tag=0x0)
cc_api_call_disconnect_done(vdbPtr=0x60B6C5D4, callID=0x2, disp=0, tag=0x0)
sess_appl: ev(10), cid(2), disp(0)
ssa: cid(2)st(7)oldst(4)cfid(-1)csize(0)in(0)fDest(0)-cid2(1)st2(7)oldst2(6)
cc_api_call_disconnect_done(vdbPtr=0x60BFB530, callID=0x1, disp=0, tag=0x0)
sess_appl: ev(10), cid(1), disp(0)
ssa: cid(1)st(7)oldst(6)cfid(-1)csize(1)in(1)fDest(0)
```

debug voip ivr

To display debug messages for Voice over IP (VOIP) IVR interactions, use the **debug voip ivr** command. To disable the debug output, use the **no** form of this command.

debug voip ivr

[**no**] **debug voip ivr** *type*

Syntax Description

all	Displays all IVR messages.
applib	Displays IVR API libraries being processed.
callsetup	Displays IVR call setup being processed.
digitcollect	Displays IVR digits collected during the call.
dynamic	Displays IVR dynamic prompt play debug.
error	Displays IVR errors.
script	Displays IVR script debug.
settlement	Displays IVR settlement activities.
states	Displays IVR states.
tclcommands	Displays the TCL commands used in the script.

Defaults

Debug is not enabled.

Command History

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

Examples

The following examples are from the code for Cisco IOS Release 12.1(3)T. The output is displayed when the **debug voip ivr** *type* command is entered.

The following output is displayed when the **debug voip ivr applib** command is entered:

```
Router# debug voip ivr applib

ivr:
 ivr app library debugging is on
Router#
Jan 10 17:42:04.180:AppManagerCCAPI_Interface:
Jan 10 17:42:04.180:AppNewLeg
Jan 10 17:42:04.180:AppPushLegORConnection:Pushing LEG[34   ][NULL
] Onto {HAN[TCL_HAND][NULL  ]   ( )}
Jan 10 17:42:04.180:Event CC_EV_CALL_SETUP_IND[29]:LEG[34
][TCL_HAND]
Jan 10 17:42:04.184:AppPushHandler:Pushing {HAN[DC_HAND ][NULL  ]
( )} Onto {HAN[TCL_HAND][NULL  ]   ( LEG[34   ][TCL_HAND] )}
Jan 10 17:42:04.184:AppPushLegORConnection:Pushing LEG[34
][TCL_HAND] Onto {HAN[DC_HAND ][TCL_HAND]   ( )}
Jan 10 17:42:04.184:$ mediaPlay():CallID 34
Jan 10 17:42:04.184:Event CC_EV_CALL_REPORT_DIGITS_DONE[45]:LEG[34
][DC_HAND ]
```

```
Jan 10 17:42:17.261:AppMediaCallback:CallID 34 received
           response 'MSW_RESPONSE_TYPE_PLAY'
           with reason 'MSW_REASON_GENERIC_SUCCESS'
Jan 10 17:42:17.261:Event APP_EV_MEDIA_CALLBACK[47]:LEG[34
][DC_HAND ]
Jan 10 17:42:18.209:%ISDN-6-DISCONNECT:Interface Serial0:0
disconnected from unknown , call lasted 13 seconds
```

The following output is displayed when the **debug voip ivr callsetup** command is entered:

```
Router# debug voip ivr callsetup

Jan 10 17:45:57.528:%SYS-5-CONFIG_I:Configured from console by lab on
console
Jan 10 17:46:37.682:InitiateCallSetup:Incoming[66] AlertTime -1
Destinations(1) [ 3450070 ]
Jan 10 17:46:37.682:DNInitiate:Destination[3450070]
Jan 10 17:46:37.682:DNSetupPeer:
Jan 10 17:46:37.682:Destination SetupPeer cid(66), destPat(3450070),
match(2), prefix(), peer(61CB5CAC)
Jan 10 17:46:37.762:DNHandler:
(DN_SETTING[1])--(CC_EV_CALL_ALERT[11])--IGNORED-->>(DN_SETTING[1])
Jan 10 17:46:37.762:CS_Setting_ALERT:
Jan 10 17:46:37.762:CSPopLegAndWait:
Jan 10 17:46:37.762:CallSetupHandler:
 (CS_SETTING[0]) -----(CS_EV_ALERT[0])------->>>(CS_CONFINGALERT[4])
Jan 10 17:46:37.762:CS_ConfingAlert_CREATEDONE:
Jan 10 17:46:37.762:CallSetupHandler:
 (CS_CONFINGALERT[4])
-----(CS_EV_CREATEDONE[4])------->>>(CS_CONFEDALERT[5])
Jan 10 17:46:37.762:CallSetupHandler:
 (CS_CONFEDALERT[5])--(DN_SETTING[APP_EV_NULL])--IGNORED-->>>(CS_CONFEDALERT[5])

Router#
Jan 10 17:46:47.682:CallSetupHandler:
 (CS_CONFEDALERT[5])--(DN_SETTING[APP_EV_NULL])--IGNORED-->>>(CS_CONFEDALERT[5])

Jan 10 17:46:48.642:CS_ConfedAlert_CONNECTED:
Jan 10 17:46:48.642:CSDiscReturnAndEmptyLegALL:
Jan 10 17:46:48.642:DNCleanup:
Jan 10 17:46:48.642:DNSettlementCleanup:cid(66) trans=0, provider=0
Jan 10 17:46:48.642:CSReturnIFDone:CallSetup Returning(Status
CS_ACTIVE)
Jan 10 17:46:48.642:CallSetupHandler:
 (CS_CONFEDALERT[5]) -----(CS_EV_CONNECTED[1])------->>>(CS_CONFED[3])
Jan 10 17:46:48.646:CallSetupCleanup:
Router #
```

The following output is displayed when the **debug voip ivr digitcollect** command is entered:

```
Router# debug voip ivr digitcollect

ivr:
 ivr digit collect debugging is on
Router#
Router#
Router#
Jan 10 17:47:55.558:DigitCollect:DialPlan=FALSE AbortKey=* TermKey=#
NumPatts=1
       Enable=FALSE InterruptPrompt=TRUE maxDigits=11
Jan 10 17:47:55.558:act_DCRunning_RDone:callid=68 Enable succeeded.
Router#
Jan 10 17:48:04.006:DCHandlerFunc:PassingThrough
Jan 10 17:48:04.066:act_DCRunning_Digit::pLeg 68 Digit 1
```

```
Jan 10 17:48:04.066:act_DCRunning_RDone:callid=68 Reporting disabled.
Jan 10 17:48:04.066:DigitCollectComplete:Status 5=DC_MATCHED_PATTERN.
Digits=1
Jan 10 17:48:04.070:DigitCollect:DialPlan=FALSE AbortKey=* TermKey=#
NumPatts=0
       Enable=FALSE InterruptPrompt=TRUE maxDigits=11
Jan 10 17:48:04.070:DCHandlerCleanup:
Jan 10 17:48:04.074:act_DCRunning_RDone:callid=68 Enable succeeded.
Router#
Router#
Jan 10 17:48:08.038:DCHandlerFunc:PassingThrough
Jan 10 17:48:09.246:DCHandlerFunc:PassingThrough
Jan 10 17:48:09.286:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:09.478:DCHandlerFunc:PassingThrough
Jan 10 17:48:09.506:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:10.739:DCHandlerFunc:PassingThrough
Jan 10 17:48:10.779:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:11.027:DCHandlerFunc:PassingThrough
Jan 10 17:48:11.067:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:11.687:DCHandlerFunc:PassingThrough
Jan 10 17:48:11.747:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:12.219:DCHandlerFunc:PassingThrough
Jan 10 17:48:12.279:act_DCRunning_Digit::pLeg 68 Digit 2
Jan 10 17:48:14.227:DCHandlerFunc:PassingThrough
Jan 10 17:48:14.287:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:14.779:DCHandlerFunc:PassingThrough
Jan 10 17:48:14.859:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:15.307:DCHandlerFunc:PassingThrough
Jan 10 17:48:15.359:act_DCRunning_Digit::pLeg 68 Digit 1
Jan 10 17:48:15.719:DCHandlerFunc:PassingThrough
Jan 10 17:48:15.759:act_DCRunning_Digit::pLeg 68 Digit 2
Jan 10 17:48:16.219:DCHandlerFunc:PassingThrough
Jan 10 17:48:16.299:act_DCRunning_Digit::pLeg 68 Digit T
Jan 10 17:48:16.299:act_DCRunning_RDone:callid=68 Reporting disabled.
Jan 10 17:48:16.299:DigitCollectComplete:Status 5=DC_MATCHED_PATTERN.
Digits=1111121112
Jan 10 17:48:16.303:DCHandlerCleanup:
Jan 10 17:48:16.335:DigitCollect:DialPlan=TRUE AbortKey=* TermKey=#
NumPatts=0
       Enable=FALSE InterruptPrompt=TRUE maxDigits=0
Jan 10 17:48:16.339:act_DCRunning_RDone:callid=68 Enable succeeded.
Router #
```

The following output is displayed when the **debug voip ivr script** command is entered:

```
Router# deb voip ivr script

ivr:
 ivr script debugging is on
Router#
Jan 10 17:49:10.250:FSM Transtion:([1
]CALL_INIT,[29]ev_setup_indication)---([10]act_Setup)--->([4
]LANGSELECTION)
Jan 10 17:49:10.250:TotalLanguages= 2
Router#
Router#
Jan 10 17:49:16.662:FSM Transtion:([4
]LANGSELECTION,[55]ev_digitcollect_done)---([1 ]act_LangSelect)--->([5
]CARDSELECTION)
Router#
Router#
Jan 10 17:49:20.630:([5 ]CARDSELECT,[47]ev_media_d) ------> NOTHANDLED
Jan 10 17:49:26.770:FSM Transtion:([5
]CARDSELECTION,[55]ev_digitcollect_done)---([2
```

```
]act_GotCardNumber)--->([6 ]AUTHORIZE)
Jan 10 17:49:26.806:FSM Transtion:([6
]AUTHORIZE,[49]ev_authorize_done)---([8 ]act_FirstAuthorized)--->([7
]GETDEST)
Jan 10 17:49:26.806: aaa authorize Status=ao_000
Router#
Router#
Router#
Jan 10 17:49:33.395:([7 ]GETDEST  ,[47]ev_media_d) ------> NOTHANDLED
Jan 10 17:49:36.411:FSM Transtion:([7
]GETDEST,[55]ev_digitcollect_done)---([3 ]act_GotDest)--->([8
]SECONDAUTHORIZE)
Jan 10 17:49:36.451:FSM Transtion:([8
]SECONDAUTHORIZE,[49]ev_authorize_done)---([5
]act_SecondAuthorized)--->([10]PLACECALL)
Jan 10 17:49:36.451: aaa authorize Status=ao_000
Jan 10 17:49:42.179:FSM Transtion:
([10]PLACECALL,[47]ev_media_done)---([9
]act_CallSetup)--->([10]PLACECALL)
```

The following output is displayed when the **debug voip ivr tclcommands** command is entered:

```
Router# debug voip ivr tclcommands

ivr tcl commands debugging is on
Router#
Jan 10 17:50:29.106:tcl_infotagCmd:infotag get leg_ani
Jan 10 17:50:29.106:tcl_getInfoCmd:get leg_ani
Jan 10 17:50:29.106:vtr_ci_incani:argc 2 argindex 2
Jan 10 17:50:29.106:tcl_infotagCmd:infotag set med_language 1
Jan 10 17:50:29.106:tcl_setInfoCmd:set med_language 1
Jan 10 17:50:29.106:vtw_ms_language:
Jan 10 17:50:29.106:tcl_legCmd:leg setupack leg_incoming
Jan 10 17:50:29.106:tcl_setupAckCmd:setupack leg_incoming
Jan 10 17:50:29.106:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:29.106:tcl_legCmd:leg proceeding leg_incoming
Jan 10 17:50:29.106:tcl_callProceedingCmd:proceeding leg_incoming
Jan 10 17:50:29.106:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:29.110:tcl_legCmd:leg connect leg_incoming
Jan 10 17:50:29.110:tcl_callConnectCmd:connect leg_incoming
Jan 10 17:50:29.110:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:29.110:tcl_legCmd:leg collectdigits leg_incoming param1
patterns
Jan 10 17:50:29.110:tcl_collectDigitsCmd:collectdigits leg_incoming
param1 patterns
Jan 10 17:50:29.110:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:29.110:tcl_mediaCmd:media play leg_incoming _welcome.au
%s1000 %c1 _lang_sel1.au %s1000 %c2 _lang_sel2.au
Jan 10 17:50:29.110:tcl_mediaPlayCmd:play leg_incoming _welcome.au
%s1000 %c1 _lang_sel1.au %s1000 %c2 _lang_sel2.au
Jan 10 17:50:29.110:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Router#
Router#
Jan 10 17:50:35.506:tcl_infotagCmd:infotag get evt_status
Jan 10 17:50:35.506:tcl_getInfoCmd:get evt_status
Jan 10 17:50:35.506:vtr_ev_status:
Jan 10 17:50:35.510:tcl_infotagCmd:infotag get evt_dcdigits
Jan 10 17:50:35.510:tcl_getInfoCmd:get evt_dcdigits
```

```
Jan 10 17:50:35.510:vtr_ev_dcdigits:
Jan 10 17:50:35.510:DCDIGITS [1]
Jan 10 17:50:35.510:tcl_infotagCmd:infotag set med_language 1
Jan 10 17:50:35.510:tcl_setInfoCmd:set med_language 1
Jan 10 17:50:35.510:vtw_ms_language:
Jan 10 17:50:35.510:tcl_legCmd:leg collectdigits leg_incoming param1
Jan 10 17:50:35.510:tcl_collectDigitsCmd:collectdigits leg_incoming

param1
Jan 10 17:50:35.510:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:35.510:tcl_mediaCmd:media play leg_incoming
_enter_card_num.au
Jan 10 17:50:35.510:tcl_mediaPlayCmd:play leg_incoming
_enter_card_num.au
Jan 10 17:50:35.514:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Router#
Jan 10 17:50:43.878:tcl_infotagCmd:infotag get evt_status
Jan 10 17:50:43.878:tcl_getInfoCmd:get evt_status
Jan 10 17:50:43.878:vtr_ev_status:
Jan 10 17:50:43.882:tcl_infotagCmd:infotag get evt_dcdigits
Jan 10 17:50:43.882:tcl_getInfoCmd:get evt_dcdigits
Jan 10 17:50:43.882:vtr_ev_dcdigits:
Jan 10 17:50:43.882:DCDIGITS [1111121112]
Jan 10 17:50:43.882:tcl_aaaCmd:aaa authorize 111112 1112 50073
leg_incoming
Jan 10 17:50:43.882:tcl_AuthorizeCmd:authorize 111112 1112 50073
leg_incoming
Jan 10 17:50:43.882:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:43.882:Authorize
Jan 10 17:50:43.882:  account=111112
Jan 10 17:50:43.882:  password=1112
Jan 10 17:50:43.882:  ani   =50073
Jan 10 17:50:43.882:  dnis  =
Jan 10 17:50:43.910:tcl_infotagCmd:infotag get evt_status
Jan 10 17:50:43.910:tcl_getInfoCmd:get evt_status
Jan 10 17:50:43.910:vtr_ev_status:
Jan 10 17:50:43.914:tcl_infotagCmd:infotag get aaa_avpair_exists
creditAmount
Jan 10 17:50:43.914:tcl_getInfoCmd:get aaa_avpair_exists creditAmount
Jan 10 17:50:43.914:vtr_ra_avpair_exists:
Jan 10 17:50:43.914:tcl_infotagCmd:infotag get aaa_avpair creditAmount

Jan 10 17:50:43.914:tcl_getInfoCmd:get aaa_avpair creditAmount
Jan 10 17:50:43.914:vtr_ra_avpair:
Jan 10 17:50:43.914:tcl_legCmd:leg collectdigits leg_incoming param2
Jan 10 17:50:43.914:tcl_collectDigitsCmd:collectdigits leg_incoming
param2
Jan 10 17:50:43.914:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
Jan 10 17:50:43.914:tcl_mediaCmd:media play leg_incoming _you_have.au
%a1000 %s1000 _enter_dest.au
Jan 10 17:50:43.914:tcl_mediaPlayCmd:play leg_incoming _you_have.au
%a1000 %s1000 _enter_dest.au
Jan 10 17:50:43.918:vtd_lg_incoming:Legs [71 ]VARTAG Translation Leg
Count=1
```

Related Commands

Command	Description
debug voip ivr call setup	Displays the call setup information.
debug voip ivr digit collect	Displays the digits collected during the call.
debug voip ivr script	Displays the scripts being processed.
debug voip ivr tclcommands	Displays the TCL commands being called.

debug voip ivr settlement

The **debug voip ivr** command is used to debug the IVR application. IVR debug messages appear when a call is being actively handled by the IVR scripts. Error outputs only occurs if something is not working or an error condition has been raised. The output when the keyword **states** is used, supplies information about the current status of the IVR script and the different events, that occur in that state. This document, for Cisco IOS Release 12.0(4)XH shows the **debug voip ivr settlement** command using the output for the **settlement** keyword only. Use the **no** form of this command to disable this command.

debug voip ivr [**states** | **error** | **settlement** | **dynamic**| **all**]

no debug voip ivr [**states** | **error** | **settlement** | **dynamic** | **all**]

Syntax Description

all	(Optional) Displays both **states** and **error** messages.
dynamic	(Optional) IVR dynamic prompt play debug.
error	(Optional) Displays information only if an error occurs.
settlement	(Optional) IVR settlement activities.
states	(Optional) Displays extensive information about how IVR is handling each call.

Defaults

Not enabled

Usage Guidelines

IVR debug messages appear when a call is handled by the IVR scripts. **Error** output should only occur if something is not working or an error condition is indicated. **States** output supplies information about the current status of the IVR script and the different events that occur in that state.

Settlement output logs activities related to settlement when a call is processed.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.0(4)XH	Settlement was added.

Examples

Example On the Originating Gateway

```
Router # debug voip ivr settlement
ivr settlement activities debugging is on
Router#
00:00:52:settlement_validate_token:cid(1), target=, tokenp=0x0
00:00:54:pcSettlementAuthorize:cid(1) authorizing using calling=408,
called=15125551212
00:00:54:pcSettlementAuthorize:cid(1) sending authorize request type=1
00:00:57:pcSettlementSetup:cid(1) settlement_curr_dest=0, num_dest=3
00:00:57:pcSettlementGetDestination:trans=0 gets error=0,
```

```
credit_time=14400
00:00:57:pcSettlementSetup:cid(1) placing call through
ip(1.14.115.85), calling(408),called(15125551212), digits(15125551212)
00:00:57:pcSettlementSetup:set settlement acct for cid(2) on
ip=1.14.115.85
Router#
```

Example On the Terminating Gateway

```
Router # debug voip ivr settlement
ivr settlement activities debugging is on
as5300-05#
00:10:02:settlement_validate_token:cid(1), target=settlement,
tokenp=0x618386B
4
00:10:02:settlement_validate_token:cid(1) return 1, credit_time=14400
00:10:02:Set settlement acct on cid(1) for trans=0, prov=0
as5300-05#
```

debug voip rawmsg

To display the raw message owner, length, and pointer, use the **debug voip rawmsg** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug voip rawmsg [*detail*]

no debug voip rawmsg [*detail*]

Syntax Description

detail	(Optional) Prints the contents of the raw message in hexadecimal.

Defaults

Disabled.

Command History

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

Examples

The following example shows output when you use the **debug voip rawmsg** command:

```
as5300# debug voip rawmsg

00:57:40:Raw Message owner is 2, length is 69, ptr is 60FE4F5C, type is 0, protocol id is
0
00:57:40:Raw Message owner is 5, length is 69, ptr is 60FE4F5C, type is 0, protocol id is
0
0
```

The following example shows output when you use the **debug voip rawmsg detail** command:

```
as5300# debug voip rawmsg detail

00:57:40:Raw Message owner is 2, length is 69, ptr is 60FE4F5C, type is 0, protocol id is
0
00:57:40:Raw Message is :04 03 80 90 A2 18 03 A9 83 97 1C 27 9F AA 06 80 01 00 82 01 00 92
01 11 8B 01 00 A1 16 02 02 01 00 06 04 2B 0C 09 00 80 0A 4D 4F 4E 49 43 41 20 33 32 33 1E
02 81 83 6C 05 09 80 33 32 33 70 04 89 38 30 30 A1
00:57:40:Raw Message owner is 5, length is 69, ptr is 60FE4F5C, type is 0, protocol id is
0
00:57:40:Raw Message is :04 03 80 90 A2 18 03 A9 83 97 1C 27 9F AA 06 80 01 00 82 01 00 92
01 11 8B 01 00 A1 16 02 02 01 00 06 04 2B 0C 09 00 80 0A 4D 4F 4E 49 43 41 20 33 32 33 1E
02 81 83 6C 05 09 80 33 32 33 70 04 89 38 30 30 A1
```

Related Commands

Command	Description
debug cdapi	Displays information about the call distributor application programming interface
debug tsp	Displays information about the telephony service provider.

debug voip settlement all

To enable debugging in all settlement areas, enter the **debug voip settlement all** EXEC command. Use the **no** form of this command to disable debugging output.

[**no**] **debug voip settlement all**

Syntax Description

enter	Displays all entrances.
error	Displays information only if an error occurs.
exit	Displays all exits.
misc	Displays the details on the code flow of each transaction.
network	Displays network connectivity data.
security	Displays security and encryption errors.
transaction	Displays transaction information.

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Usage Guidelines

The **debug voip settlement all** EXEC command enables the following debug settlement commands:

- **debug voip settlement enter**
- **debug voip settlement error**
- **debug voip settlement exit**
- **debug voip settlement security**
- **debug voip settlement misc**
- **debug voip settlement security**
- **debug voip settlement transaction**

debug voip settlement enter

To show all the settlement function entrances, enter the **debug voip settlement enter** command. Use the **no** form of this command to disable debugging output.

[**no**] **debug voip settlement enter**

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

```
00:43:40:OSP:ENTER:OSPPMimeMessageCreate()
00:43:40:OSP:ENTER:OSPPMimeMessageInit()
00:43:40:OSP:ENTER:OSPPMimeMessageSetContentAndLength()
00:43:40:OSP:ENTER:OSPPMimeMessageBuild()
00:43:40:OSP:ENTER:OSPPMimeDataFree()
00:43:40:OSP:ENTER:OSPPMimePartFree()
00:43:40:OSP:ENTER:OSPPMimePartFree()
00:43:40:OSP:ENTER:OSPPMsgInfoAssignRequestMsg()
00:43:40:OSP:ENTER:osppHttpSelectConnection
00:43:40:OSP:ENTER:OSPPSockCheckServicePoint() ospvConnected = <1>
00:43:40:OSP:ENTER:OSPPSockWaitTillReady()
00:43:40:OSP:ENTER:osppHttpBuildMsg()
00:43:40:OSP:ENTER:OSPPSSLSessionWrite()
00:43:40:OSP:ENTER:OSPPSockWrite()
00:43:40:OSP:ENTER:OSPPSockWaitTillReady()
```

debug voip settlement error

To show all the settlement errors, enter the **debug voip settlement error** command. Use the **no** form of this command to disable debugging output.

[**no**] **debug voip settlement error**

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

```
00:45:50:OSP:OSPPSockProcessRequest:http recv init header failed
00:45:50:OSP:osppHttpSetupAndMonitor:attempt#0 on http=0x6141A514, limit=1 error=14310
```

Usage Guidelines

See "Error Code Definitions" section on page 1019.

Error Code Definitions

```
-1:OSP internal software error.
16:A bad service was chosen.
17:An invalid parameter was passed to OSP.
9010:Attempted to access an invalid pointer.
9020:A time related error occurred.

10010:OSP provider module failed initialization.
10020:OSP provider tried to access a NULL pointer.
10030:OSP provider could not fine transaction collection.
10040:OSP provider failed to obtain provider space.
10050:OSP provider tried to access an invalid handle.
10060:OSP provider has reached the maximum number of providers.

11010:OSP transaction tried to delete a transaction which was not allowed.
11020:OSP transaction tried a transaction which does not exist.
11030:OSP transaction tried to start a transaction, but data had already been delivered.
11040:OSP transaction could not identify the response given.
11050:OSP transaction failed to obtain transaction space.
11060:OSP transaction failed (possibly ran out) to allocate memory.
11070:OSP transaction tried to perform a transaction which is not allowed.
11080:OSP transaction found no more responses.
11090:OSP transaction could not find a specified value.
11100:OSP transaction did not have enough space to copy.
11110:OSP transaction - call id did not match destination.
11120:OSP transaction encountered an invalid entry.
11130:OSP transaction tried to use a token too soon.
11140:OSP transaction tried to use a token too late.
11150:OSP transaction - source is invalid.
11160:OSP transaction - destination is invalid.
11170:OSP transaction - calling number is invalid.
11180:OSP transaction - called number is invalid.
11190:OSP transaction - call id is invalid.
11200:OSP transaction - authentication id is invalid.
```

```
11210:OSP transaction - call id was not found
11220:OSP transaction - The IDS of the called number was invalid.
11230:OSP transaction - function not implemented.
11240:OSP transaction tried to access an invalid handle.
11250:OSP transaction returned an invalid return code.
11260:OSP transaction reported an invalid status code.
11270:OSP transaction encountered an invalid token.
11280:OSP transaction reported a status which could not be identified.
11290:OSP transaction in now valid after it was not found.
11300:OSP transaction could not find the specified destination.
11310:OSP transaction is valid until not found.
11320:OSP transaction - invalid signaling address.
11330:OSP transaction could not find the ID of the transmitter.
11340:OSP transaction could not find the source number.
11350:OSP transaction could not find the destination number.
11360:OSP transaction could not find the token.
11370:OSP transaction could not find the list.
11380:OSP transaction was not allowed to accumulate.
11390:OSP transaction - transaction usage was already reported.
11400:OSP transaction could not find statistics.
11410:OSP transaction failed to create new statistics.
11420:OSP transaction made an invalid calculation.
11430:OSP transaction was not allowed to get the destination.
11440:OSP transaction could not fine the authorization request.
11450:OSP transaction - invalid transmitter ID.
11460:OSP transaction could not find any data.
11470:OSP transaction found no new authorization requests.

12010:OSP security did not have enough space to copy.
12020:OSP security received and invalid argument.
12030:OSP security could not find the private key.
12040:OSP security encountered an un-implemented function.
12050:OSP security ran out of memory.
12060:OSP security received an invalid signal.
12065:OSP security could not initialize the SSL database.
12070:OSP security could not find space for the certificate.
12080:OSP security has no local certificate info defined.
12090:OSP security encountered a zero length certificate.

12100:OSP security encountered a certificate that is too big.
12110:OSP security encountered an invalid certificate.
12120:OSP security encountered a NULL certificate.
12130:OSP security has too many certificates.
12140:OSP security has no storage provided.
12150:OSP security has no private key.
12160:OSP security encountered an invalid context.
12170:OSP security was unable to allocate space.
12180:OSP security - CA certificates do not match.
12190:OSP security found no authority certificates

12200:OSP security - CA certificate index overflow.

13010:OSP error message - failed to allocate memory.

13110:OSP MIME error - buffer is too small.
13115:OSP MIME error - failed to allocate memory.
13120:OSP MIME error - could not find variable.
13125:OSP MIME error - no input was found.
13130:OSP MIME error - invalid argument.
13135:OSP MIME error - no more space.
13140:OSP MIME error - received an invalid type.
13145:OSP MIME error - received an invalid subtype.
13150:OSP MIME error - could not find the specified protocol.
13155:OSP MIME error - could not find MICALG.
```

```
13160:OSP MIME error - boundary was not found.
13165:OSP MIME error - content type was not found.
13170:OSP MIME error - message parts were not found.

13301:OSP XML error - received incomplete XML data.
13302:OSP XML error - bad encoding of XML data.
13303:OSP XML error - bad entity in XML data.
13304:OSP XML error - bad name in XML data.
13305:OSP XML error - bad tag in XML data.
13306:OSP XML error - bad attribute in XML data.
13307:OSP XML error - bad CID encoding in XML data.
13308:OSP XML error - bad element found in XML data.
13309:OSP XML error - no element found in XML data.
13310:OSP XML error - no attribute found in XML data.
13311:OSP XML error - OSP received invalid arguments.
13312:OSP XML error - failed to create a new buffer.
13313:OSP XML error - failed to get the size of a buffer.
13314:OSP XML error - failed to send the buffer.
13315:OSP XML error - failed to read a block from the buffer.
13316:OSP XML error - failed to allocate memory.
13317:OSP XML error - could not find the parent.
13318:OSP XML error - could not find the child.
13319:OSP XML error - data type not found in XML data.
13320:OSP XML error - failed to write a clock to the buffer.

13410:OSP data error - no call id preset.
13415:OSP data error - no token present.
13420:OSP data error - bad number presented.
13425:OSP data error - no destination found.
13430:OSP data error - no usage indicator present.
13435:OSP data error - no status present.
13440:OSP data error - no usage configured.
13445:OSP data error - no authentication indicator.
13450:OSP data error - no authentication request.
13455:OSP data error - no authentication response.
13460:OSP data error - no authentication configuration.
13465:OSP data error - no re-authentication request.
13470:OSP data error - no re-authentication response.
13475:OSP data error - invalid data type present.
13480:OSP data error - no usage information available.
13485:OSP data error - no token info present.
13490:OSP data error - invalid data present.

13500:OSP data error - no alternative info present.
13510:OSP data error - no statistics available.
13520:OSP data error - no delay present.
13610:OSP certificate error - memory allocation failed.

14010:OSP communications error - invalid communication size.
14020:OSP communications error - bad communication value.
14030:OSP communications error - parser error.
14040:OSP communications error - no more memory available.
14050:OSP communications error - communication channel currently in use.
14060:OSP communications error - invalid argument passed.
14070:OSP communications error - no service points present.
14080:OSP communications error - no service points available.
14085:OSP communications error - thread initialization failed.
14086:OSP communications error - communications is shutdown.

14110:OSP message queue error - no more memory available.
14120:OSP message queue error - failed to add a request.
14130:OSP message queue error - no event queue present.
14140:OSP message queue error - invalid arguments passed.
```

```
14210:OSP HTTP error - 100 - bad header.
14220:OSP HTTP error - 200 - bad header.
14221:OSP HTTP error - 400 - bad request.
14222:OSP HTTP error - bas service port present.
14223:OSP HTTP error - failed to add a request.
14230:OSP HTTP error - invalid queue present.
14240:OSP HTTP error - bad message received.
14250:OSP HTTP error - invalid argument passed.
14260:OSP HTTP error - memory allocation failed.
14270:OSP HTTP error - failed to create a new connection.
14280:OSP HTTP error - server error.
14290:OSP HTTP error - HTTP server is shutdown.
14292:OSP HTTP error - failed to create a new SSL connection.
14295:OSP HTTP error - failed to create a new SSL context.
14297:OSP HTTP error - service unavailable.

14300:OSP socket error - socket select failed.
14310:OSP socket error - socket receive failed.
14315:OSP socket error - socket send failed.
14320:OSP socket error - failed to allocate memory for the receive buffer.
14320:OSP socket error - socket reset.
14330:OSP socket error - failed to create the socket.
14340:OSP socket error - failed to close the socket.
14350:OSP socket error - failed to connect the socket.
14360:OSP socket error - failed to block I/O on the socket.
14370:OSP socket error - failed to disable nagle on the socket.

14400:OSP SSL error - failed to allocate memory.
14410:OSP SSL error - failed to initialize the context.
14420:OSP SSL error - failed to retrieve the version.
14430:OSP SSL error - failed to initialize the session.
14440:OSP SSL error - failed to attach the socket.
14450:OSP SSL error - handshake failed.
14460:OSP SSL error - failed to close SSL.
14470:OSP SSL error - failed to read from SSL.
14480:OSP SSL error - failed to write to SSL.
14490:OSP SSL error - could not get certificate.
14495:OSP SSL error - no root certificate found.
14496:OSP SSL error - failed to set the private key.
14497:OSP SSL error - failed to parse the private key.
14498:OSP SSL error - failed to add certificates.
14499:OSP SSL error - failed to add DN.

15410:OSP utility error - not enough space for copy.
15420:OSP utility error - no time stamp has been created.
15430:OSP utility error - value not found.
15440:OSP utility error - failed to allocate memory.
15450:OSP utility error - invalid argument passed.

15500:OSP buffer error - buffer is empty.
15510:OSP buffer error - buffer is incomplete.

15980:OSP POW error.
15990:OSP Operating system conditional variable timeout.

16010:OSP X509 error - serial number undefined.
16020:OSP X509 error - certificate undefined.
16030:OSP X509 error - invalid context.
16040:OSP X509 error - decoding error.
16050:OSP X509 error - unable to allocate space.
16060:OSP X509 error - invalid data present.
16070:OSP X509 error - certificate has expired.
16080:OSP X509 error - certificate not found.
```

```
17010:OSP PKCS1 error - tried to access invalid private key pointer
17020:OSP PKCS1 error - unable to allocate space.
17030:OSP PKCS1 error - invalid context found.
17040:OSP PKCS1 error - tried to access NULL pointer.
17050:OSP PKCS1 error - private key overflow.

18010:OSP PKCS7 error - signer missing.
18020:OSP PKCS7 error - invalid signature found.
18020:OSP PKCS7 error - unable to allocate space.
18030:OSP PKCS7 error - encoding error.
18040:OSP PKCS7 error - tried to access invalid pointer.
18050:OSP PKCS7 error - buffer overflow.

19010:OSP ASN1 error - tried to access NULL pointer.
19020:OSP ASN1 error - invalid element tag found.
19030:OSP ASN1 error - unexpected high tag found.
19040:OSP ASN1 error - invalid primitive tag found.
19050:OSP ASN1 error - unable to allocate space.
19060:OSP ASN1 error - invalid context found.
19070:OSP ASN1 error - invalid time found.
19080:OSP ASN1 error - parser error occurred.
19090:OSP ASN1 error - parsing complete.
19100:OSP ASN1 error - parsing defaulted.
19110:OSP ASN1 error - length overflow.
19120:OSP ASN1 error - unsupported tag found.
19130:OSP ASN1 error - object ID not found.
19140:OSP ASN1 error - object ID mismatch.
19150:OSP ASN1 error - unexpected int base.
19160:OSP ASN1 error - buffer overflow.
19170:OSP ASN1 error - invalid data reference ID found.
19180:OSP ASN1 error - no content value for element found.
19190:OSP ASN1 error - integer overflow.

20010:OSP Crypto error - invalid parameters found.
20020:OSP Crypto error - unable to allocate space.
20030:OSP Crypto error - could not verify signature.
20040:OSP Crypto error - implementation specific error.
20050:OSP Crypto error - tried to access invalid pointer.
20060:OSP Crypto error - not enough space to perform operation.

21010:OSP PKCS8 error - invalid private key pointer found.
21020:OSP PKCS8 error - unable to allocate space for operation.
21030:OSP PKCS8 error - invalid context found.
21040:OSP PKCS8 error - tried to access NULL pointer.
21050:OSP PKCS8 error - private key overflow.

22010:OSP Base 64 error - encode failed.
22020:OSP Base 64 error - decode failed.

22510:OSP audit error - failed to allocate memory.

156010:OSP RSN failure error - no data present.
156020:OSP RSN failure error - data is invalid.
```

debug voip settlement exit

To show all the settlement function exits, enter the **debug voip settlement exit** command. Use the **no** form of this command to disable debugging output.

debug voip settlement exit

no debug voip settlement exit

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

```
01:21:10:OSP:EXIT :OSPPMimeMessageInit()
01:21:10:OSP:EXIT :OSPPMimeMessageSetContentAndLength()
01:21:10:OSP:EXIT :OSPPMimeMessageBuild()
01:21:10:OSP:EXIT :OSPPMimePartFree()
01:21:10:OSP:EXIT :OSPPMimePartFree()
01:21:10:OSP:EXIT :OSPPMimeDataFree()
01:21:10:OSP:EXIT :OSPPMimeMessageCreate()
01:21:10:OSP:EXIT :OSPPMsgInfoAssignRequestMsg()
01:21:10:OSP:EXIT :osppHttpSelectConnection
01:21:10:OSP:EXIT :OSPPSockCheckServicePoint() isconnected(1)
01:21:10:OSP:EXIT :osppHttpBuildMsg()
01:21:10:OSP:EXIT :OSPPSockWrite() (0)
01:21:10:OSP:EXIT :OSPPSSLSessionWrite() (0)
01:21:10:OSP:EXIT :OSPPSSLSessionRead() (0)
01:21:10:OSP:EXIT :OSPPSSLSessionRead() (0)
01:21:10:OSP:EXIT :OSPPHttpParseHeader
01:21:10:OSP:EXIT :OSPPHttpParseHeader
01:21:10:OSP:EXIT :OSPPSSLSessionRead() (0)
01:21:10:OSP:EXIT :OSPPUtilMemCaseCmp()
```

debug voip settlement misc

To show the details on the code flow of each settlement transaction, enter the **debug voip settlement misc** command. Use the **no** form of this command to disable debugging output.

debug voip settlement misc

no debug voip settlement misc

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

```
00:52:03:OSP:osp_authorize:callp=0x6142770C
00:52:03:OSP:OSPPTransactionRequestNew:ospvTrans=0x614278A8
00:52:03:OSP:osppCommMonitor:major:minor=(0x2:0x1)
00:52:03:OSP:HTTP connection:reused
00:52:03:OSP:osppHttpSetupAndMonitor:HTTP=0x6141A514, QUEUE_EVENT from eventQ=0x6141A87C,
comm=0x613F16C4, msginfo=0x6142792C
00:52:03:OSP:osppHttpSetupAndMonitor:connected = <TRUE>
00:52:03:OSP:osppHttpSetupAndMonitor:HTTP=0x6141A514, build msginfo=0x6142792C, trans=0x2
00:52:04:OSP:osppHttpSetupAndMonitor:HTTP=0x6141A514, msg built and sent:error=0,
msginfo=0x6142792C
00:52:04:OSP:osppHttpSetupAndMonitor:monitor exit. errorcode=0
00:52:04:OSP:osppHttpSetupAndMonitor:msginfo=0x6142792C, error=0, shutdown=0
00:52:04:OSP:OSPPMsgInfoProcessResponse:msginfo=0x6142792C, err=0, trans=0x614278A8,
handle=2
00:52:04:OSP:OSPPMsgInfoChangeState:transp=0x614278A8, msgtype=12 current state=2
00:52:04:OSP:OSPPMsgInfoChangeState:transp=0x614278A8, new state=4
00:52:04:OSP:OSPPMsgInfoProcessResponse:msginfo=0x6142792C, context=0x6142770C, error=0
00:52:04:OSP:osp_get_destination:trans_handle=2, get_first=1, callinfop=0x614275E0
00:52:04:OSP:osp_get_destination:callinfop=0x614275E0 get dest=1.14.115.51,
validafter=1999-01-20T02:04:32Z, validuntil=1999-01-20T02:14:32Z
00:52:04:OSP:osp_parse_destination:dest=1.14.115.51
00:52:04:OSP:osp_get_destination:callinfop=0x614275E0, error=0, ip_addr=1.14.115.51,
credit=60
00:52:06:OSP:stop_settlement_ccapi_accounting:send report for callid=0x11, transhandle=2
00:52:06:OSP:osp_report_usage:transaction=2, duration=0, lostpkts=0, lostfrs=0,
lostpktr=0, lostfrr=0
```

debug voip settlement network

To show all the messages exchanged between a router and a settlement provider, enter the **debug voip settlement network** command. Use the **no** form of this command to disable debugging output.

debug voip settlement network

no debug voip settlement network

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Usage Guidelines

Using the **debug voip settlement network** command shows messages, in detail, in HTTP and XML formats.

Examples

```
00:47:25:OSP:HTTP connection:reused
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=0, timeout=0, select=1
00:47:25:OSP:osppHttpBuildAndSend():http=0x6141A514 sending:
POST /scripts/simulator.dll?handler HTTP/1.1
Host:1.14.115.12
content-type:text/plain
Content-Length:439
Connection:Keep-Alive

Content-Type:text/plain
Content-Length:370

<?xml version="1.0"?><Message messageId="1" random="8896">
<AuthorisationRequest componentId="1">
<Timestamp>
1993-03-01T00:47:25Z</Timestamp>
<CallId>
<![CDATA[12]]></CallId>
<SourceInfo type="e164">
5551111</SourceInfo>
<DestinationInfo type="e164">
5552222</DestinationInfo>
<Service/>
<MaximumDestinations>
3</MaximumDestinations>
</AuthorisationRequest>
</Message>

00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=0, timeout=1, select=1
00:47:25:OSP:OSPM_SEND:bytes_sent = 577
00:47:25:OSP:OSPPSockProcessRequest:SOCKFD=0, Expecting 100, got
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=1, timeout=1, select=1
```

```
00:47:25:OSP:OSPPSSLSessionRead() recving 1 bytes:
HTTP/1.1 100 Continue
Server:Microsoft-IIS/4.0
Date:Wed, 20 Jan 1999 02:01:54 GMT
00:47:25:OSP:OSPPSockProcessRequest:SOCKFD=0, Expecting 200, got
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=1, timeout=1, select=1
00:47:25:OSP:OSPPSSLSessionRead() recving 1 bytes:
HTTP/1.1 200 OK
Server:Microsoft-IIS/4.0
Date:Wed, 20 Jan 1999 02:01:54 GMT
Connection:Keep-Alive
Content-Type:multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1;
boundary=bar
Content-Length:1689

00:47:25:OSP:OSPPSockProcessRequest:SOCKFD=0, error=0, HTTP response

00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=1, timeout=1, select=1
00:47:25:OSP:OSPPSSLSessionRead() recving 1689 bytes:

--bar
Content-Type:text/plain
Content-Length:1510

<?xml version="1.0"?><Message messageId="1" random="27285">
<AuthorisationResponse componentId="1">
<Timestamp>
1999-01-20T02:01:54Z</Timestamp>
<Status>
<Description>
success</Description>
<Code>
200</Code>
</Status>
<TransactionId>
101</TransactionId>
<Destination>
<AuthorityURL>
http://www.myauthority.com</AuthorityURL>
<CallId>
<![CDATA[12]]></CallId>
<DestinationInfo type="e164">
5552222</DestinationInfo>
<DestinationSignalAddress>
1.14.115.51</DestinationSignalAddress>
<Token encoding="base64">
PD94bWwgdmVyc2lvbj0xLjA/PjxNZXNzYWdlIG1lc3NhZ2VJZD0iMSIgcmFuZG9tPSIxODM0OSI+PFRva2VuSW5mbz
48U291cmNlSW5mbyB0eXBlPSJlMTY0Ij41NTUxMTExPC9Tb3VyY2VJbmZvPjxEZXN0aW5hdGlvbkluZm8gdHlwZT0i
ZTE2NCI+NTU1MjIyMjwvRGVzdGluYXRpb25JbmZvPjxDYWxsSWQ+PCFbQ0RBVEFbMV1dPjwvQ2FsbElkPjxWYWxpZE
FmdGVyPjE5OTgtMTItMDhUMjA6MDQ6MFo8L1ZhbGlkQWZ0ZXI+PFZhbGlkVW50aWw+MTk5OS0xMi0zMVQyMzo1OTo1
OVo8L1ZhbGlkVW50aWw+PFRyYW5zYWN0aW9uSWQ+MTAxPC9UcmFuc2FjdGlvbklkPjxVc2FnZURldGFpbD48QW1vdW
50PjE0NDAwPC9BbW91bnQ+PEluY3JlbWVudD4xPC9JbmNyZW1lbnQ+PFNlcnZpY2UvPjxVbml0PnM8L1VuaXQ+PC9V
c2FnZURldGFpbD48L1Rva2VuSW5mbz48L01lc3NhZ2U+</Token>
<UsageDetail>
<Amount>
60</Amount>
<Increment>
1</Increment>
<Service/>
<Unit>
s</Unit>
</UsageDetail>
```

```
<ValidAfter>
1999-01-20T01:59:54Z</ValidAfter>
<ValidUntil>
1999-01-20T02:09:54Z</ValidUntil>
</Destination>
<transnexus.com:DelayLimit critical="False">
1000</transnexus.com:DelayLimit>
<transnexus.com:DelayPreference critical="False">
1</transnexus.com:DelayPreference>
</AuthorisationResponse>
</Message>

--bar
Content-Type:application/pkcs7-signature
Content-Length:31

This is your response signature

--bar--
```

debug voip settlement security

To show all the tracing related to security, such as SSL or S/MIME, enter the **debug voip settlement security** command. Use the **no** form of this command to disable debugging output.

debug voip settlement security

no debug voip settlement security

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

Not available due to security issues.

debug voip settlement transaction

To see all the attributes of the transactions on the settlement gateway, use the **debug voip settlement transaction** EXEC command. Use the **no** form of this command to disable debugging output.

[**no**] **debug voip settlement transaction**

Defaults

Not enabled

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

Sample output from the originating gateway:

```
00:44:54:OSP:OSPPTransactionNew:trans=0, err=0
00:44:54:OSP:osp_authorize:authorizing trans=0, err=0
router>
00:45:05:OSP:stop_settlement_ccapi_accounting:send report for
callid=7, trans
=0, calling=5710868, called=15125551212, curr_Dest=1
00:45:05:OSP:OSPPTransactionDelete:deleting trans=0
```

Sample output from the terminating gateway:

```
00:44:40:OSP:OSPPTransactionNew:trans=0, err=0
00:44:40:OSP:osp_validate:validated trans=0, error=0, authorised=1
```