

debug vpm dsp

To show messages from the DSP on the VPM to the router, use the **debug vpm dsp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm dsp

no debug vpm dsp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vpm dsp** command shows messages from the DSP on the VPM to the router; this command can be useful if you suspect that the VPM is not functional. It is a simple way to check if the VPM is responding to off-hook indications and to evaluate timing for signaling messages from the interface.

Examples

The following example shows the DSP time stamp and the router time stamp for each event. For SIG_STATUS, the state value shows the state of the ABCD bits in the signaling message. This sample shows a call coming in on an FXO interface.

The router waits for ringing to terminate before accepting the call. State=0x0 indicates ringing; state 0x4 indicates not ringing.

```
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x0 timestamp=58172 systime=40024
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x4 timestamp=59472 systime=40154
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x4 timestamp=59589 systime=40166
```

The following output shows the digits collected:

```
vcsd_dsp_message: MSG_TX_DTMF_DIGIT: digit=4
vcsd_dsp_message: MSG_TX_DTMF_DIGIT: digit=1
vcsd_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vcsd_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vcsd_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
```

This shows the disconnect indication and the final call statistics reported by the DSP (which are then populated in the call history table):

```
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0xC timestamp=21214 systime=42882
vcsd_dsp_message: MSG_TX_GET_TX_STAT: num_tx_pkts=1019 num_signaling_pkts=0
num_comfort_noise_pkts=0 transmit_durtation=24150 voice_transmit_duration=20380
fax_transmit_duration=0
```

debug vpm error

To enable DSP error tracing in voice port modules (VPMs), use the **debug vpm error** command. To disable DSP error tracing, use the **no** form of this command.

debug vpm error

no debug vpm error

Syntax Description This command has no arguments or keywords.

Defaults VPM debugging is not enabled.

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(7)XK | This command was introduced on the Cisco 2600, 3600, and MC3810 series devices. |
| | 12.1(2)T | This command was integrated into 12.1(2)T release. |

Usage Guidelines Execution of **no debug all** will turn off all port level debugging. You should turn off all debugging and then enter the debug commands you are interested in one by one. This will help avoid confusion about which ports you are actually debugging.

Examples The following example shows **debug vpm error** messages for Cisco 2600 or 3600 series router or a Cisco MC3810 series concentrator:

```
Router# deb vpm error
00:18:37: [1:0.1, FXSLS_NULL, E_DSP_SIG_0100] -> ERROR:INVALID INPUT
Router#
```

The following example turns off **debug vpm error** debugging messages:

```
Router# no debug vpm error
```

| Related Commands | Command | Description |
|------------------|--------------------------------|--|
| | debug vpm all | Enables all VPM debugging. |
| | debug vpm port | Limits the debug vpm error command to a specified port. |
| | show debug | Displays which debug commands are enabled. |

debug vpm port

To observe the behavior of the Holst state machine, use the **debug vpm port** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm port [*slot-number* | *subunit-number* | *port*]

no debug vpm [*slot-number* | *subunit-number* | *port*]

| Syntax Description | | |
|-----------------------|---|------------------------------|
| <i>slot-number</i> | (Optional) Specifies the slot number in the Cisco router where the voice interface card is installed. Valid entries are from 0 to 3, depending on the router being used and the slot where the voice interface card has been installed. | |
| <i>subunit-number</i> | (Optional) Specifies the subunit on the voice interface card where the voice port is located. Valid entries are 0 or 1. | |
| <i>port</i> | (Optional) Specifies the voice port. Valid entries are 0 or 1. | |
| Command History | Release | Modification |
| | 11.3(1) | This command was introduced. |

Usage Guidelines

This command is not supported on Cisco 7200 series routers or on the Cisco MC3810.

Use this command to limit the debug output to a particular port. The debug output can be quite voluminous for a single channel. A 12-port box might create problems. Use this debug command with any or all of the other debug modes.

Execution of **no debug vpm all** will turn off all port level debugging. Cisco recommends that you turn off all debugging and then enter the debug commands you are interested in one by one. This process helps to avoid confusion about which ports you are actually debugging.

Examples

The following example shows sample output from the **debug vpm port 1/1/0** command during trunk establishment after the **no shutdown** command has been executed on the voice port:

```
Router# debug vpm port 1/1/0

*Mar 1 03:21:39.799: htsp_process_event: [1/1/0, 0.1 , 2]act_down_inserve
*Mar 1 03:21:39.807: htsp_process_event: [1/1/0, 0.0 , 14]
  act_go_trunkhtsp_trunk_createhtsp_trunk_sig_linkfxols_trunk
*Mar 1 03:21:39.807: htsp_process_event: [1/1/0, 1.0 , 1]trunk_offhookfxols_trunk_down
*Mar 1 03:21:39.807: dsp_sig_encap_config: [1/1/0] packet_len=28 channel_id=128
  packet_id=42 transport_protocol=1 playout_delay=100 signaling_mode=0
  t_ssrc=0 r_ssrc=0 t_vpxcc=0 r_vpxcc=0
*Mar 1 03:21:39.811: dsp_set_sig_state: [1/1/0] packet_len=12
  channel_id=128 packet_id=39 state=0xC timestamp=0x0
*Mar 1 03:21:39.811: trunk_offhook: Trunk Retry Timer Enabled
*Mar 1 03:22:13.095: htsp_process_event: [1/1/0, 1.1, 39]act_trunk_setuphtsp_setup_ind
*Mar 1 03:22:13.095: htsp_process_event: [1/1/0, 1.2 , 8]
*Mar 1 03:22:13.099: hdsprm_vtsp_codec_loaded_ok: G726 firmware needs download
*Mar 1 03:22:13.103: dsp_download: p=0x60E73844 size=34182 (t=1213310):39 FA 6D
*Mar 1 03:22:13.103: htsp_process_event: [1/1/0, 1.2 , 6]act_trunk_proc_connect
*Mar 1 03:22:13.191: dsp_receive_packet: MSG_TX_RESTART_INDICATION: code=0 t=1213319
*Mar 1 03:22:13.191: dsp_download: p=0x60EA8924 size=6224 (t=1213319): 8 55 AE
```

```
*Mar 1 03:22:13.207: dsp_receive_packet: MSG_TX_RESTART_INDICATION: code=0 t=1213320
*Mar 1 03:22:13.207: htsp_process_event: [1/1/0, 1.3 , 11] trunk_upfxols_trunk_up
*Mar 1 03:22:13.207: dsp_set_sig_state: [1/1/0] packet_len=12
    channel_id=128 packet_id=39 state=0x4 timestamp=0x0
*Mar 1 03:22:13.207: dsp_sig_encap_config: [1/1/0] packet_len=28 channel_id=128
    packet_id=42 transport_protocol=3 playout_delay=100 headerbytes = 0xA0
```

Note in the above display that “transport_protocol = 3” indicates Voice-over-Frame Relay. Also note that the second line of the display indicates that a **shutdown/no shutdown** command sequence was executed on the voice port.

Related Commands

| Command | Description |
|---------------------------------------|--|
| debug vpdn pppoe-data | Enables debugging of all VPM areas. |
| debug vpm dsp | Shows messages from the DSP on the VPM to the router. |
| debug vpm signal | Collects debug information only for signalling events. |
| debug vpm spi | Displays information about how each network indication and application request is handled. |

debug vpm signal

To collect debug information only for signalling events, use the **debug vpm signal** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm signal

no debug vpm signal

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vpm signal** command collects debug information only for signalling events. This command can also be useful in resolving problems with signalling to a PBX.

Examples

The following output shows that a ring is detected, and that the router waits for the ringing to stop before accepting the call:

```
ssm_process_event: [1/0/1, 0.2, 15] fxols_onhook_ringing
ssm_process_event: [1/0/1, 0.7, 19] fxols_ringing_not
ssm_process_event: [1/0/1, 0.3, 6]
ssm_process_event: [1/0/1, 0.3, 19] fxols_offhook_clear
```

The following output shows that the call is connected:

```
ssm_process_event: [1/0/1, 0.3, 4] fxols_offhook_proc
ssm_process_event: [1/0/1, 0.3, 8] fxols_proc_voice
ssm_process_event: [1/0/1, 0.3, 5] fxols_offhook_connect
```

The following output confirms a disconnect from the switch and release with higher layer code:

```
ssm_process_event: [1/0/1, 0.4, 27] fxols_offhook_disc
ssm_process_event: [1/0/1, 0.4, 33] fxols_disc_confirm
ssm_process_event: [1/0/1, 0.4, 3] fxols_offhook_release
```

debug vpm signaling

To see information about the voice port module signalling, use the **debug vpm signaling** command. To disable debugging output, use the **no** form of this command.

debug vpm signaling

no debug vpm 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 12.1(2)T release. |

Examples The following example shows output from the command:

```
Router# debug vpm signaling

01:52:55: [1:1.1, S_TRUNK_BUSYOUT, E_HTSP_OUT_BUSYOUT]
01:52:55: htsp_timer - 0 msec
01:52:55: [1:1.1, S_TRUNK_PEND, E_HTSP_EVENT_TIMER]
01:52:55: htsp_timer_stop htsp_setup_ind
01:52:55: htsp_timer - 2000 msec
01:52:55: [1:1.1, S_TRUNK_PROC, E_HTSP_SETUP_ACK]
01:52:55: htsp_timer_stop
01:52:55: htsp_timer - 20000 msec
01:52:55: [1:6.6, S_TRUNK_PROC, E_HTSP_SETUP_ACK]
01:52:55: htsp_timer_stop
01:52:55: htsp_timer - 20000 msec
01:52:55: [1:1.1, S_TRUNK_PROC, E_HTSP_VOICE_CUT_THROUGH]
01:52:55: %HTSP-5-UPDOWN: Trunk port(channel) [1:1.1] is up
```

debug vpm spi

To trace how the voice port module SPI interfaces with the call control API, use the **debug vpm spi** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm spi

no debug vpm spi

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vpm spi** command traces how the voice port module SPI interfaces with the call control API. This debug command displays information about how each network indication and application request is handled.

This debug level shows the internal workings of the voice telephony call state machine.

Examples

The following output shows that the call is accepted and pre-sented to a higher layer code:

```
dsp_set_sig_state: [1/0/1] packet_len=14 channel_id=129 packet_id=39 state=0xC
timestamp=0x0
vcs_m_process_event: [1/0/1, 0.5, 1] act_up_setup_ind
```

The following output shows that the higher layer code accepts the call, requests addressing information, and starts DTMF and dial-pulse collection. It also shows that the digit timer is started.

```
vcs_m_process_event: [1/0/1, 0.6, 11] act_setup_ind_ack
dsp_voice_mode: [1/0/1] packet_len=22 channel_id=1 packet_id=73 coding_type=1
voice_field_size=160 VAD_flag=0 echo_length=128 comfort_noise=1 fax_detect=1
dsp_dtmf_mode: [1/0/1] packet_len=12 channel_id=1 packet_id=65 dtmf_or_mf=0
dsp_CP_tone_on: [1/0/1] packet_len=32 channel_id=1 packet_id=72 tone_id=3 n_freq=2
freq_of_first=350 freq_of_second=440 amp_of_first=4000 amp_of_second=4000 direction=1
on_time_first=65535 off_time_first=0 on_time_second=65535 off_time_second=0
dsp_digit_collect_on: [1/0/1] packet_len=22 channel_id=129 packet_id=35
min_inter_delay=550 max_inter_delay=3200 mim_make_time=18 max_make_time=75
min_brake_time=18 max_brake_time=75
vcs_m_timer: 46653
```

The following output shows the collection of digits one by one until the higher level code indicates it has enough. The input timer is restarted with each digit and the device waits in idle mode for connection to proceed.

```
vcs_m_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcs_m_timer: 47055
vcs_m_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcs_m_timer: 47079
vcs_m_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcs_m_timer: 47173
vcs_m_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcs_m_timer: 47197
vcs_m_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
```

```
vcsn_timer: 47217
vcsn_process_event: [1/0/1, 0.7, 13] act_dcollect_proc
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
dsp_digit_collect_off: [1/0/1] packet_len=10 channel_id=129 packet_id=36
dsp_idle_mode: [1/0/1] packet_len=10 channel_id=1 packet_id=68
```

The following output shows that the network voice path cuts through:

```
vcsn_process_event: [1/0/1, 0.8, 15] act_bridge
vcsn_process_event: [1/0/1, 0.8, 20] act_caps_ind
vcsn_process_event: [1/0/1, 0.8, 21] act_caps_ack
dsp_voice_mode: [1/0/1] packet_len=22 channel_id=1 packet_id=73 coding_type=6
voice_field_size=20 VAD_flag=1 echo_length=128 comfort_noise=1 fax_detect=1
```

The following output shows that the called-party end of the connection is connected:

```
vcsn_process_event: [1/0/1, 0.8, 8] act_connect
```

The following output shows the voice quality statistics collected periodically:

```
vcsn_process_event: [1/0/1, 0.13, 17]
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=0
vcsn_process_event: [1/0/1, 0.13, 28]
vcsn_process_event: [1/0/1, 0.13, 29]
vcsn_process_event: [1/0/1, 0.13, 32]
vcsn_process_event: [1/0/1, 0.13, 17]
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=0
vcsn_process_event: [1/0/1, 0.13, 28]
vcsn_process_event: [1/0/1, 0.13, 29]
vcsn_process_event: [1/0/1, 0.13, 32]
vcsn_process_event: [1/0/1, 0.13, 17]
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=0
vcsn_process_event: [1/0/1, 0.13, 28]
vcsn_process_event: [1/0/1, 0.13, 29]
vcsn_process_event: [1/0/1, 0.13, 32]
```

The following output shows that the disconnection indication is passed to higher level code. The call connection is torn down, and final call statistics are collected:

```
vcsn_process_event: [1/0/1, 0.13, 4] act_generate_disc
vcsn_process_event: [1/0/1, 0.13, 16] act_bdrop
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcsn_process_event: [1/0/1, 0.13, 18] act_disconnect
dsp_get_levels: [1/0/1] packet_len=10 channel_id=1 packet_id=89
vcsn_timer: 48762
vcsn_process_event: [1/0/1, 0.15, 34] act_get_levels
dsp_get_tx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=86 reset_flag=1
vcsn_process_event: [1/0/1, 0.15, 31] act_stats_complete
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
dsp_digit_collect_off: [1/0/1] packet_len=10 channel_id=129 packet_id=36
dsp_idle_mode: [1/0/1] packet_len=10 channel_id=1 packet_id=68
vcsn_timer: 48762
dsp_set_sig_state: [1/0/1] packet_len=14 channel_id=129 packet_id=39 state=0x4
timestamp=0x0
vcsn_process_event: [1/0/1, 0.16, 5] act_wrelease_release
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
dsp_idle_mode: [1/0/1] packet_len=10 channel_id=1 packet_id=68
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=1
```

debug vpm trunk_sc

To enable the display of trunk conditioning supervisory component trace information, use the **debug vpm trunk_sc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm trunk_sc

no debug vpm trunk_sc

Syntax Description This command has no arguments or keywords.

Defaults Trunk conditioning supervisory component trace information is not displayed.

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(7)XK | This command was introduced on the Cisco 2600, 3600, and MC3810 series devices. |
| | 12.1(2)T | This command was integrated into the 12.1(2)T release. |

Usage Guidelines Use the **debug vpm port** command with the *slot-number/subunit-number/port* argument to limit the **debug vpm trunk_sc** debug output to a particular port. If you do not use the **debug vpm port** command, the **debug vpm trunk_sc** displays output for all ports.

Execution of the **no debug all** command will turn off all port level debugging. It is usually a good idea to turn off all debugging and then enter the debug commands you are interested in one by one. This process helps avoid confusion about which ports you are actually debugging.

Examples The following example shows **debug vpm trunk_sc** messages for port 1/0/0 on a Cisco 2600 or 3600 series router:

```
Router# debug vpm trunk_sc
```

```
Router# debug vpm port 1/0/0
```

The following example shows **debug vpm trunk_sc** messages for port 1/1 on a Cisco MC3810 device:

```
Router# debug vpm trunk_sc
```

```
Router# debug vpm port 1/1
```

The following example turns off **debug vpm trunk_sc** debugging messages:

```
Router# no debug vpm trunk_sc
```

Related Commands

| Command | Description |
|-----------------------|---|
| debug vpm all | Enables all VPM debugging |
| debug vpm port | Limits the debug vpm trunk_sc command to a specified port. |
| show debug | Displays which debug commands are enabled. |

debug vpm voaal2 all

To display type 1 (voice) and type 3 (control) ATM Adaptation Layer type 2 (AAL2) packets sent to and received from the domain-specific part (DSP), use the **debug vpm voaal2 all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug vpm voaal2 all {all_dsp | from_dsp | to_dsp}
```

```
no debug vpm voaal2 all
```

Syntax Description

| | |
|-----------------|---------------------------------------|
| all_dsp | Display messages to and from the DSP. |
| from_dsp | Display messages from the DSP. |
| to_dsp | Display messages to the DSP. |

Defaults

Debugging for display of AAL2 packets is not enabled.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|---|
| 12.1(1)XA | This command was introduced for the Cisco MC3810 series. |
| 12.1(2)T | This command was integrated in Cisco IOS Release 12.1(2)T. |
| 12.2(2)T | Support for this command was introduced on the Cisco 7200 series routers. |

Usage Guidelines

Do not enter this debug command on a system carrying live traffic. Continuous display of AAL2 type 1 (voice) packets results in high CPU utilization and loss of console access to the system. Calls will be dropped and trunks may go down. For AAL2 debugging, use the **debug vpm voaal2 type3** debug command and identify a specific type 3 (control) packet type.

Examples

The following example shows a sample output from the **debug vpm voaal2 all** command, where the example selection is to display channel-associated switching (CAS) packets sent to and from the DSP:

```
Router# debug vpm voaal2 all all_dsp

*Jan  9 20:10:36.965:TYPE 3, len = 8, cid = 34, uui = 24 :TO_DSP
*Jan  9 20:10:36.965:CAS
  redundancy = 3, timestamp = 10270, signal = 0
- 22 13 12 E8 1E 0 E 15 -

*Jan  9 20:10:41.617:TYPE 3, len = 8, cid = 34, uui = 24 :FROM_DSP
*Jan  9 20:10:41.617:CAS
  redundancy = 3, timestamp = 980, signal = 0
- 22 13 12 C3 D4 0 F 87 -
```

```

*Jan  9 20:10:41.965:TYPE 3, len = 8, cid = 34, uui = 24 :TO_DSP
*Jan  9 20:10:41.965:CAS
  redundancy = 3, timestamp = 10270, signal = 0
- 22 13 12 E8 1E 0 E 15 -

*Jan  9 20:10:46.621:TYPE 3, len = 8, cid = 34, uui = 24 :FROM_DSP
*Jan  9 20:10:46.621:CAS
  redundancy = 3, timestamp = 980, signal = 0
- 22 13 12 C3 D4 0 F 87 -

....

*Jan  9 20:10:57.101:TYPE 1, len = 43, cid = 34, uui = 8- 22 9D 1 CC FC
C7
  3E 22 23 FE DF F8 DE 1C FF E5 12 22 43 EC 2E 9E CC DE A7 EF 14 E3 F1 2C
2D
  BC 1B FC FE D7 E1 1F 2F ED 11 FC 1F -

*Jan  9 20:10:57.105:TYPE 3, len = 9, cid = 34, uui = 24 :FROM_DSP
*Jan  9 20:10:57.105:DIALED DIGITS
  redundancy = 0,
                                timestamp = 940, digitcode = 1
- 22 17 3 3 AC 1 1 8 E5 -

*Jan  9 20:10:57.113:TYPE 1, len = 43, cid = 34, uui = 10- 22 9D 4B 3F
1F
11 FC CD CC BE B7 E2 F3 32 2E 1F F9 DA CC BF 12 F1 37 31 11 2C FE 9D DA
D2
E1 C7 4A 34 3F FA 21 AD CC 1F EE 16 E1 -

*Jan  9 20:10:57.113:TYPE 3, len = 9, cid = 34, uui = 24 :FROM_DSP
*Jan  9 20:10:57.113:DIALED DIGITS
  redundancy = 1,
                                timestamp = 940, digitcode = 1
- 22 17 3 43 AC 1 1 B 12 -

*Jan  9 20:10:57.121:TYPE 1, len = 43, cid = 34, uui = 12- 22 9D 95 F1
1E
E1 DF 1E 21 31 21 1D D9 EB BB DF 22 17 13 12 1F 58 FF ED ED E1 4D B7 3E
3F
21 F3 8E FD EF DF F4 12 E4 32 FE B4 D8 -

```

Related Commands

| Command | Description |
|-------------------------------|---|
| debug vpm voaal2 type1 | Displays type 1 (voice) AAL2 packets sent to and received from the DSP. |
| debug vpm voaal2 type3 | Displays type 3 (control) AAL2 packets sent to and received from the DSP. |
| show debug | Shows which debug commands are enabled. |

debug vpm voaal2 type1

To display type 1 (voice) ATM Adaptation Layer type 2 (AAL2) packets sent to and received from the domain-specific part (DSP), use the **debug vpm voaal2 type1** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug vpm voaal2 type1 {all_dsp | from_dsp | to_dsp}
```

```
no debug vpm voaal2 type1
```

| Syntax Description | all_dsp | Display messages to and from the DSP. |
|--------------------|----------|---------------------------------------|
| | from_dsp | Display messages from the DSP. |
| | to_dsp | Display messages to the DSP. |

Defaults Debugging for display of AAL2 packets is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.1(1)XA | This command was introduced for the Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated in Cisco IOS Release 12.1(2)T. |
| | 12.2(2)T | Support for this command was introduced on the Cisco 7200 series routers. |

Usage Guidelines Do not enter this debug command on a system carrying live traffic. Continuous display of AAL2 type 1 (voice) packets results in high CPU utilization and loss of console access to the system. Calls will be dropped and trunks may go down. For AAL2 debugging, use the **debug vpm voaal2 type3** debug command and identify a specific type 3 (control) packet type.

Examples The following example shows a sample output from the **debug vpm voaal2 type1** command:



Note

The display of voice packets on a live system will continue indefinitely. The debugging output cannot be interrupted, because console access will be lost.

```
Router# debug vpm voaal2 type1 all_dsp
```

```
TYPE 1, len = 43, cid = 17, uui = 15- 11 9D E6 1B 52 9D 95 9B DB 1D 14
1C 5F 9C 95 9C EA 1C 15 1B 74 9C 94 9D 6B 1C 14 1D E4 9B 94 9D 5B 1B 14
1D D7 9B 94 9D 50 1B 14 -
```

```
TYPE 1, len = 43, cid = 22, uui = 15- 16 9D ED 1D 14 1B 53 9D 94 9C DB
1D 14 1C 5F 9C 95 9C EB 1C 14 1C 78 9D 94 9D 6F 1C 14 1E E4 9B 94 9D 5B
1B 14 1D D7 9B 94 9E 52 -
```

```
TYPE 1, len = 43, cid = 12, uui = 14- C 9D D1 29 AB 96 96 A9 2B 16 16 2A
AA 96 96 AB 2A 16 17 2B A9 96 97 AC 28 16 17 2C A8 96 97 AD 27 15 17 2E
A7 97 97 AE 26 16 17 -
```

```
TYPE 1, len = 43, cid = 34, uui = 14- 22 9D DF D7 31 20 19 15 14 15 19
1E 2C 60 AF 9F 99 96 94 95 99 9F AD EC 2F 1F 1A 15 14 15 19 1F 2E ED AD
9F 99 96 93 95 99 9F AF -
```

```
TYPE 1, len = 43, cid = 12, uui = 15- C 9D F4 2F A5 96 97 AF 25 15 18 31
A4 95 98 B3 23 15 18 33 A3 95 98 B5 22 15 18 37 A2 95 98 B7 21 15 18 39
A0 95 99 BB 21 14 19 -
```

```
TYPE 1, len = 43, cid = 34, uui = 15- 22 9D FA 5D 2D 1E 19 15 14 15 1A
21 31 D9 AC 9E 98 95 94 95 9A A4 B3 52 2B 1D 18 14 14 16 1B 22 36 CA AA
9D 98 94 94 96 9B A4 B6 -
```

Related Commands

| Command | Description |
|-------------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| debug vpm voaal2 all | Displays type 1 (voice) and type 3 (control) AAL2 packets sent to and received from the DSP. |
| debug vpm voaal2 type3 | Displays type 3 (control) AAL2 packets sent to and received from the DSP. |
| show debug | Shows which debug commands are enabled. |

debug vpm voaal2 type3

To display type 3 (control) ATM Adaptation Layer type 2 (AAL2) packets sent to and received from the domain-specific part (DSP), use the **debug vpm voaal2 type3** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug vpm voaal2 type3 {alarms | alltype3 | cas | dialed | faxrelay | state} {all_dsp | from_dsp | to_dsp}
```

```
no debug vpm voaal2 type3
```

| Syntax Description | | |
|--------------------|-----------------|--|
| | alarms | Display type 3 alarm packets. |
| | alltype3 | Display all type 3 packets. |
| | cas | Display type 3 channel-associated switching (CAS) packets. |
| | dialed | Display type 3 dialed digit packets. |
| | faxrelay | (Not supported) Display type 3 fax relay packets. |
| | state | Display type 3 user state packets. |
| | all_dsp | Display messages to and from the DSP. |
| | from_dsp | Display messages from the DSP. |
| | to_dsp | Display messages to the DSP. |

Defaults Debugging for display of AAL2 packets is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.1(1)XA | This command was introduced for the Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated in Cisco IOS Release 12.1(2)T. |
| | 12.2(2)T | Support for this command was introduced on the Cisco 7200 series routers. |

Usage Guidelines This is the preferred debug command for displaying specific types of control packets. It is usually preferable to specify a particular type of control packet rather than use the **alltype3** to avoid excessive output display and CPU utilization.

Examples

The following example shows a sample output from the **debug vpm voaal2 type3** command, where the example selection is to display messages to and from the DSP:

```
Router# debug vpm voaal2 type3 all_dsp

00:43:02:TYPE 3, len = 8, cid = 58, uui = 24 :TO_DSP
00:43:02:CAS
  redundancy = 3, timestamp = 10484, signal = 0
- 3A 13 18 E8 F4 0 C DA -

00:43:02:TYPE 3, len = 8, cid = 93, uui = 24 :FROM_DSP
00:43:02:CAS
  redundancy = 3, timestamp = 6528, signal = 0
- 5D 13 1E D9 80 0 F 33 -

00:43:02:TYPE 3, len = 8, cid = 102, uui = 24 :FROM_DSP
00:43:02:CAS
  redundancy = 3, timestamp = 5988, signal = 0
- 66 13 4 D7 64 0 F DF -

00:43:02:TYPE 3, len = 8, cid = 194, uui = 24 :FROM_DSP
00:43:02:CAS
  redundancy = 3, timestamp = 6212, signal = 0
- C2 13 10 D8 44 0 F AC -

00:43:02:TYPE 3, len = 8, cid = 92, uui = 24 :FROM_DSP
TYPE 3, len = 8, cid = 66, uui = 24 :TO_DSP:43:00:CAS
  redundancy = 3, times signal = 0
- 5C 13 5 D9 E4 0 C 1F -

00:43:02:TYPE 3, len = 8, cid = 40, uui = 24 :TO_DSP
00:43:02:CAS
  redundancy = 3, timestamp = 8658, signal = 0
- 28 13 7 E1 D2 0 E 79 -

00:43:02:TYPE 3, len = 8, cid = 137, uui = 24 :FROM_DSP
00:43:02:CAS
  redundancy = 3, timestamp = 6836, signal = 0
- 89 13 B DA B4 0 E 78 -
```

Related Commands

| Command | Description |
|-------------------------------|---|
| debug vpm voaal2 type1 | Displays type 1 (voice) AAL2 packets sent to and received from the DSP. |
| debug vpm voaal2 type3 | Displays type 3 (control) AAL2 packets sent to and received from the DSP. |
| show debug | Shows which debug commands are enabled. |

debug vrrp all

To display debug messages for Virtual Router Redundancy Protocol (VRRP) errors, events, and state transitions, use the **debug vrrp all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vrrp all

no debug vrrp all

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|------------|---|
| | 12.0(18)ST | This command was introduced. |
| | 12.0(22)S | This command was integrated into Cisco IOS Release 12.0(22)S. |
| | 12.2(13)T | This command was integrated into Cisco IOS Release 12.2(13)T. |

Examples The following example provides sample output for the **debug vrrp all** command:

```
Router# debug vrrp all

00:15:30: %IP-4-DUPADDR: Duplicate address 10.18.0.2 on Ethernet1/0, sourced by
0000.5e00.0101
May 22 18:41:54.447: VRRP: Grp 1 Advertisement Primary address 10.18.0.2
different from ours 10.18.0.1
May 22 18:41:57.443: VRRP: Grp 1 Advertisement Primary address 10.18.0.2
different from ours 10.18.0.1
May 22 18:42:00.443: VRRP: Grp 1 Advertisement Primary address 10.18.0.2
different from ours 10.18.0.1

May 22 18:48:41.521: VRRP: Grp 1 Event - Advert higher or equal priority
May 22 18:48:44.521: VRRP: Grp 1 Event - Advert higher or equal priority
May 22 18:48:47.521: VRRP: Grp 1 Event - Advert higher or equal priority

May 22 18:53:23.390: VRRP: Grp 1 changing to V_STATE_INIT

May 22 18:54:26.143: VRRP: Grp 1 changing to V_STATE_BACKUP
May 22 18:54:35.755: VRRP: Grp 1 changing to V_STATE_MASTER
May 22 18:53:23.390: VRRP: Grp 1 changing to V_STATE_INIT

May 22 18:54:26.143: VRRP: Grp 1 changing to V_STATE_BACKUP
May 22 18:54:35.755: VRRP: Grp 1 changing to V_STATE_MASTER
```

Related Commands

| Command | Description |
|-----------------------------------|---|
| debug vrrp error | Displays debug messages about VRRP error conditions. |
| debug vrrp events | Displays debug messages about VRRP events. |
| debug vrrp state | Displays debug messages about the VRRP state transitions. |

debug vrrp error

To display debug messages about Virtual Router Redundancy Protocol (VRRP) error conditions, use the **debug vrrp error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vrrp error

no debug vrrp error

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|------------|---|
| | 12.0(18)ST | This command was introduced. |
| | 12.0(22)S | This command was integrated into Cisco IOS Release 12.0(22)S. |
| | 12.2(13)T | This command was integrated into Cisco IOS Release 12.2(13)T. |

Examples The following example provides sample output for the **debug vrrp error** command:

```
Router# debug vrrp error

Router#
00:15:30: %IP-4-DUPADDR: Duplicate address 10.18.0.2 on Ethernet1/0, sourced by
0000.5e00.0101
May 22 18:41:54.447: VRRP: Grp 1 Advertisement Primary address 10.18.0.2
different from ours 10.18.0.1
May 22 18:41:57.443: VRRP: Grp 1 Advertisement Primary address 10.18.0.2
different from ours 10.18.0.1
May 22 18:42:00.443: VRRP: Grp 1 Advertisement Primary address 10.18.0.2
different from ours 10.18.0.1
```

In the example, the error being observed is that the router has a virtual address of 10.18.0.1 for group 1, but it received a virtual address of 10.18.0.2 for group 1 from another router on the same LAN.

| Related Commands | Command | Description |
|------------------|--------------------------------|---|
| | debug vrrp all | Displays debug messages for VRRP errors, events, and state transitions. |

debug vrrp events

To display debug messages about Virtual Router Redundancy Protocol (VRRP) events that are occurring, use the **debug vrrp events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vrrp events

no debug vrrp events

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|------------|---|
| | 12.0(18)ST | This command was introduced. |
| | 12.0(22)S | This command was integrated into Cisco IOS Release 12.0(22)S. |
| | 12.2(13)T | This command was integrated into Cisco IOS Release 12.2(13)T. |

Examples The following example provides sample output for the **debug vrrp events** command:

```
Router# debug vrrp events
```

```
May 22 18:48:41.521: VRRP: Grp 1 Event - Advert higher or equal priority  
May 22 18:48:44.521: VRRP: Grp 1 Event - Advert higher or equal priority  
May 22 18:48:47.521: VRRP: Grp 1 Event - Advert higher or equal priority
```

In the example, the event being observed is that the router received an advertisement from another router for group 1 that has a higher or equal priority to itself.

| Related Commands | Command | Description |
|------------------|--------------------------------|---|
| | debug vrrp all | Displays debug messages for VRRP errors, events, and state transitions. |

debug vrrp packets

To display summary information about Virtual Router Redundancy Protocol (VRRP) packets being sent or received, use the **debug vrrp packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vrrp packets

no debug vrrp packets

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|------------|---|
| | 12.0(18)ST | This command was introduced. |
| | 12.0(22)S | This command was integrated into Cisco IOS Release 12.0(22)S. |
| | 12.2(13)T | This command was integrated into Cisco IOS Release 12.2(13)T. |

Examples The following example provides sample output for the **debug vrrp packets** command. The output is on the master virtual router; the router for group 1 is sending an advertisement with a checksum of 6BE7.

```
Router# debug vrrp packets
```

```
VRRP Packets debugging is on
```

```
May 22 18:51:03.220: VRRP: Grp 1 sending Advertisement checksum 6BE7
May 22 18:51:06.220: VRRP: Grp 1 sending Advertisement checksum 6BE7
```

In the following example, the router with physical address 10.18.0.3 is advertising a priority of 105 for VRRP group 1:

```
Router# debug vrrp packets
```

```
VRRP Packets debugging is on
```

```
May 22 18:51:09.222: VRRP: Grp 1 Advertisement priority 105, ipaddr 10.18.0.3
May 22 18:51:12.222: VRRP: Grp 1 Advertisement priority 105, ipaddr 10.18.0.3
```

debug vrrp state

To display debug messages about the state transitions occurring for Virtual Router Redundancy Protocol (VRRP) groups, use the **debug vrrp state** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vrrp state

no debug vrrp state

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|------------|---|
| | 12.0(18)ST | This command was introduced. |
| | 12.0(22)S | This command was integrated into Cisco IOS Release 12.0(22)S. |
| | 12.2(13)T | This command was integrated into Cisco IOS Release 12.2(13)T. |

Examples The following example provides sample output for the **debug vrrp state** command:

```
Router# debug vrrp state
```

```
May 22 18:53:23.390: VRRP: Grp 1 changing to V_STATE_INIT
```

```
May 22 18:54:26.143: VRRP: Grp 1 changing to V_STATE_BACKUP
```

```
May 22 18:54:35.755: VRRP: Grp 1 changing to V_STATE_MASTER
```

| Related Commands | Command | Description |
|------------------|--------------------------------|---|
| | debug vrrp all | Displays debug messages for VRRP errors, events, and state transitions. |

debug vsi api

To display information on events associated with the external ATM application programming interface (API) interface to the Virtual Switch Interface (VSI) master, use the **debug vsi api** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vsi api

no debug vsi api

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History

| Release | Modification |
|----------|--|
| 12.0(5)T | This command was introduced. |
| 12.2(4)T | This command was integrated into Cisco IOS 12.2(4)T. |

Usage Guidelines

Use the **debug vsi api** command to monitor the communication between the VSI master and the XmplsATM component regarding interface changes and cross-connect requests.

Examples

The following is sample output from the **debug vsi api** command:

```
Router# debug vsi api

VSI_M: vsi_exatm_conn_req: 0x000C0200/1/35 -> 0x000C0100/1/50
        desired state up, status OK
VSI_M: vsi_exatm_conn_resp: 0x000C0200/1/33 -> 0x000C0100/1/49
        curr state up, status OK
```

[Table 300](#) describes the significant fields shown in the display.

Table 300 *debug vsi api* Command Field Descriptions

| Field | Description |
|--------------------|---|
| vsi_exatm_conn_req | The type of connection request (connect or disconnect) that was submitted to the VSI master. |
| 0x000C0200 | The logical interface identifier of the primary endpoint, in hexadecimal form. |
| 1/35 | The virtual path identifier (VPI) and virtual channel identifier (VCI) of the primary endpoint. |

Table 300 *debug vsi api Command Field Descriptions (continued)*

| Field | Description |
|---------------------------------------|---|
| -> | The type of traffic flow. A right arrow (->) indicates unidirectional traffic flow (from the primary endpoint to the secondary endpoint). A bidirectional arrow (<->) indicates bidirectional traffic flow. |
| 0x000C0100 | Logical interface identifier of the secondary endpoint. |
| 1/50 | VPI and VCI of the secondary endpoint. |
| desired state | The status of a connect request. Up indicates a connect request; Down indicates a disconnect request. |
| status (in vsi_exatm_conn_req output) | <p>The status of a request. One of following status indications appears:</p> <p>OK INVALID_ARGS NONEXIST_INTF TIMEOUT NO_RESOURCES FAIL</p> <p>OK means only that the request is successfully queued for transmission to the switch; it does not indicate completion of the request.</p> |

debug vsi errors

To display information about errors encountered by the Virtual Switch Interface (VSI) master, use the **debug vsi errors** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vsi errors [**interface** *interface* [**slave number**]]

no debug vsi errors [**interface** *interface* [**slave number**]]

Syntax Description

| | |
|-----------------------------------|---|
| interface <i>interface</i> | (Optional) Specifies the interface number. |
| slave number | (Optional) Specifies the slave number (beginning with 0). |

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|--|
| 12.0(5)T | This command was introduced. |
| 12.2(4)T | This command was integrated into Cisco IOS 12.2(4)T. |

Usage Guidelines

Use the **debug vsi errors** command to display information about errors encountered by the VSI master when parsing received messages, as well as information about unexpected conditions encountered by the VSI master.

If the interface parameter is specified, output is restricted to errors associated with the indicated VSI control interface. If the slave number is specified, output is further restricted to errors associated with the session with the indicated slave.



Note

Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple commands that specify slave numbers allow multiple slaves to be debugged immediately. For example, the following commands display errors associated with sessions 0 and 1 on control interface atm2/0, but for no other sessions.

```
Router# debug vsi errors interface atm2/0 slave 0
Router# debug vsi errors interface atm2/0 slave 1
```

Some errors are not associated with any particular control interface or session. Messages associated with these errors are printed, regardless of the **interface** or **slave** options currently in effect.

Examples

The following is sample output from the **debug vsi errors** command:

```
Router# debug vsi errors
```

```
VSI Master: parse error (unexpected param-group contents) in GEN ERROR RSP rcvd on
ATM2/0:0/51 (slave 0)
```

```
    errored section is at offset 16, for 2 bytes:
```

```
    01.01.00.a0 00.00.00.00 00.12.00.38 00.10.00.34
*00.01*00.69 00.2c.00.00 01.01.00.80 00.00.00.08
00.00.00.00 00.00.00.00 00.00.00.00 0f.a2.00.0a
00.01.00.00 00.00.00.00 00.00.00.00 00.00.00.00
00.00.00.00
```

[Table 301](#) describes the significant fields shown in the display.

Table 301 *debug vsi errors Command Field Descriptions*

| Field | Description |
|---------------------------------|--|
| parse error | An error was encountered during the parsing of a message received by the VSI master. |
| unexpected param-group contents | The type of parsing error. In this case, a parameter group within the message contained invalid data. |
| GEN ERROR RSP | The function code in the header of the error message. |
| ATM2/0 | The control interface on which the error message was received. |
| 0/51 | The virtual path identifier (VPI) or virtual channel identifier (VCI) of the virtual circuit (VC) (on the control interface) on which the error message is received. |
| slave | Number of the session on which the error message is received. |
| offset <n> | The number of bytes between the start of the VSI header and the start of that portion of the message in error. |
| <n> bytes | Length of the error section. |
| 00.01.00.a0 [...] | The entire error message, as a series of hexadecimal bytes. Note that the error section is between asterisks (*). |

debug vsi events

To display information about events that affect entire sessions, as well as events that affect only individual connections, use the **debug vsi events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vsi events [**interface** *interface* [**slave** *number*]]

no debug vsi events [**interface** *interface* [**slave** *number*]]

Syntax Description

| | |
|---|---|
| interface <i>interface</i> (Optional) | The interface number. |
| slave <i>number</i> (Optional) | The slave number (beginning with zero). |

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|--|
| 12.0(5)T | This command was introduced. |
| 12.2(4)T | This command was integrated into Cisco IOS 12.2(4)T. |

Usage Guidelines

Use the **debug vsi events** command to display information about events associated with the per-session state machines of the Virtual Switch Interface (VSI) master, as well as the per-connection state machines. If you specify an interface, the output is restricted to events associated with the indicated VSI control interface. If you specify the slave number, output is further restricted to events associated with the session with the indicated slave.



Note

Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple commands that specify slave numbers allow multiple slaves to be debugged at once. For example, the following commands restrict output to events associated with sessions 0 and 1 on control interface atm2/0, but for no other sessions. Output associated with all per-connection events are displayed, regardless of the **interface** or **slave** options currently in effect.

```
Router# debug vsi events interface atm2/0 slave 0
Router# debug vsi events interface atm2/0 slave 1
```

Examples

The following is sample output from the **debug vsi events** command:

```
Router# debug vsi events

VSI Master: conn 0xC0200/1/37->0xC0100/1/51:
      CONNECTING -> UP
VSI Master(session 0 on ATM2/0):
      event CONN_CMT_RSP, state ESTABLISHED -> ESTABLISHED
VSI Master(session 0 on ATM2/0):
      event KEEPALIVE_TIMEOUT, state ESTABLISHED -> ESTABLISHED
VSI Master(session 0 on ATM2/0):
      event SW_GET_CNFG_RSP, state ESTABLISHED -> ESTABLISHED
debug vsi packets
```

[Table 302](#) describes the significant fields shown in the display.

Table 302 *debug vsi events Command Field Descriptions*

| Field | Description |
|----------------------------|--|
| conn | The event applies to a particular connection. |
| 0xC0200 | Logical interface identifier of the primary endpoint, in hexadecimal form. |
| 1/37 | The virtual path identifier (VPI) or virtual channel identifier (VCI) of the primary endpoint. |
| -> | The type of traffic flow. A right arrow (->) indicates unidirectional traffic flow (from the primary endpoint to the secondary endpoint). A bidirectional arrow (<->) indicates bidirectional traffic flow. |
| 0xC0100 | Logical interface identifier of the secondary endpoint. |
| 1/51 | VPI or VCI of the secondary endpoint. |
| <state1> -> <state2> | <state1> is a mnemonic for the state of the connection before the event occurred. <state2> represents the state of the connection after the event occurred. |
| session | The number of the session with which the event is associated. |
| ATM2/0 | The control interface associated with the session. |
| event | The event that has occurred. This includes mnemonics for the function codes of received messages (for example, CONN_CMT_RSP), as well as mnemonics for other events (for example, KEEPALIVE_TIMEOUT). |
| state <state1> -> <state2> | Mnemonics for the session states associated with the transition triggered by the event. <state1> is a mnemonic for the state of the session before the event occurred; <state2> is a mnemonic for the state of the session after the event occurred. |

debug vsi packets

To display a one-line summary of each Virtual Switch Interface (VSI) message sent and received by the label switch controller (LSC), use the **debug vsi packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vsi packets [*interface interface* [*slave number*]]

no debug vsi packets [*interface interface* [*slave number*]]

Syntax Description

| | |
|-----------------------------------|--|
| interface <i>interface</i> | (Optional) The interface number. |
| slave <i>number</i> | (Optional) The slave number (beginning with zero). |

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|--|
| 12.0(5)T | This command was introduced. |
| 12.2(4)T | This command was integrated into Cisco IOS 12.2(4)T. |

Usage Guidelines

If you specify an interface, output is restricted to messages sent and received on the indicated VSI control interface. If you specify a slave number, output is further restricted to messages sent and received on the session with the indicated slave.



Note

Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple commands that specify slave numbers allow multiple slaves to be debugged at once. For example, the following commands restrict output to messages received on atm2/0 for sessions 0 and 1, but for no other sessions.

```
Router# debug vsi packets interface atm2/0 slave 0
Router# debug vsi packets interface atm2/0 slave 1
```

Examples

The following is sample output from the **debug vsi packets** command:

```
Router# debug vsi packets

VSI master(session 0 on ATM2/0): sent msg SW GET CNFG CMD on 0/51
VSI master(session 0 on ATM2/0): rcvd msg SW GET CNFG RSP on 0/51
VSI master(session 0 on ATM2/0): sent msg SW GET CNFG CMD on 0/51
VSI master(session 0 on ATM2/0): rcvd msg SW GET CNFG RSP on 0/51
```

Table 303 describes the significant fields shown in the display.

Table 303 *debug vsi packets Command Field Descriptions*

| Field | Description |
|---------|--|
| session | Session number identifying a particular VSI slave. Numbers begin with zero. See the show controllers vsi session command. |
| ATM2/0 | Identifier for the control interface on which the message is sent or received. |
| sent | The message is sent by the VSI master. |
| rcvd | The message is received by the VSI master. |
| msg | The function code from the message header. |
| 0/51 | The virtual path identifier (VPI) or virtual channel identifier (VCI) of the virtual circuit (VC) (on the control interface) on which the message is sent or received. |

debug vsi param-groups

To display the first 128 bytes of each Virtual Switch Interface (VSI) message sent and received by the Multiprotocol Label Switching (MPLS) label switch controller (LSC) (in hexadecimal form), use the **debug vsi param-groups** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug vsi param-groups [interface interface [slave number]]
```

```
no debug vsi param-groups [interface interface [slave number]]
```



Note

param-groups stands for parameter groups. A parameter group is a component of a VSI message.

Syntax Description

interface *interface* (Optional) The interface number.

slave number (Optional) The slave number (beginning with zero).

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|--|
| 12.0(5)T | This command was introduced. |
| 12.2(4)T | This command was integrated into Cisco IOS 12.2(4)T. |

Usage Guidelines

This command is most commonly used with the **debug vsi packets** command to monitor incoming and outgoing VSI messages.

If you specify an interface, output is restricted to messages sent and received on the indicated VSI control interface.

If you specify a slave, output is further restricted to messages sent and received on the session with the indicated slave.



Note

Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple commands that specify slave numbers allow multiple slaves to be debugged at once. For example, the following commands restrict output for messages received on atm2/0 for sessions 0 and 1, but for no other sessions:

```
Router# debug vsi param-groups interface atm2/0 slave 0
Router# debug vsi param-groups interface atm2/0 slave 1
```

Examples

The following is sample output from the **debug vsi param-groups** command:

```
Router# debug vsi param-groups

Outgoing VSI msg of 12 bytes (not including encaps):
 01.02.00.80 00.00.95.c2 00.00.00.00
Incoming VSI msg of 72 bytes (not including encaps):
 01.02.00.81 00.00.95.c2 00.0f.00.3c 00.10.00.08
 00.01.00.00 00.00.00.00 01.00.00.08 00.00.00.09
 00.00.00.09 01.10.00.20 01.01.01.00 0c.08.80.00
 00.01.0f.a0 00.13.00.15 00.0c.01.00 00.00.00.00
 42.50.58.2d 56.53.49.31
Outgoing VSI msg of 12 bytes (not including encaps):
 01.02.00.80 00.00.95.c3 00.00.00.00
Incoming VSI msg of 72 bytes (not including encaps):
 01.02.00.81 00.00.95.c3 00.0f.00.3c 00.10.00.08
 00.01.00.00 00.00.00.00 01.00.00.08 00.00.00.09
 00.00.00.09 01.10.00.20 01.01.01.00 0c.08.80.00
 00.01.0f.a0 00.13.00.15 00.0c.01.00 00.00.00.00
 42.50.58.2d 56.53.49.31
```

[Table 304](#) describes the significant fields shown in the display.

Table 304 *debug vsi param-groups Command Field Descriptions*

| Field | Description |
|----------|---|
| Outgoing | The message is sent by the VSI master. |
| Incoming | The message is received by the VSI master. |
| bytes | Number of bytes in the message, starting at the VSI header, and excluding the link layer encapsulation. |
| 01.02... | The first 128 bytes of the message, in hexadecimal form. |

debug vtemplate

To display cloning information for a virtual access interface from the time it is cloned from a virtual template to the time the virtual access interface comes down when the call ends, use the **debug vtemplate** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtemplate

no debug vtemplate

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug vtemplate** command when a virtual access interface comes up. The virtual access interface is cloned from virtual template 1.

```
Router# debug vtemplate

VTEMPLATE Reuse vaccess8, New Recycle queue size:50

VTEMPLATE set default vaccess8 with no ip address

Virtual-Access8 VTEMPLATE hardware address 0000.0c09.ddfd
VTEMPLATE vaccess8 has a new cloneblk vtemplate, now it has vtemplate
VTEMPLATE undo default settings vaccess8

VTEMPLATE ***** CLONE VACCESS8 *****

VTEMPLATE Clone from vtemplate1 to vaccess8
interface Virtual-Access8
no ip address
encap ppp
ip unnumbered Ethernet0
no ip mroute-cache
fair-queue 64 256 0
no cdp enable
ppp authentication chap
end

%LINK-3-UPDOWN: Interface Virtual-Access8, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to up
```

The following is sample output from the **debug vtemplate** command when a virtual access interface goes down. The virtual interface is uncloned and returns to the recycle queue.

```
Router# debug vtemplate

%LINK-3-UPDOWN: Interface Virtual-Access8, changed state to down
VTEMPLATE Free vaccess8

%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to down
VTEMPLATE clean up dirty vaccess queue, size:1

VTEMPLATE Found a dirty vaccess8 clone with vtemplate
VTEMPLATE ***** UNCLONE VACCESS8 *****
VTEMPLATE Unclone to-be-freed vaccess8 command#7
interface Virtual-Access8
```

```

default ppp authentication chap
default cdp enable
default fair-queue 64 256 0
default ip mroute-cache
default ip unnumbered Ethernet0
default encaps ppp
default ip address
end

```

VTEMPLATE set default vaccess8 with no ip address

VTEMPLATE remove cloneblk vtemplate from vaccess8 with vtemplate

VTEMPLATE Add vaccess8 to recycle queue, size=51

Table 305 describes the significant fields shown in the display.

Table 305 *debug vtemplate Field Descriptions*

| Field | Description |
|--|---|
| VTEMPLATE Reuse vaccess8, New Recycle queue size:50 VTEMPLATE set default vaccess8 with no ip address | Virtual access interface 8 is reused; the current queue size is 50. |
| Virtual-Access8 VTEMPLATE hardware address 0000.0c09.ddfd | MAC address of virtual interface 8. |
| VTEMPLATE vaccess8 has a new cloneblk vtemplate, now it has vtemplate | Recording that virtual access interface 8 is cloned from the virtual interface template. |
| VTEMPLATE undo default settings vaccess8 | Removing the default settings. |
| VTEMPLATE ***** CLONE VACCESS8 ***** | Banner: Cloning is in progress on virtual access interface 8. |
| VTEMPLATE Clone from vtemplate1 to vaccess8 interface Virtual-Access8 no ip address encaps ppp ip unnumbered Ethernet0 no ip mroute-cache fair-queue 64 256 0 no cdp enable ppp authentication chap end | Specific configuration commands in virtual interface template 1 that are being applied to the virtual access interface 8. |
| %LINK-3-UPDOWN: Interface Virtual-Access8, changed state to up | Link status: The link is up. |
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to up | Line protocol status: The line protocol is up. |
| %LINK-3-UPDOWN: Interface Virtual-Access8, changed state to down | Link status: The link is down. |
| VTEMPLATE Free vaccess8 | Freeing virtual access interface 8. |

Table 305 debug vtemplate Field Descriptions (continued)

| Field | Description |
|---|--|
| %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to down | Line protocol status: The line protocol is down. |
| VTEMPLATE clean up dirty vaccess queue, size:1 VTEMPLATE Found a dirty vaccess8 clone with vtemplate VTEMPLATE ***** UNCLONE VACCESS8 ***** | Access queue cleanup is proceeding and the template is being uncloned. |
| VTEMPLATE Unclone to-be-freed vaccess8 command#7 interface Virtual-Access8 default ppp authentication chap default cdp enable default fair-queue 64 256 0 default ip mroute-cache default ip unnumbered Ethernet0 default encap ppp default ip address end | Specific configuration commands to be removed from the virtual access interface 8. |
| VTEMPLATE set default vaccess8 with no ip address | Default is set again. |
| VTEMPLATE remove cloneblk vtemplate from vaccess8 with vtemplate | Removing the record of cloning from a virtual interface template. |
| VTEMPLATE Add vaccess8 to recycle queue, size=51 | Virtual access interface is added to the recycle queue. |

debug vtemplate subinterface

To display debug message relating to virtual access subinterfaces, use the **debug vtemplate subinterface** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtemplate subinterface

no debug vtemplate subinterface

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|--|
| | 12.2(8)B | This command was introduced. |
| | 12.2(13)T | The command was integrated into Cisco IOS Release 12.2(13)T. |

Usage Guidelines These debug messages are displayed if the user configures virtual templates with commands that are incompatible with virtual access subinterfaces.

Examples The following example displays virtual access subinterface debug messages:

```
Router# debug vtemplate subinterface

Virtual Template subinterface debugging is on
Router#
Router#
Sep 19 15:09:41.989:VT[Vt11]:Config prevents subinterface creation
  carrier-delay 45
  ip rtp priority 2000 2010 500
```

[Table 306](#) describes the significant fields shown in the display.

Table 306 *debug vtemplate subinterface Field Descriptions*

| Field | Description |
|---------|---|
| VT | Indicates that this is a debug virtual template subinterface message. |
| [Vt11]: | Indicates that this message concerns virtual template 11. |

Table 306 debug vtemplate subinterface Field Descriptions (continued)

| Field | Description |
|---|---|
| Config prevents subinterface creation | Indicates that this virtual template cannot support the creation of virtual access subinterfaces. |
| carrier-delay 45 ip rtp priority 2000 2010 500 | These are the commands that make the virtual template incompatible with subinterfaces. |

Related Commands

| Command | Description |
|---|---|
| test virtual-template subinterface | Tests a virtual template to determine if it can support virtual access subinterfaces. |
| virtual-template subinterface | Enables the creation of virtual access subinterfaces. |

debug vtsp

To display the state of the gateway and the call events, use the **debug vtsp event** command in privileged EXEC configuration mode. To display the machine state during VTSP event processing, use the **no** form of the command.

```
debug vtsp {all | dsp | error | event | session | stats | tone | rtp}
```

```
no debug vtsp {all | dsp | error | event | session | stats | tone | rtp}
```



Note

The **debug vtsp** command with the **event** keyword must be turned on before the **voice call debug** command can be used.

Syntax Description

| | |
|----------------|--|
| all | All VTSP debugging except stats, tone, and event is enabled. |
| dsp | Digital signal processor (DSP) message trace is enabled. |
| error | VTSP error debugging is enabled. |
| event | State machine debugging is enabled. |
| session | Session debugging is enabled. |
| stats | Statistics debugging is enabled. |
| tone | Tone debugging is enabled. |
| rtp | Real-Time Transport Protocol (RTP) debugging is enabled. |

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|---|
| 12.0(3)T | This command was introduced on the Cisco AS5300 universal access servers. |
| 12.0(7)XK | This command was introduced on the Cisco 2600 series router, Cisco 3600 series router, and MC3810 multiservice access concentrators. |
| 12.1(2)T | This command was integrated into 12.1(2)T release. |
| 12.2(11)T | The enhancement of debug capabilities, which effects this command by adding a single call identification header, for Cisco voice gateways was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660 series; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Examples

The following is sample output for a Cisco AS5300 and Cisco 3640 when the **debug vtsp all** command is entered:

Cisco AS5300 Access Server

```
Router# debug vtsp all
!
Voice telephony call control all debugging is on
!
00:10:53: %SYS-5-CONFIG_I: Configured from console by console
00:10:54: %SYS-5-CONFIG_I: Configured from console by console
!
00:11:09:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_do_regxrule_translate:
00:11:09:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_do_regxrule_translate:
00:11:09:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_tsp_call_setup_ind:
00:11:09:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_tsp_fill_setup_ind:
```

Cisco 3640 Router

```
3640-orig# debug vtsp all
!
Voice telephony call control all debugging is on
!
3640-orig# show debug
Voice Telephony session debugging is on
Voice Telephony dsp debugging is on
Voice Telephony error debugging is on
!
20:58:16:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_tsp_apply_voiceport_xrule:
20:58:16: vtsp_tsp_apply_voiceport_xrule: vtsp_sdb 0x63797720; called_number 0x6294E0F0
called_oct3 128
20:58:16:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_tsp_apply_voiceport_xrule:
20:58:16: vtsp_tsp_apply_voiceport_xrule: No called number translation rule configured
20:58:16:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_do_regxrule_translate:
20:58:16:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_do_regxrule_translate:
calling_number(original)=
calling_number(xlated)=8880000 called_number(original)= called_number(xlated)=8881111
redirectNumber(original)= redirectNumber(xlated)=
20:58:16:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_tsp_call_setup_ind:
(sdb=0x63797720, tdm_info=0x0,
tsp_info=0x63825254, calling_number=8880000 calling_oct3 = 0x0, called_number=8881111
called_oct3 = 0x80, oct3a=0
3640-orig#x80): peer_tag=70
20:58:16:
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP():-1:-1:-1/vtsp_tsp_fill_setup_ind:
ev.clg.clir is 0
ev.clg.clid_transparent is 0
ev.clg.null_orig_clg is 0
ev.clg.calling_translated is false
// -1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/VTSP:(3/0:23):-1:0:0/vtsp_do_call_setup_ind:
Call ID=101123, guid=63EB9AC8
```

Table 307 describes the significant fields shown in the display.

Table 307 *debug vtsp all Field Descriptions*

| Field | Description |
|---------------------------------|--|
| VTSP():-1:-1:-1 | Identifies the VTSP module, port name, channel number, DSP slot, and DSP channel number. |
| vtsp_tsp_apply_voiceport_xrule: | Identifies a function name. |
| called_number | Identifies a called number. |
| called | Identifies the date the call was made. |
| peer_tag | Identifies the dial peer number. |
| guid | Identifies the GUID (hexadecimal address). |

Related Commands

| Command | Description |
|-------------------------|--|
| debug voip ccapi | Debugs the call control API. |
| voice call debug | Debugs a voice call by displaying a full GUID or header. |

debug vtsp all

To show debugging information for all **debug vtsp** commands, use the **debug vtsp all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp all

no debug vtsp all

Syntax Description This command has no arguments or keywords.

Defaults Debugging for voice telephony service provider (VTSP) is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(3)T | This command was introduced on the Cisco AS5300. |
| | 12.0(7)XK | This command was first supported on the Cisco 2600, Cisco 3600 and Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines The **debug vtsp all** command enables the following **debug vtsp** commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**. For more information or sample output, see the individual commands.

Execution of the **no debug vtsp all** command will turn off all VTSP-level debugging. You should turn off all debugging and then enter the debug commands you are interested in one by one. This process helps avoid confusion about which ports you are actually debugging.



Caution

Using this command can severely impact network performance and prevent any faxes from succeeding.

Examples

The following example shows the **debug vtsp all** command on a Cisco 3640 modular access router:

```
Router# debug vtsp all
```

```
Voice telephony call control all debugging is on
```

At this point, the VTSP is not aware of anything. The format of this message is

//callid/GUID/VTSP:(voice-port):T1-channel_number:DSP_number:DSP_channel_number:

- CallEntry ID is **-1**.
- GUID is **xxxxxxxxxx**.
- The voice port is blank.
- Channel ID is **-1**.
- DSP ID is **-1**.
- DSP channel ID is **-1**.

```
*Mar 1 08:23:10.869: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate:
```

The original and the translated calling number are the same (**55555**) and the original and the translated called number are the same (**888545**). These numbers are often the same because if a translation rule is applied, it will be on the dial peers or the ports, both of which comes later than these VTSP messages in the Cisco IOS code execution.

```
*Mar 1 08:23:10.869: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate:
calling_number(original)= calling_number(xlated)=55555 called_number(original)=
called_number(xlated)=888545 redirectNumber(original)= redirectNumber(xlated)=
```

The VTSP got a call setup indicator from the TSP layer with called number **888545** and calling number **55555**. There is no awareness of the CallEntry ID (**-1**) or the GUID (**xxxxxxxxxx**).

```
*Mar 1 08:23:10.873: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_tsp_call_setup_ind:
(sdb=0x634C90EC, tdm_info=0x0, tsp_info=0x63083950, calling_number=55555 calling_oct3 =
0x80, called_number=888545 called_oct3 = 0x80, oct3a=0x0): peer_tag=10002
*Mar 1 08:23:10.873: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_tsp_fill_setup_ind
: ev.clg.clir is 0
ev.clg.clid_transparent is 0
ev.clg.null_orig_clg is 0
ev.clg.calling_translated is false
*Mar 1 08:23:10.873: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_call_setup_ind: .
*Mar 1 08:23:10.873: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_allocate_cdb: ,cdb
0x635FC480
*Mar 1 08:23:10.873: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_call_setup_ind:
*Mar 1 08:23:10.873: source route label
```

At this point, the VTSP is not aware of anything. The format of this message is

//callid/GUID/VTSP:(voice-port):T1-channel_number:DSP_number:DSP_channel_number:

- CallEntry ID is **-1**.
- GUID is **D2F6429A8A8A**.
- The voice port is **1/0:23** where **23** indicates D channel.
- The T1 channel is still unknown at this point (**-1**).
- The digital signal processor (DSP) is **0**.
- The DSP channel is **4**.

```
*Mar 1 08:23:10.873: //-1/D2F6429A8A8A/VTSP:(1/0:23):-1:0:4/vtsp_do_call_setup_
ind: Call ID=101002, guid=635FCB08
```

The VTSP learns about the B channel (changed from **-1** to **22**), and the CallEntry ID is still unknown (**-1**).

```
*Mar 1 08:23:10.873: //-1/D2F6429A8A8A/VTSP:(1/0:23):22:0:4/vtsp_do_call_setup_ind:
type=0, under_spec=1615186336, name=, id0=23, id1=0, id2=0, calling=55555, called=888545
subscriber=RegularLinevtsp_do_call_setup_ind: redirect DN = reason = -1
*Mar 1 08:23:10.877: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_normal_call_setup_ind: .
```

The VTSP learns the CallEntry ID. The format of this message is

//callid/GUID/VTSP:(voice-port):T1-channel_number:DSP_number:DSP_channel_number:

- CallEntry ID is **899** (changed from -1 to 899)
- GUID is **D2F6429A8A8A**
- The voice port is **1/0:23** where **23** indicates D channel
- The T1 channel is **22**
- The DSP is **12**
- The DSP channel is **4**

```
*Mar 1 08:23:10.877: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_insert_cdb:,cdb
0x635FC480, CallID=899
*Mar 1 08:23:10.877:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_open_voice_and_set_params: .
```

In the following outputs VTSP sets some of the voice parameters for this call:

- Modem capability
- Playout delay
- Dial-peer tag **10003**
- Digit timeouts

```
*Mar 1 08:23:10.877: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_modem_proto_from_cdb:
cap_modem_proto 0
*Mar 1 08:23:10.881: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/set_playout_cdb:playout
default
*Mar 1 08:23:10.881:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_dsp_echo_canceller_control: echo_cancel: 1
*Mar 1 08:23:10.885: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_save_dialpeer_tag: tag
= 10003
*Mar 1 08:23:10.885: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_report_digit_control:
vtsp_report_digit_control: enable=0:
*Mar 1 08:23:10.885: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_report_digit_control:
digit reporting disabled
*Mar 1 08:23:10.885: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_get_digit_timeouts: :
vtsp_get_digit_timeouts
```

VTSP sends out a call-proceeding message to the POTS leg.

```
*Mar 1 08:23:10.885:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:vtsp: [1/0:23:899,
S_SETUP_INDICATED, E_CC_PROCEEDING]
*Mar 1 08:23:10.885: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_proceeding: .
*Mar 1 08:23:10.941: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_get_dialpeer_tag: tag
= 10003
```

```
*Mar 1 08:23:10.949: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_get_dialpeer_tag: tag
= 10003
```

VTSP sends out an alerting to the POTS leg; the phone is ringing at this time.

```
*Mar 1 08:23:10.949: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_PROCEEDING, E_CC_ALERT]
*Mar 1 08:23:10.949: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_alert: .
*Mar 1 08:23:10.949: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer_stop:3019095
*Mar 1 08:23:18.769: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_get_dialpeer_tag: tag
= 10003
```

The phone gets answered here, a bridge is now set up between the two call legs.

```
*Mar 1 08:23:18.769: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_ALERTING, E_CC_BRIDGE]
*Mar 1 08:23:18.769: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_bridge: .
```

The call is now connected.

```
*Mar 1 08:23:18.769: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_ALERTING, E_CC_CONNECT]
*Mar 1 08:23:18.769: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_alert_connect: .
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_ring_noan_timer_stop:
3019877
```

The VTSP received a capabilities indication event from the CCAPI. The VTSP needs to be aware of this because it handles the DSPs.

```
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CONNECT, E_CC_CAPS_IND]
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind: .
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind: RTP

PT:NTE[101],NTEtx[101],NSE[100],FaxInd[96],FaxAck[97],CiscoDTMF[121],FaxRelay[122],CASsig[
123],ClearChan[125],PCMu[0],PCMa[8]Codec[4],TxDynamicPayload[0],RxDynamicPayload[0]
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind: dtmf relay:
mode=32, codec=1
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind: passthrough:
cap_modem_proto 0, cap_modem_codec 0, cap_modem_redundancy 0, payload100, modem_relay 0,
gw-xid=0
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind: Encap 1, Vad
2, Codec 0x4, CodecBytes 20,
    FaxRate 2, FaxBytes 20, FaxNsf 0xAD0051
    SignalType 2
    DtmfRelay 32, Modem 0, SeqNumStart 0x1343
*Mar 1 08:23:18.773: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind:
*Mar 1 08:23:18.777: FORKING Parameters are forking mask: 0, simple_forking_codec_mask:
0, complex_forking_codec_mask 0
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ind: [
mode:0,init:60, min:40, max:200]
```

The VTSP received events regarding capabilities acknowledged from the call control API (CCAPI).

```
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CONNECT, E_CC_CAPS_ACK]
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ack: .
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ack: passthrough:
cap_modem_proto 0, cap_modem_codec 0, cap_modem_redundancy 0, payload100, modem_relay 0,
gw-xid=0
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_caps_ack: Named
Telephone Event payload: rcv 101, tx 101
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_switch_codec:
*Mar 1 08:23:18.777: DTMF Relay in act_switch_codec is 32
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/set_dsp_encap_config:
```

```

*Mar 1 08:23:18.777: set_dsp_encap_config: logical ssrc 40
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_modem_proto_from_cdb:
cap_modem_proto 0
*Mar 1 08:23:18.777: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_switch_codec: codec =
16
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer: 3019878
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, SP_PENDING_CODEC_SWITCH, E_DSPRM_PEND_SUCCESS]
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_pend_codec_success: .
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer_stop:3019878
*Mar 1 08:23:18.781:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_open_voice_and_set_params: .
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/set_dsp_encap_config:
*Mar 1 08:23:18.781: set_dsp_encap_config: logical ssrc 40
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_modem_proto_from_cdb:
cap_modem_proto 0
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/set_playout_cdb:playout
default
*Mar 1 08:23:18.781:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_dsp_echo_canceller_control: echo_cancel: 1
*Mar 1 08:23:18.781: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_add_fork:
*Mar 1 08:23:18.785: vtsp_add_fork
*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_update_fork_info:
*Mar 1 08:23:18.785: vtsp_update_fork_info: add_fork=0
*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_get_xmit_info_node:
*Mar 1 08:23:18.785: vtsp_get_xmit_info_node
*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_update_fork_info:
*Mar 1 08:23:18.785: vtsp_update_fork_info xmit func is 60FC43F0, context is
635BC51Cpeer_call_id: 900, stream_count: 1, update_flag 0
Router#
*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_update_fork_info:
*Mar 1 08:23:18.785: The stream bit-mask is 1
*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_update_fork_info:
*Mar 1 08:23:18.785: The stream type is 0

*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_update_fork_info:
*Mar 1 08:23:18.785: The logical ssrc is 64 for stream 0
*Mar 1 08:23:18.785: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_update_stream_count:
*Mar 1 08:23:18.785: g711_voice_count=0 g711_avt_count = 0
g711_voice_avt_count = 0 complex_voice_count = 1
complex_avt_count = 0 complex_voice_avt_count = 0

```

A digit begin event was detected while in the connect state. Digit 1 is dialed outbound on the POTS legs.

```

*Mar 1 08:23:26.745: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_call_digit_begin:
vtsp_call_digit_begin: digit=1, digit_begin_flags=0x0, rtp_timestamp=0, rtp_expiration=0
*Mar 1 08:23:26.745: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CONNECT, E_CC_DIGIT_BEGIN]
*Mar 1 08:23:26.745:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_digit_begin:act_digit_begin
*Mar 1 08:23:27.045: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_call_digit_end:
vtsp_call_digit_end: digit=1, duration=300

```

A digit end event was detected while in the connect state. The total duration of the digit was 300 ms.

```

*Mar 1 08:23:27.045: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CONNECT, E_CC_DIGIT_END,]
*Mar 1 08:23:27.045: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_digit_end:
act_digit_end

```

The call is hung up at this point, VTSP receives a bridge drop event from the CCAPI.

```

*Mar 1 08:23:39.393: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CONNECT, E_CC_BRIDGE_DROP]
*Mar 1 08:23:39.393: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_remove_stream_node:

```

```
*Mar 1 08:23:39.393: vtsp_remove_stream_node
*Mar 1 08:23:39.393: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_get_xmit_info_node:
*Mar 1 08:23:39.393: vtsp_get_xmit_info_node
*Mar 1 08:23:39.393: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_remove_stream_node:
*Mar 1 08:23:39.393: Stream count is 1 in function vtsp_remove_stream_node
*Mar 1 08:23:39.393: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_bdrops: .
*Mar 1 08:23:39.393: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_is_record_active:
*Mar 1 08:23:39.393: vtsp_is_record_active: false
```

VTSP gets a disconnect event from the CCAPI.

```
*Mar 1 08:23:39.397: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CONNECT, E_CC_DISCONNECT]
*Mar 1 08:23:39.397: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_disconnect: .
```

Following the disconnect event from the CCAPI, the timers are stopped.

```
*Mar 1 08:23:39.397: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_ring_noan_timer_stop:
3021940
*Mar 1 08:23:39.397:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_pcm_tone_detect_timer_stop: 3021940
*Mar 1 08:23:39.397:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_pcm_switchover_timer_stop: 3021940
*Mar 1 08:23:39.397: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_cm_detect_timer_stop:
3021940
*Mar 1 08:23:39.397:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_modem_relay_mode_timer_stop: 3021940
*Mar 1 08:23:39.397:
//899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_modem_relay_stats_timer_stop: 3021940
*Mar 1 08:23:39.397: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer_stop:3021940
*Mar 1 08:23:39.397: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_disconnect: cdb
0x635FC480, cause 0x10
*Mar 1 08:23:39.401: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer: 3021940
```

Statistics are collected for the DSP.

```
*Mar 1 08:23:39.405: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_WAIT_STATS, E_DSP_GET_ERROR]
*Mar 1 08:23:39.405: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_get_error: .
*Mar 1 08:23:39.405: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_print_error_stats:
rx_dropped=0 tx_dropped=0
*Mar 1 08:23:39.405: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_print_error_stats:
rx_control=55 tx_control=18 tx_control_dropped=0 dsp_mode_channel_1=0
dsp_mode_channel_2=0c[0]=0c[1]=2c[2]=6c[3]=87c[4]=83c[5]=84c[6]=106c[7]=78c[8]=0c[9]=32639
c[10]=32639c[11]=32639c[12]=32639c[13]=32639c[14]=32639c[15]=32639
*Mar 1 08:23:39.409: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer_stop:3021941
*Mar 1 08:23:39.409: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer: 3021941
*Mar 1 08:23:39.409: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_WAIT_STATS, E_DSP_GET_LEVELS]
*Mar 1 08:23:39.409: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_get_levels: .
*Mar 1 08:23:39.413: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_stats_complete: .
*Mar 1 08:23:39.413: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer_stop:3021941
*Mar 1 08:23:39.413: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_ring_noan_timer_stop:
3021941
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer: 3021942
```

The VTSP received a disconnect confirmation from the TSP layer.

```
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_WAIT_RELEASE, E_TSP_DISCONNECT_CONF]
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_wrelease_release: .
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_play_busy_timer_stop:
*Mar 1 08:23:39.417: vtsp_play_busy_timer_stop: 3021942
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_timer_stop:3021942
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_do_call_history: .
```

■ debug vtsp all

```
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_do_call_history:
*Mar 1 08:23:39.417: vtsp_do_call_history : src carrier id
*Mar 1 08:23:39.417: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_do_call_history:
*Mar 1 08:23:39.421: vtsp_do_call_history : tgt carrier id
*Mar 1 08:23:39.421: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_do_call_history:
CoderRate 16
```

DSP resource manager updates the state.

```
*Mar 1 08:23:39.421: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_process_event:
vtsp:[1/0:23:899, S_CLOSE_DSPRM, E_DSPRM_CLOSE_COMPLETE]
*Mar 1 08:23:39.421: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/act_terminate: .
*Mar 1 08:23:39.421: //899/D2F6429A8A8A/VTSP:(1/0:23):22:12:4/vtsp_free_cdb: ,cdb
0x635FC4803
```

Related Commands

| Command | Description |
|-------------------------|--|
| debug vtsp port | Limits VTSP debug output to a specific voice port. |
| show debug | Displays which debug commands are enabled. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vtsp dsp

To show messages from the digital signal processor (DSP) to the universal access server or router, use the **debug vtsp dsp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp dsp

no debug vtsp dsp

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for voice telephony service provider (VTSP) DSP is not enabled.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|---|
| 12.0(3)T | This command was introduced on the Cisco AS5300 series access servers. |
| 12.0(7)XK | This command was first supported on the Cisco 2600 series, Cisco 3600 series, and Cisco MC3810 multiservice access concentrators. |
| 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines

On Cisco AS5300 series access servers

The **debug vtsp dsp** command shows messages from the DSP on the voice feature card (VFC) to the router; this command can be useful if you suspect that the VFC is not functional. It is a simple way to check if the VFC is responding to off-hook indications.

On Cisco 2600, 3600, MC3810 series

The **debug vtsp dsp** command shows messages from the DSP to the router.



Note

We recommend that you log output from the **debug vtsp dsp** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows the VTSP DSP usage on a Cisco 3640 modular access router:

```
Router# debug vtsp dsp
```

```
Voice telephony call control dsp debugging is on
```

```
Router#
```

```
*Mar 1 01:05:18.539:
```

```
//12/A76D98838014/VTSP:(1/0:23):22:14:2/vtsp_dsp_echo_canceller_control: echo_cancel: 1
```

[Table 308](#) describes the significant fields shown in the display.

Table 308 *debug vtsp dsp Field Descriptions*

| Field | Descriptions |
|----------------|--|
| //12 | CallEntry ID. |
| /A76D98838014 | GUID. |
| 1/0:23 | Controller 1/0, D channel. |
| :22 | B-channel number. This can also be found using the show voice call summary command. |
| :14 | DSP number. This can also be found using the show voice dsp command. |
| :2 | Channel number on the DSP. This can also be found using the show voice dsp command. |
| echo_cancel: 1 | Echo cancel is on. |

Related Commands

| Command | Description |
|-------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| debug vtsp port | Limits VTSP debug output to a specific voice port. |
| show debug | Displays which debug commands are enabled. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vtsp error

To display processing errors in the voice telephony service provider (VTSP), use the **debug vtsp error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp error

no debug vtsp error

Syntax Description This command has no arguments or keywords.

Defaults Debugging for VTSP errors is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(7)XK | This command was first supported on the Cisco 2600, Cisco 3600 and Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines The **debug vtsp error** command can be used to check for mismatches in interface capabilities.



Note

We recommend that you log output from the **debug vtsp error** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

| Related Commands | Command | Description |
|------------------|-------------------------|--|
| | debug vpm all | Enables all VPM debugging. |
| | debug vtsp port | Limits VTSP debug output to a specific voice port. |
| | show debug | Displays which debug commands are enabled. |
| | voice call debug | Allows configuration of the voice call debug output. |

debug vtsp event

To display the state of the gateway and the call events, use the **debug vtsp event** command in privileged EXEC mode. To display the machine state during voice telephony service provider (VTSP) event processing, use the **no** form of this command.

debug vtsp event

no debug vtsp event

Syntax Description This command has no arguments or keywords.

Defaults Debugging for VTSP events is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(3)T | This command was introduced on the Cisco AS5300 universal access servers. |
| | 12.0(7)XK | This command was first supported on the Cisco 2600, Cisco 3600 and Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines The **debug vtsp event** command can be used to enable state machine debugging.



Note

We recommend that you log output from the **debug vtsp event** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows sample output from the **debug vtsp event** command:

```
Router# debug vtsp event

Voice Telephony event debugging is on
```

The following events are seen when the call is set up.

```
*Mar 1 22:20:39.138: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_SETUP_INDICATED, event: E_CC_PROCEEDING]
```

When the phone starts ringing, the **ALERT** event appears.

```
*Mar 1 22:20:39.202: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_PROCEEDING, event: E_CC_ALERT]
Router#
```

As soon as the call is answered, the bridge comes up and the **CONNECT** event appears.

```
*Mar 1 22:20:47.798: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_ALERTING, event: E_CC_BRIDGE]
*Mar 1 22:20:47.802: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_ALERTING, event: E_CC_CONNECT]
```

The capabilities are exchanged as soon as the connection occurs.

```
*Mar 1 22:20:47.802: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_CAPS_IND]
*Mar 1 22:20:47.802: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_CAPS_ACK]
*Mar 1 22:20:47.802: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:SP_PENDING_CODEEC_SWITCH, event: E_DSPRM_PEND_SUCCESS]
```

The following debug outputs are regularly seen as the call progresses. The outputs indicate that collection of Tx/Rx/Delay/Error statistics is occurring.

```
*Mar 1 22:20:49.470: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_REQ_PACK_STAT]
*Mar 1 22:20:49.482: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_TX]
*Mar 1 22:20:49.482: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_RX]
*Mar 1 22:20:49.486: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_VP_DELAY]
*Mar 1 22:20:49.486: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_VP_ERROR]
*Mar 1 22:20:51.638: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_REQ_PACK_STAT]
*Mar 1 22:20:51.638: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_TX]
*Mar 1 22:20:51.638: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_RX]
*Mar 1 22:20:51.642: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_VP_DELAY]
*Mar 1 22:20:51.642: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_DSP_GET_VP_ERROR]
Router#
```

When digits are passed during the conversation, the digit begin and digit end events are seen.

```
*Mar 1 22:21:01.542: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_DIGIT_BEGIN]
*Mar 1 22:21:01.842: //72/D14258FE806E/VTSP:(1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_DIGIT_END,]
```

```
*Mar 1 22:21:01.962: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_DIGIT_BEGIN]
*Mar 1 22:21:02.262: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_DIGIT_END,]
Router#
```

Once the call is hung up from one side, the **bridge_drop** and the **disconnect** events appear.

```
*Mar 1 22:21:10.834: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_TSP_DISCONNECT_IND]
*Mar 1 22:21:10.838: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_BRIDGE_DROP]
*Mar 1 22:21:10.838: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_CONNECT, event: E_CC_DISCONNECT]
```

Following the disconnect event, the signaling state becomes **S_WAIT_STATS**, during which the DSP stats are collected.

```
*Mar 1 22:21:10.842: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_WAIT_STATS, event: E_DSP_GET_ERROR]
*Mar 1 22:21:10.846: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_WAIT_STATS, event: E_DSP_GET_LEVELS]
*Mar 1 22:21:10.854: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_WAIT_STATS, event: E_DSP_GET_TX]
```

The conference is torn down and the DSP is released.

```
*Mar 1 22:21:10.854: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_WAIT_RELEASE, event: E_TSP_DISCONNECT_CONF]
*Mar 1 22:21:10.858: //72/D14258FE806E/VTSP: (1/0:23):22:14:2/vtsp_process_event:
[state:S_CLOSE_DSPRM, event: E_DSPRM_CLOSE_COMPLETE]
```

Related Commands

| Command | Description |
|-------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| debug vtsp error | Displays processing errors in the VTSP. |
| debug vtsp port | Limits VTSP debug output to a specific voice port. |
| show debug | Displays which debug commands are enabled. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vtsp port

To observe the behavior of the voice telephony service provider (VTSP) state machine on a specific voice port, use the **debug vtsp port** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

For Cisco 2600 and Cisco 3600 series with analog voice ports

debug vtsp port *slot/subunit/port*

no debug vtsp port *slot/subunit/port*

For Cisco 2600 and Cisco 3600 series with digital voice ports (with T1 packet voice trunk network modules)

debug vtsp port *slot/port:ds0-group*

no debug vtsp port *slot/port:ds0-group*

For Cisco MC3810 series with analog voice ports

debug vtsp port *slot/port*

no debug vtsp port *slot/port*

For Cisco MC3810 series with digital voice ports

debug vtsp port *slot/port*

no debug vtsp port *slot/ds0-group*

Syntax Description

| | |
|--------------------------|---|
| <i>slot/subunit/port</i> | <ul style="list-style-type: none"> <i>slot</i> specifies a router slot in which a voice network module (NM) is installed. Valid entries are router slot numbers for the specific platform. <i>subunit</i> specifies a voice interface card (VIC) where the voice port is located. Valid entries are 0 and 1. (The VIC fits into the voice network module.) <i>port</i> specifies an analog voice port number. Valid entries are 0 and 1. |
|--------------------------|---|

For the Cisco 2600 and Cisco 3600 series with digital voice ports

| | |
|----------------------------|--|
| <i>slot/port:ds0-group</i> | <p>Debugs the digital voice port you specify with the <i>slot/port:ds0-group</i> designation.</p> <ul style="list-style-type: none"> <i>slot</i> specifies a router slot in which the packet voice trunk network module (NM) is installed. Valid entries are router slot numbers for the specific platform. <i>port</i> specifies a T1 or E1 physical port in the voice WAN interface card (VWIC). Valid entries are 0 and 1. (One VWIC fits in an NM.) <i>ds0-group</i> specifies a T1 or E1 logical port number. Valid entries are 0 to 23 for T1 and 0 to 30 for E1. |
|----------------------------|--|

For the Cisco MC3810 series with analog voice ports

| | |
|------------------|---|
| <i>slot/port</i> | <p>Debugs the analog voice port you specify with the <i>slot/port</i> designation.</p> <ul style="list-style-type: none"> <i>slot</i> is the physical slot in which the analog voice module (AVM) is installed. The <i>slot</i> is always 1 for analog voice ports in the Cisco MC3810 series. <i>port</i> specifies an analog voice port number. Valid entries are 1 to 6. |
|------------------|---|

For the Cisco MC3810 series with digital voice ports

| | |
|-----------------------|---|
| <i>slot:ds0-group</i> | <p>Debugs the digital voice port you specify with the <i>slot:ds0-group</i> designation.</p> <ul style="list-style-type: none"> <i>slot</i> specifies the module (and controller). Valid entries are 0 for the MFT (controller 0) and 1 for the DVM (controller 1). <i>ds0-group</i> specifies a T1 or E1 logical voice port number. Valid entries are 0 to 23 for T1 and 0 to 30 for E1. |
|-----------------------|---|

Defaults

Debug VTSP commands are not limited to a specific port.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|---|
| 12.0(3)XG | This command was introduced on Cisco 2600 and Cisco 3600 series routers. |
| 12.0(3)T | This command was introduced on the Cisco AS5300 series access servers. |
| 12.0(7)XK | This command was first supported on the Cisco MC3810 series. |
| 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines

Use the **debug vtsp port** command to limit the debug output to a specific voice port. The debug output can be quite voluminous for a single channel. The entire VTSP debug output from a platform with 12 voice ports might create problems. Use this debug with any or all of the other debug modes.

Execution of **no debug vtsp all** will turn off all VTSP-level debugging. It is usually a good idea to turn off all debugging and then enter the debug commands you are interested in one by one. This will help to avoid confusion about which ports you are actually debugging.

**Note**

We recommend that you log output from the **debug vtsp port** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Related Commands

| Command | Description |
|-------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| show debug | Displays which debug commands are enabled. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vtsp rtp

To show the voice telephony service provider (VTSP) Real-Time Protocol (RTP) packet debugging, use the **debug vtsp rtp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug vtsp rtp { both | from-dsp | to-dsp } { payload payload-type codec }
```

```
no debug vtsp rtp
```

| Syntax Description | |
|---------------------|---|
| both | Displays packets that are both sent and received from the digital signal processor (DSP). |
| from-dsp | Displays packets received from the DSP. |
| to-dsp | Displays packets sent to the DSP. |
| payload | (Optional) Specifies a specific type of payload. |
| <i>payload-type</i> | (Optional) Valid payload types are as follows: <ul style="list-style-type: none"> • all—All packets are displayed. No codec is specified. • equal-to—Packets in payloads equal to the specified codec are displayed. • greater-than—Packets in payloads greater than the specified codec are displayed. • less-than—Packets in payloads less than the specified codec are displayed. • other-than—Packets in payloads other than the specified codec are displayed. • other-than-fax-and—Packets in payloads other than fax relay and the specified codec are displayed. • other-than-silence-and—Packets in payloads other than silence and the specified codec are displayed. |
| <i>codec</i> | (Optional) If a codec needs to be specified for the payload type, valid codecs are as follows: <ul style="list-style-type: none"> • 0 to 123—Custom value of the payload. • g711alaw—G.711 alaw 64000 bps. • g711ulaw—G.711 ulaw 64000 bps. • g723.1—G.723.1. • g726—G.726. • g728—G.728. • g729a—G.729a. |

Defaults No default behavior or values

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(3)T | This command was introduced on the Cisco AS5300 series access servers. |
| | 12.0(7)XK | This command was first supported on the Cisco 2600, Cisco 3600, and MC3810 series devices. |
| | 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines

We recommend that you log output from the **debug vtsp rtp** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows the VTSP RTP debugging:

```
Router# debug vtsp rtp both pay all
```

```
Voice telephony RTP Packet debugging enabled for payloads of all types of packets from and to DSP
```

The following line shows the payload from the DSP (telephony leg) to the IP leg:

```
*Mar 1 01:10:05.687: //20/4DD959B48020/VTSP:(1/0:23):22:14:2/vtsp_print_rtp_header: s=DSP d=VoIP payload 0x12 ssrc 0x40 sequence 0x19E3 timestamp 0xCCDCE092
```

The following line shows the payload from the IP leg to the DSP (telephony leg):

```
*Mar 1 01:10:05.699: //20/4DD959B48020/VTSP:(1/0:23):22:14:2/vtsp_print_rtp_header: s=VoIP d=DSP payload 0x12 ssrc 0xAF0534E3 sequence 0x92A timestamp 0x6BE50
```

Related Commands

| Command | Description |
|-------------------------|--|
| debug vtsp dsp | Shows messages from the DSP. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vtsp send-nse

To trigger the voice telephony service provider (VTSP) software module to send a triple redundant network services engine (NSE), use the **debug vtsp send-nse** command in privileged EXEC mode. To disable this action, use the **no** form of this command.

debug vtsp send-nse

no debug vtsp send-nse

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(7)XK | This command was introduced on the Cisco MC3810 and the Cisco 3600 series routers (except the Cisco 3620) in a release that was not generally available. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines We recommend that you log output from the **debug vtsp send-nse** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

| Related Commands | Command | Description |
|------------------|------------------------------|--|
| | debug rtpspi all | Debugs all RTP SPI errors, sessions, and in/out functions. |
| | debug rtpspi errors | Debugs RTP SPI errors. |
| | debug rtpspi inout | Debugs RTP SPI in/out functions. |
| | debug rtpspi send-nse | Triggers the RTP SPI to send a triple redundant NSE. |
| | debug sgcp errors | Debugs SGCP errors. |
| | debug sgcp events | Debugs SGCP events. |
| | debug sgcp packet | Debugs SGCP packets. |
| | voice call debug | Allows configuration of the voice call debug output. |

debug vtsp session

To trace how the router interacts with the digital signal processor (DSP) based on the signaling indications from the signaling stack and requests from the application, use the **debug vtsp session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp session

no debug vtsp session

Syntax Description This command has no arguments or keywords.

Defaults Debugging for the VTSP session is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(3)T | This command was introduced on the Cisco AS5300 universal access servers. |
| | 12.0(7)XK | This command was first supported on the Cisco 2600, Cisco 3600 and Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines The **debug vtsp session** command traces how the router interacts with the DSP based on the signaling indications from the signaling stack and requests from the application. This debug command displays information about how each network indication and application request is handled, signaling indications, and DSP control messages.

This debug level shows the internal workings of the voice telephony call state machine.



Note

We recommend that you log output from the **debug vtsp send-nse** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows sample output from the **debug vtsp session** command:

```
Router# debug vtsp session
```

```
Voice telephony call control session debugging is on
```

At this point, the VTSP is not aware of anything. The format of this message is

//callid/GUID/VTSP:(voice-port):T1-channel_number:DSP_number:DSP_channel_number:

- CallEntry ID is **-1**.
- GUID is **xxxxxxxxxx**.
- The voice port is blank.
- Channel ID is **-1**.
- DSP ID is **-1**.
- DSP channel ID is **-1**.

```
*Mar 2 01:20:43.225: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate: .
```

The original and the translated calling number are the same (**55555**) and the original and the translated called number are the same (**888545**). These numbers are often the same because if a translation rule is applied, it will be on the dial peers or the ports both of which comes later than these VTSP messages in the Cisco IOS code execution.

```
*Mar 2 01:20:43.225: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate:
calling_number(original)= calling_number(xlated)=55555 called_number(original)=
called_number(xlated)=888545 redirectNumber(original)= redirectNumber(xlated)=
```

The VTSP got a call setup indicator from the TSP layer with called number **888545** and calling number **55555**. There is no awareness of the CallEntry ID (**-1**) or the GUID (**xxxxxxxxxxxx**).

```
*Mar 2 01:20:43.225: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_tsp_call_setup_ind:
(sdb=0x637AA6C0, tdm_info=0x0, tsp_info=0x630B6050, calling_number=55555 calling_oct3 =
0x80, called_number=888545 called_oct3 = 0x80, oct3a=0x0): peer_tag=10002
```

```
*Mar 2 01:20:43.225: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_tsp_fill_setup_ind:
ev.clg.clir is 0
ev.clg.clid_transparent is 0
ev.clg.null_orig_clg is 0
ev.clg.calling_translated is false
```

```
*Mar 2 01:20:43.229: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_call_setup_ind: .
*Mar 2 01:20:43.229: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_allocate_cdb: ,cdb
0x637B2A68
*Mar 2 01:20:43.229: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_call_setup_ind:
*Mar 2 01:20:43.229: source route label
```

At this point, the VTSP is not aware of the anything. The format of this message is

//callid/GUID/VTSP:(voice-port):T1-channel_number:DSP_number:DSP_channel_number:

- CallEntry ID is **-1**.
- GUID is **F90073EB8080**.
- The voice port is **1/0:23** where **23** indicates D channel.
- The T1 channel is still unknown at this point (**-1**).
- The DSP is **0**.
- The DSP channel is **2**.

```
*Mar 2 01:20:43.229: //-1/F90073EB8080/VTSP:(1/0:23):-1:0:2/vtsp_do_call_setup_ind: Call
ID=98432, guid=637B43F4
```

The VTSP learns that the B channel used changed from **-1** to **22**.

```
*Mar 2 01:20:43.229: //-1/F90073EB8080/VTSP:(1/0:23):22:0:2/vtsp_do_call_setup_ind:
type=0, under_spec=1615186336, name=, id0=23, id1=0, id2=0, calling=55555, called=888545
subscriber=RegularLinevtsp_do_call_setup_ind: redirect DN = reason = -1
*Mar 2 01:20:43.229: //-1/xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_normal_call_setup_ind: .
```

The VTSP learns the CallEntry ID. The format of this message is

//callid/GUID/VTSP:(voice-port):T1-channel_number:DSP_number:DSP_channel_number:

- CallEntry ID is **84** (changed from -1 to 84).
- GUID is **F90073EB8080**.
- The voice port is **1/0:23** where **23** indicates D channel.
- The T1 channel is **22**.
- The DSP is **14**.
- The DSP channel is **2**.

```
*Mar 2 01:20:43.233: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_insert_cdb: ,cdb
0x637B2A68, CallID=84
*Mar 2 01:20:43.233:
//84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_open_voice_and_set_params: .
```

In the following outputs VTSP sets some of the voice parameters for this call:

- Modem capability
- Playout-delay
- Dial-peer tag = **10003**
- Digit-timeouts

```
*Mar 2 01:20:43.233: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_modem_proto_from_cdb:
cap_modem_proto 0
*Mar 2 01:20:43.233: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/set_playout_cdb: playout
default

*Mar 2 01:20:43.237: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_save_dialpeer_tag: tag
= 10003
*Mar 2 01:20:43.237: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_report_digit_control:
vtsp_report_digit_control: enable=0:
*Mar 2 01:20:43.237: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_report_digit_control:
digit reporting disabled
*Mar 2 01:20:43.237: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_get_digit_timeouts: :
vtsp_get_digit_timeouts
```

The VTSP sends out a call-proceeding message to the POTS leg.

```
*Mar 2 01:20:43.241: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_process_event:
vtsp:[1/0:23:84, S_SETUP_INDICATED, E_CC_PROCEEDING]
*Mar 2 01:20:43.241: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/act_proceeding: .
Router#
*Mar 2 01:20:43.297: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_get_dialpeer_tag: tag =
10003
*Mar 2 01:20:43.301: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_get_dialpeer_tag: tag =
10003
```

VTSP sends out an alerting to the POTS leg; the phone is ringing now.

```
*Mar 2 01:20:43.301: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_process_event:
vtsp:[1/0:23:84, S_PROCEEDING, E_CC_ALERT]
*Mar 2 01:20:43.301: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/act_alert: .
*Mar 2 01:20:43.301: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_timer_stop: 9124331
Router#
*Mar 2 01:20:52.289: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_get_dialpeer_tag: tag =
10003
```

The phone gets answered here, and a bridge is now set up between the two call legs.

```
*Mar 2 01:20:52.289: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_process_event:
vtsp:[1/0:23:84, S_ALERTING, E_CC_BRIDGE]
*Mar 2 01:20:52.289: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/act_bridge: .
```

The call is now connected.

```
*Mar 2 01:20:52.289: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_process_event:
vtsp:[1/0:23:84, S_ALERTING, E_CC_CONNECT]
*Mar 2 01:20:52.289: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/act_alert_connect: .
*Mar 2 01:20:52.289: //84/F90073EB8080/VTSP:(1/0:23):22:14:2/vtsp_ring_noan_timer_stop:
9125229
```

Related Commands

| Command | Description |
|------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| debug vtsp port | Limits VTSP debug output to a specific voice port. |
| show debug | Displays which debug commands are enabled. |

debug vtsp stats

To debug periodic statistical-information-request messages sent and received from the DSP during a call, use the **debug vtsp stats** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp stats

no debug vtsp stats

Syntax Description This command has no arguments or keywords.

Defaults Debugging for VTSP statistics is not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.0(3)T | This command was introduced on the Cisco AS5300 universal access servers. |
| | 12.0(7)XK | This command was first supported on the Cisco 2600, Cisco 3600 and Cisco MC3810 series. |
| | 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines The **debug vtsp stats** command generates a collection of DSP statistics for generating Real-Time Transport Protocol (RTCP) packets and a collection of other statistical information.



Note

We recommend that you log output from the **debug vtsp stats** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Related Commands

| Command | Description |
|-------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| debug vtsp port | Limits VTSP debug output to a specific voice port. |
| show debug | Displays which debug commands are enabled. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vtsp tone

To display debug messages showing the types of tones generated by the Voice over IP (VoIP) gateway, use the **debug vtsp tone** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp tone

no debug vtsp tone

Syntax Description This command has no keywords or arguments.

Defaults Tone generation messages are not enabled.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|---|
| | 12.1(3)XI | This command was introduced. |
| | 12.1(5)T | This command was integrated into Cisco IOS Release 12.1(5)T. |
| | 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines We recommend that you log output from the **debug vtsp tone** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

| Related Commands | Command | Description |
|------------------|---------------------------|--|
| | debug vtsp dsp | Shows messages from the DSP on the modem to the router. |
| | debug vtsp session | Traces how the router interacts with the DSP, based on the signaling indications from the signaling stack and requests from the application. |
| | voice call debug | Allows configuration of the voice call debug output. |

debug vtsp vofr subframe

To display the first 10 bytes (including header) of selected Voice over Frame Relay (VoFR) subframes for the interface, use the **debug vtsp vofr subframe** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vtsp vofr subframe *payload* [**from-dsp**] [**to-dsp**]

no debug vtsp vofr subframe

Syntax Description

payload Number used to selectively display subframes of a specific payload. Payload types are:

- 0:** Primary Payload
- 1:** Annex-A
- 2:** Annex-B
- 3:** Annex-D
- 4:** All other payloads
- 5:** All payloads



Caution Options 0 and 5 can cause network instability.

from-dsp Displays only the subframes received from the digital signal processor (DSP).

to-dsp Displays only the subframes going to the DSP.

Defaults

Debugging for voice telephony service provider (VTSP) VoFR subframe is not enabled.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|---------------------|---|
| 12.0(3)XG, 12.0(4)T | This command was introduced on the Cisco 2600 and Cisco 3600 series. |
| 12.0(4)T | This command was integrated into Cisco IOS Release 12.0(4)T. |
| 12.0(7)XK | This command was first supported on the Cisco MC3810 series. |
| 12.1(2)T | This command was integrated into Cisco IOS Release 12.1(2)T. |
| 12.2(11)T | The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300, and Cisco AS5800; and, on the Cisco MC3810 multiservice access concentrators. |

Usage Guidelines

Each debug output displays the first 10 bytes of the FRF.11 subframe, including header bytes. The **from-dsp** and **to-dsp** options can be used to limit the debugs to a single direction. If not specified, debugs are displayed for subframes when they are received from the DSP and before they are sent to the DSP.

Use extreme caution in selecting payload options 0 and 6. These options may cause network instability.

**Note**

We recommend that you log output from the **debug vtsp vofr subframe** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Related Commands

| Command | Description |
|-------------------------|--|
| debug vpm all | Enables all VPM debugging. |
| debug vtsp port | Limits VTSP debug output to a specific voice port. |
| show debug | Displays which debug commands are enabled. |
| voice call debug | Allows configuration of the voice call debug output. |

debug vxml

To display debugging messages for VoiceXML features, use the **debug vxml** command in privileged EXEC mode. To disable VoiceXML debugging output, use the **no** form of this command.

debug vxml [**all** | **application** | **background** | **error** | **event** | **grammar** | **puts** | **ssml** | **trace** | **warning**]

no debug vxml [**all** | **application** | **background** | **error** | **event** | **grammar** | **puts** | **ssml** | **trace** | **warning**]

| Syntax Description | | |
|--------------------|--------------------|---|
| | all | (Optional) Displays all VoiceXML debugging messages. |
| | application | (Optional) Displays VoiceXML application states information. |
| | background | (Optional) Displays VoiceXML background messages. |
| | error | (Optional) Displays VoiceXML application error messages. |
| | event | (Optional) Displays VoiceXML asynchronous events. |
| | grammar | (Optional) Enables syntax checking of XML grammar by the VoiceXML interpreter and displays syntax debugging messages. |
| | puts | (Optional) Displays the results of VoiceXML <cisco-puts> and <cisco-putvar> tags. |
| | ssml | (Optional) Enables syntax checking of Speech Synthesis Markup Language (SSML) by the VoiceXML interpreter and displays syntax debugging messages. |
| | trace | (Optional) Displays a trace of all activities for the current VoiceXML document. |
| | warning | (Optional) Displays VoiceXML warning messages. |

Defaults No default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|--|
| | 12.2(2)XB | This command was introduced on the Cisco AS5300, Cisco AS5350, and Cisco AS5400. |
| | 12.2(11)T | This command was implemented on the Cisco 3640 and Cisco 3660, and the background , grammar , and ssml keywords were added. |

Usage Guidelines

- The output of this command is affected by the **debug condition application voice** command. If the **debug condition application voice** command is configured and the <cisco-debug> element is enabled in the VoiceXML document, debugging output is limited to the VoiceXML application named in the **debug condition application voice** command.
- The **debug vxml** command enables all VoiceXML debugging messages except those displayed by the **grammar** and **ssml** keywords. The **debug vxml all** command enables all VoiceXML debugging messages including grammar and SSML.



Caution

When the **debug vxml grammar** or **debug vxml ssml** command is enabled, the VoiceXML document could abort if there is a fatal syntax error in its eXtensible Markup Language (XML) grammar or SSML.

Examples

The following example shows output from the **debug vxml application** command:

```
Router# debug vxml application

vxml application debugging is on
Router#
lw5d: //-1//VAPP:/vapp_get_apphandler:
lw5d: vapp_get_apphandler: Script callme
lw5d: //-1//VAPP:/vapp_get_apphandler_core:
lw5d: //-1/000000000000/VAPP:/vapp_InterpInitConfigParams:
lw5d: //-1/000000000000/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event CC_D
lw5d: //-1/000000000000/VAPP:/vapp_driver: pInterp[660E10FC]:
lw5d: //-1/000000000000/VAPP:/vapp_driver: evtID: 28 vapp record state: 0
lw5d: //-1/000000000000/VAPP:/vapp_evt_setup:
lw5d: //-1//VAPP:/vapp_incoming_cal
doc-rtr54-01#lblock:
lw5d: vapp_incoming_callblock:
lw5d: //39/924083218026/VAPP:/vapp_load_or_run_script:
lw5d: //39/924083218026/VAPP:/vapp_load_or_run_script:
lw5d: The VXML Script with len=1450 starts:
-----
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">

<property name="fetchtimeout" value="20s"/>
<var name="phone_num"/>
    <form id="main">
        <noinput>
            <prompt>
                <audio src="flas
lw5d: //39/924083218026/VAPP:/vapp_media_play:
lw5d: //39/
Router#924083218026/VAPP:/vapp_media_play: prompt=flash:welcme_test.au:
lw5d: //39/924083218026/VAPP:/vapp_checksessionstate:
lw5d: //39/924083218026/VAPP:/vapp_checkifdone:
lw5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event CC_E
lw5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
lw5d: //39/924083218026/VAPP:/vapp_driver: evtID: 36 vapp record state: 0
lw5d: //39/924083218026/VAPP:/vapp_checksessionstate:
lw5d: //39/924083218026/VAPP:/vapp_checkifdo
Router#ne:
lw5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event MSWR
lw5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
lw5d: //39/924083218026/VAPP:/vapp_driver: evtID: 77 vapp record state: 0
lw5d: //39/924083218026/VAPP:/vapp_media_done: evID=77 status=0, protocol=0, st0
```

```

1w5d: //39/924083218026/VAPP:/vapp_media_play:
1w5d: //39/924083218026/VAPP:/vapp_media_play: prompt=flash:enter_dest.au:
1w5d: //39/924083218026/VAPP:/vapp_c
Router#hecksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
Router#
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event MSWR
1w5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 77 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_media_done: evID=77 status=0, protocol=0, st0
1w5d: //39/924083218026/VAPP:/vapp_digit_collect:
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
Router#
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event APPE
1w5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 87 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_digit_collection_done:
1w5d: //39/924083218026/VAPP:/vapp_digit_collection_done: digits [5551234], sta]
1w5d: //39/924083218026/VAPP:/vapp_gain_control_default:
1w5d: //39/924083218026/VAPP:/vapp_placecall:

Router#1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
Router#
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event APPE
1w5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 84 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_evt_setupdone:
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
Router#
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event CC_D
1w5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 15 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_call_disconnected:
1w5d: //39/924083218026/VAPP:/vapp_connection_destroy:
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: Sta
Router#te VAPP_ACTIVE got event CC_EV_CONF_DESTROY_DONE
1w5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]:
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 34 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_leg_disconnect:
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event CC_E
1w5d: //39/924083218026/VAPP:/vapp_driver: pInterp[660E10FC]
Router#
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 16 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_terminate:
1w5d: //39/924083218026/VAPP:/vapp_session_exit_event_name: Exit Event vxml.sese
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_terminate_initiation:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_CLEANING got event CE
1w5d: //39/924083218
Router#026/VAPP:/vapp_cleaner:
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
1w5d: //39/924083218026/VAPP:/vapp_evt_handler: State VAPP_CLEANING got event AE
1w5d: //39/924083218026/VAPP:/vapp_cleaner:
1w5d: //39/924083218026/VAPP:/vapp_cleaner: VxmlDialogDone event=vxml.session.c0
1w5d: //39/924083218026/VAPP:/vapp_popifdone:

```

```

1w5d: //39/924083218026/VAPP:/vapp_checkifdone:
1w5d: //39/924083218026/VAPP:/vapp_
Router#cleanup_apphandler:
1w5d: vapp_cleanup_apphandler: Terminate FALSE Terminated TRUE{HAN[VXML_HAN] [NU}
1w5d: //39/924083218026/VAPP:/vapp_free_apphandler: {HAN[VXML_HAN] [NULL  ] }

```

The following example shows output from the **debug vxml background** command:

```

Router# debug vxml background

vxml background messages debugging is on
Router#
1w5d: //-1//VAPP:/vapp_init_apphandler:
1w5d: //-1//VXML:/vxml_create: url=flash:call.vxml vapphandle=660E10FC
Router#
1w5d: //-1//VAPP:/vapp_process: Interp Done

```

The following examples show output from the **debug vxml error** command:

```
Router# debug vxml error
```

This example output shows an error when the version header is missing:

```

*May 10 20:08:57.572://7/98119BD78008/VXML:/vxml_vxml_build:tftp://demo/scripts/test.vxml
at line 2:<vxml version> required attribute missing
*May 10 20:08:57.576://7/98119BD78008/VXML:/vxml_create:
*May 10 20:08:57.576:code=ERROR vapp=VAPP_SUCCESS vxml=VXML_ERROR_INVALID

```

This example output shows an error when a field item is not used according to the DTD:

```

*May 10
20:16:23.315://8/A1BCF458800B/VXML:/vxml_start_element_handler:tftp://demo/scripts/test.vx
ml at line 4:Element <field> is not used according to DTD
*May 10 20:16:23.315://8/A1BCF458800B/VXML:/vxml_create:
*May 10 20:16:23.315:code=ERROR vapp=VAPP_SUCCESS vxml=VXML_ERROR_INVALID

```

This example output shows an error when there is a tag mismatch:

```

*May 10 20:17:44.485://10/D21DEAB58011/VXML:/vxml_parse:tftp://demo/scripts/test.vxml at
line 48:mismatched tag
*May 10 20:17:44.485://10/D21DEAB58011/VXML:/vxml_create:
*May 10 20:17:44.485:code=ERROR vapp=VAPP_SUCCESS vxml=VXML_ERROR_INVALID

```

The following example shows output from the **debug vxml event** command:

```

Router# debug vxml event

vxml events debugging is on
Router#
1w5d: //47/000000000000/VXML:/vxml_media_done: status 0 async_status 100000000
Router#
1w5d: //47/000000000000/VXML:/vxml_media_done: status 0 async_status 300000000
Router#
1w5d: //47/000000000000/VXML:/vxml_digit_collection_done: vxmlp 6534C7C8 status0
1w5d: //47/000000000000/VXML:/vxml_digit_collection_done: digits 5551234
1w5d: //47/000000000000/VXML:/vxml_digit_collection_done: name v0
Router#
1w5d: //47/000000000000/VXML:/vxml_placecall_done: duration=0 status=0 async_st0
Router#
1w5d: //47/000000000000/VXML:/vxml_user_hangup: duration 3 status=A async_statu0

```

The following example shows output from the **debug vxml grammar** command:

```
Router# debug vxml grammar

vxml xml grammar syntax checking debugging is on
Router#
Feb 11 13:47:25.110: //-1//VAPP:/vapp_get_apphandler:
*Feb 11 13:47:25.114: vapp_get_apphandler: Script help
*Feb 11 13:47:25.114: //-1//VAPP:/vapp_get_apphandler_core:
*Feb 11 13:47:25.114: //-1/000000000000/VAPP:/vapp_InterpInitConfigParams:
*Feb 11 13:47:25.114: //-1//VAPP:/vapp_init_apphandler:
*Feb 11 13:47:25.114: //-1/000000000000/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got
event CC_EV_CALL_SETUP_IND
*Feb 11 13:47:25.114: //-1/000000000000/VAPP:/vapp_driver: pInterp[62DD481C]:
*Feb 11 13:47:25.114: //-1/000000000000/VAPP:/vapp_driver: evtID: 28 vapp record state: 0
*Feb 11 13:47:25.114: //-1/000000000000/VAPP:/vapp_evt_setup:
*Feb 11 13:47:25.114: //-1//VAPP:/vapp_incoming_callblock:
*Feb 11 13:47:25.114: vapp_incoming_callblock:
*Feb 11 13:47:25.114: //7/9AC9CCF28008/VAPP:/vapp_load_or_run_script:
*Feb 11 13:47:25.114: //7/9AC9CCF28008/VAPP:/vapp_load_or_run_script:
*Feb 11 13:47:25.114: The VXML Script with len=741 starts:
-----
<?xml version = "1.0"?>
<vxml version = "2.0">

<property name="universals" value="all"/>
<form id="check_help">
  <field name="book">
    <grammar version="1.0" mode="voice" xml:lang="en-US">

*Feb 11 13:47:25.114: //-1//VXML:/vxml_create: url=tftp://dirt/lshen/regression/help.vxml
vapphandle=62DD481C
*Feb 11 13:47:25.114: //-1//VXML:/vxml_mem_init:
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VXML:/vxml_rule_build:
tftp://dirt/lshen/regression/help.vxml at line 8: attribute <rule> with invalid value
(wrong_scope)
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VXML:/vxml_create:
*Feb 11 13:47:25.118: code=ERROR vapp=VAPP_SUCCESS vxml=VXML_ERROR_INVALID
*Feb 11 13:47:25.118: //-1//VXML:/vxml_mem_free:
*Feb 11 13:47:25.118: //-1//VXML:/vxml_mem_free1:
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VAPP:/vapp_terminate:
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VAPP:/vapp_session_exit_event_name: Exit Event
vxml.session.complete
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VAPP:/vapp_checksessionstate:
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VAPP:/vapp_terminate_initiation:
*Feb 11 13:47:25.118: //7/9AC9CCF28008/VAPP:/vapp_checkifdone:
*Feb 11 13:47:25.122: //7/9AC9CCF28008/VAPP:/vapp_evt_handler: State VAPP_CLEANING got
event CC_EV_CALL_MODIFY_DONE
*Feb 11 13:47:25.122: //7/9AC9CCF28008/VAPP:/vapp_cleaner:
*Feb 11 13:47:25.122: //7/9AC9CCF28008/VAPP:/vapp_cleaner: Ignoring Event
CC_EV_CALL_MODIFY_DONE(36) in Cleanup
*Feb 11 13:47:25.122: //7/9AC9CCF28008/VAPP:/vapp_checksessionstate:
*Feb 11 13:47:25.122: //7/9AC9CCF28008/VAPP:/vapp_checkifdone:
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_evt_handler: State VAPP_CLEANING got
event CC_EV_CALL_DISCONNECT_DONE
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_cleaner:
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_checksessionstate:
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_checkifdone:
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_evt_handler: State VAPP_CLEANING got
event APP_EV_VXMLINTERP_DONE
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_cleaner:
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_cleaner: VxmlDialogDone
event=vxml.session.complete, status 3
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_popifdone:
```

```
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_checkifdone:
*Feb 11 13:47:25.138: //-1//VAPP:/vapp_process: Interp Done
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_cleanup_apphandler:
*Feb 11 13:47:25.138: vapp_cleanup_apphandler: Terminate FALSE Terminated
TRUE{HAN[VXML_HAN] [NULL ] ( )}
*Feb 11 13:47:25.138: //7/9AC9CCF28008/VAPP:/vapp_free_apphandler: {HAN[VXML_HAN] [NULL
] ( )}
```

The following example shows output from the **debug vxml ssml** command:

```
Router# debug vxml ssml
```

```
Router#
vxml ssml syntax checking debugging is on
Feb 11 13:55:28.994: //-1//VAPP:/vapp_get_apphandler:
*Feb 11 13:55:28.994: vapp_get_apphandler: Script help
*Feb 11 13:55:28.994: //-1//VAPP:/vapp_get_apphandler_core:
*Feb 11 13:55:28.994: //-1/A93E3F8F800E/VAPP:/vapp_InterpInitConfigParams:
*Feb 11 13:55:28.998: //-1//VAPP:/vapp_init_apphandler:
*Feb 11 13:55:28.998: //-1/003E3F8F800E/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got
event CC_EV_CALL_SETUP_IND
*Feb 11 13:55:28.998: //-1/003E3F8F800E/VAPP:/vapp_driver: pInterp[62DD481C]:
*Feb 11 13:55:28.998: //-1/003E3F8F800E/VAPP:/vapp_driver: evtID: 28 vapp record state: 0
*Feb 11 13:55:28.998: //-1/003E3F8F800E/VAPP:/vapp_evt_setup:
*Feb 11 13:55:28.998: //-1//VAPP:/vapp_incoming_callblock:
*Feb 11 13:55:28.998: vapp_incoming_callblock:
*Feb 11 13:55:28.998: //10/BB2F243F8011/VAPP:/vapp_load_or_run_script:
*Feb 11 13:55:28.998: //10/BB2F243F8011/VAPP:/vapp_load_or_run_script:
*Feb 11 13:55:28.998: The VXML Script with len=760 starts:
-----
<?xml version = "1.0"?>
<vxml version = "2.0">

<property name="universals" value="all"/>
<form id="check_help">
  <field name="book">
    <grammar version="1.0" mode="voice" xml:lang="en-US">

*Feb 11 13:55:28.998: //-1//VXML:/vxml_create: url=tftp://dirt/lshen/regression/help.vxml
vappphandle=62DD481C
*Feb 11 13:55:28.998: //-1//VXML:/vxml_mem_init:
*Feb 11 13:55:29.002: //10/BB2F243F8011/VXML:/vxml_parse:
tftp://dirt/lshen/regression/help.vxml at line 16: mismatched tag
*Feb 11 13:55:29.002: //10/BB2F243F8011/VXML:/vxml_create:
*Feb 11 13:55:29.002: code=ERROR vapp=VAPP_SUCCESS vxml=VXML_ERROR_INVALID
*Feb 11 13:55:29.002: //-1//VXML:/vxml_mem_free:
*Feb 11 13:55:29.002: //-1//VXML:/vxml_mem_free1:
*Feb 11 13:55:29.002: //10/BB2F243F8011/VAPP:/vapp_terminate:
*Feb 11 13:55:29.002: //10/BB2F243F8011/VAPP:/vapp_session_exit_event_name: Exit Event
vxml.session.complete
*Feb 11 13:55:29.002: //10/BB2F243F8011/VAPP:/vapp_checksessionstate:
*Feb 11 13:55:29.002: //10/BB2F243F8011/VAPP:/vapp_terminate_initiation:
*Feb 11 13:55:29.002: //10/BB2F243F8011/VAPP:/vapp_checkifdone:
*Feb 11 13:55:29.006: //10/BB2F243F8011/VAPP:/vapp_evt_handler: State VAPP_CLEANING got
event CC_EV_CALL_MODIFY_DONE
*Feb 11 13:55:29.006: //10/BB2F243F8011/VAPP:/vapp_cleaner:
*Feb 11 13:55:29.006: //10/BB2F243F8011/VAPP:/vapp_cleaner: Ignoring Event
CC_EV_CALL_MODIFY_DONE(36) in Cleanup
*Feb 11 13:55:29.006: //10/BB2F243F8011/VAPP:/vapp_checksessionstate:
*Feb 11 13:55:29.006: //10/BB2F243F8011/VAPP:/vapp_checkifdone:
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_evt_handler: State VAPP_CLEANING got
event CC_EV_CALL_DISCONNECT_DONE
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_cleaner:
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_checksessionstate:
```

```
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_checkifdone:
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_evt_handler: State VAPP_CLEANING got
event APP_EV_VXMLINTERP_DONE
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_cleaner:
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_cleaner: VxmlDialogDone
event=vxml.session.complete, status 3
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_popifdone:
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_checkifdone:
*Feb 11 13:55:29.022: //-1//VAPP:/vapp_process: Interp Done
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_cleanup_apphandler:
*Feb 11 13:55:29.022: vapp_cleanup_apphandler: Terminate FALSE Terminated
TRUE{HAN[VXML_HAN][NULL]()}
*Feb 11 13:55:29.022: //10/BB2F243F8011/VAPP:/vapp_free_apphandler: {HAN[VXML_HAN][NULL]
}()
```

The following example shows output from the **debug vxml trace** command:

```
Router# debug vxml trace
```

```
vxml trace debugging is on
Router#
1w5d: //-1//VXML:/vxml_mem_init:
1w5d: //51/359408288031/VXML:/vxml_offramp_mailhdrs_get:
1w5d: //51/359408288031/VXML:/vxml_start: vxmlhandle=65350A7C vapphandle=660E100
1w5d: //51/359408288031/VXML:/vxml_vxml_proc:
1w5d: <vxml> URI(abs):flash:call.vxml scheme=flash path=call.vxml base= URI(abs0
1w5d: <var>: namep=phone_num
1w5d: //-1//VXML:/vxml_stand_alone: scope=document, application = document
1w5d: //51/359
Router#408288031/VXML:/vxml_form_proc:
1w5d: <form>: id=main scope=dialog
1w5d: vxml_form_init current scope: dialog
1w5d: vxml_counter_reset:
1w5d: vxml_counter_reset:
1w5d: //51/359408288031/VXML:/vxml_formitem_select: Status=VXML_STATUS_OK,
1w5d: //51/359408288031/VXML:/vxml_formitem_select: AsyncStatus=VXML_STATUS_OK
1w5d: //51/359408288031/VXML:/vxml_block_proc:
1w5d: <block>:
1w5d: //51/359408288031/VXML:/vxml_item_attrs_proc: name=_in6
1w5d: //51/359408288031/VXML:/vxml_expr_eval: exp
Router#r=dialog._in6='defined'
1w5d: //51/359408288031/VXML:/vxml_prompt_proc:
1w5d: <prompt>: bargein=0 count=1 typeaheadflush=0
1w5d: //51/359408288031/VXML:/vxml_audio_proc:
1w5d: <audio>: URI(abs):flash:welcome_test.au scheme=flash path=welcome_tes0
1w5d: //51/359408288031/VXML:/vxml_vapp_media_play: bargein=0 timeout=0 typeahe0
1w5d: //51/359408288031/VXML:/vxml_vapp_media_play:
1w5d: //51/359408288031/VXML:/vxml_vapp_me
Router#dia_play: audio=flash:welcome_test.au cachable=1 timeout20
1w5d: //51/359408288031/VXML:/vxml_leave_scope: scope=8
1w5d: //51/359408288031/VXML:/vxml_vapp_vcr_control_disable:
1w5d: //51/359408288031/VXML:/vxml_start: vxmlhandle=65350A7C vapphandle=660E100
1w5d: //51/359408288031/VXML:/vxml_vxml_proc:
1w5d: <vxml> URI(abs):flash:call.vxml scheme=flash path=call.vxml base= URI(abs0
1w5d: //51/359408288031/VXML:/vxml
Router#l_block_proc:
1w5d: <block>:
1w5d: //51/359408288031/VXML:/vxml_item_attrs_proc: name=_in6
1w5d: //51/359408288031/VXML:/vxml_form_proc:
1w5d: <form>: id=main scope=dialog
1w5d: //51/359408288031/VXML:/vxml_formitem_select: Status=VXML_STATUS_OK,
1w5d: //51/359408288031/VXML:/vxml_formitem_select: AsyncStatus=VXML_STATUS_OK
1w5d: //51/359408288031/VXML:/vxml_field_proc:
1w5d: <field>: type=number
1w5d: //51/359408288031/VXML:/vxml_item_attrs_proc: name=get_phone_num modal=am
```

```

Router#pt_counter=1
lw5d: //51/359408288031/VXML:/vxml_prompt_proc:
lw5d:   <prompt>: bargein=1 count=1 typeaheadflush=0
lw5d: //51/359408288031/VXML:/vxml_audio_proc:
lw5d:   <audio>: URI(abs):flash:enter_dest.au scheme=flash path=enter_dest.au0
lw5d: //51/359408288031/VXML:/vxml_vapp_media_play: bargein=1 timeout=0 typeahe0
lw5d: //51/359408288031/VXML:/vxml_vapp_media_play:
lw5d: //51/359408288031/VXML:/vxml_vapp_media_play: audio
Router#=flash:enter_dest.au cachable=1 timeout20
lw5d: //51/359408288031/VXML:/vxml_vapp_vcr_control_disable:
lw5d: //51/359408288031/VXML:/vxml_vapp_digit_collect: termchar # maxDigits 0 t0
lw5d: //51/359408288031/VXML:/vxml_start: vxmlhandle=65350A7C vapphandle=660E100
lw5d: //51/359408288031/VXML:/vxml_vxml_proc:
lw5d: <vxml> URI(abs):flash:call.vxml scheme=flash path=call.vxml base= URI(abs1
Router#.0
lw5d: //51/359408288031/VXML:/vxml_field_proc:
lw5d:   <field>: type=number
lw5d: //51/359408288031/VXML:/vxml_item_attrs_proc: name=get_phone_num modal=a2
lw5d: //51/359408288031/VXML:/vxml_filled_proc:
lw5d:
lw5d: <filled>: mode=all
lw5d: //51/359408288031/VXML:/vxml_assign_proc:
lw5d:   <assign>: namep=phone_num expr=get_phone_num
lw5d: //51/359408288031/VXML:/vxml_goto_proc:
lw5d:   <goto>: caching=fast fetchhint=invalid fetchtimeout=20 URI:#transfer_mm
Router#entp=transfer_me
lw5d:   vxml_dialog_reset:
lw5d: //51/359408288031/VXML:/vxml_leave_scope: scope=110
lw5d: //51/359408288031/VXML:/vxml_leave_scope: scope=8
lw5d: //51/359408288031/VXML:/vxml_vxml_proc:
lw5d: <vxml> URI(abs):flash:call.vxml scheme=flash path=call.vxml base= URI(abs0
lw5d: //51/359408288031/VXML:/vxml_form_proc:
lw5d:   <form>: id=transfer_me scope=dialog
lw5d: vxml_form_init current scope: dialog
lw5d:   <var>: namep=myd
Router#ur
lw5d:   vxml_counter_reset:
lw5d: //51/359408288031/VXML:/vxml_formitem_select: Status=VXML_STATUS_OK,
lw5d: //51/359408288031/VXML:/vxml_formitem_select: AsyncStatus=VXML_STATUS_OK
lw5d: //51/359408288031/VXML:/vxml_transfer_proc:
lw5d:   <transfer>:
lw5d: //51/359408288031/VXML:/vxml_item_attrs_proc: name=mycall dest_expr='phoe
Router#ctreason=-1
lw5d: //51/359408288031/VXML:/vxml_vapp_placecall: dest 5551234 timeout 15 maxl0
lw5d: //51/359408288031/VXML:/vxml_vapp_gain_control_default:
lw5d: //51/359408288031/VXML:/vxml_expr_eval: expr=dialog.mycall = 'far_end_dis'
lw5d: //51/359408288031/VXML:/vxml_expr_eval: expr=dialog.mycall$.duration = 2
lw5d: //51/359408288031/VXML:/vxml_start: vxmlhandle=65350A7C vapphandle=660E100
lw5d: //51/359408288031/VXML:/vxml_vxml
Router#_proc:
lw5d: <vxml> URI(abs):flash:call.vxml scheme=flash path=call.vxml base= URI(abs0
lw5d: //51/359408288031/VXML:/vxml_transfer_proc:
lw5d:   <transfer>:
lw5d: //51/359408288031/VXML:/vxml_item_attrs_proc: name=mycall URI(abs):phone-
Router#1, redirectreason=-1
lw5d: //51/359408288031/VXML:/vxml_form_proc:
lw5d:   <form>: id=transfer_me scope=dialog
lw5d: //51/359408288031/VXML:/vxml_filled_proc:
lw5d:
lw5d: <filled>: mode=all
lw5d: //51/359408288031/VXML:/vxml_assign_proc:
lw5d:   <assign>: namep=mydur expr=mycall$.duration
lw5d: //51/359408288031/VXML:/vxml_if_proc:
lw5d:   <if>: cond=mycall == 'busy'

```

```

1w5d: //51/359408288031/VXML:/vxml_leave_scope: scope=8
1w5d: //51/359408288031/VXML:/vxml_foritem_select: Status=VXML_ST
Router#ATUS_OK,
1w5d: //51/359408288031/VXML:/vxml_foritem_select: AsyncStatus=VXML_STATUS_OK
1w5d: //51/359408288031/VXML:/vxml_foritem_select: the form is full
1w5d: //51/359408288031/VXML:/vxml_vapp_terminate: vapp_status=0 ref_count 0
1w5d: //-1//VXML:/vxml_mem_free:
1w5d: //-1//VXML:/vxml_mem_free1:

```

Related Commands

| Command | Description |
|--|--|
| debug condition application voice | Displays debugging messages for only the specified VoiceXML application. |
| debug http client | Displays debugging messages for the HTTP client. |
| debug voip ivr | Displays debug messages for VoIP IVR interactions. |