debug vpm spi through voice call debug

debug vpm spi

To trace how the voice port module security parameter index (SPI) interfaces with the call control application programming interface (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.

Command Modes

Privileged EXEC

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 presented 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
vcsm_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.

```
vcsm_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
vcsm_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.

```
vcsm_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
vcsm_timer: 47055
vcsm_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
vcsm_timer: 47079
vcsm_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
vcsm_timer: 47173
vcsm_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
vcsm_timer: 47197
vcsm_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
vcsm_timer: 47217
vcsm_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:

```
vcsm_process_event: [1/0/1, 0.8, 15] act_bridge
vcsm_process_event: [1/0/1, 0.8, 20] act_caps_ind
vcsm_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:

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

The following output shows the voice quality statistics collected periodically:

```
vcsm_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
vcsm_process_event: [1/0/1, 0.13, 28]
vcsm_process_event: [1/0/1, 0.13, 29]
vcsm_process_event: [1/0/1, 0.13, 32]
vcsm_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
vcsm_process_event: [1/0/1, 0.13, 28]
vcsm_process_event: [1/0/1, 0.13, 29]
vcsm_process_event: [1/0/1, 0.13, 32]
vcsm_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
vcsm_process_event: [1/0/1, 0.13, 28]
vcsm_process_event: [1/0/1, 0.13, 29]
vcsm_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:

```
vcsm_process_event: [1/0/1, 0.13, 4] act_generate_disc
vcsm_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
vcsm_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
vcsm_timer: 48762
vcsm_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
vcsm_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
vcsm_timer: 48762
dsp_set_sig_state: [1/0/1] packet_len=14 channel_id=129 packet_id=39 state=0x4 timestamp=0x0
vcsm_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.

Command Default

Trunk conditioning supervisory component trace information is not displayed.

Command Modes

Privileged EXEC

Command History

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

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 Cisco 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	Displays messages to and from the DSP.
from_dsp	Displays messages from the DSP.
to_dsp	Displays messages to the DSP.

Command Default

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 integrated on the Cisco 7200 series.

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 is 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	Displays messages to and from the DSP.
from_dsp	Displays messages from the DSP.
to_dsp	Displays messages to the DSP.

Command Default

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 implemented on the Cisco 7200 series.

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 type 3** command and identify a specific type 3 (control) packet type.

Examples

The following is 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	Displays type 3 alarm packets.
alltype3	Displays all type 3 packets.
cas	Displays type 3 channel-associated switching (CAS) packets.
dialed	Displays type 3 dialed digit packets.
faxrelay	(Not supported) Displays type 3 fax relay packets.
state	Displays type 3 user state packets.
all_dsp	Displays messages to and from the DSP.
from_dsp	Displays messages from the DSP.
to_dsp	Displays messages to the DSP.

Command Default

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 implemented on the Cisco 7200 series.

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 is 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 vrf

To get debugging information on virtual routing and forwarding (VRF) instances, use the **debug vrf**command in privileged EXEC mode. To turn off the debug output, use the **undebug** version of the command.

debug vrf {**create**| **delete**| **error**| **ha**| **initialization**| **interface**| **ipv4**| **ipv6**| **issu**| **lock**| **lookup**| **mpls**| **selection**}

undebug vrf {**create**| **delete**| **error**| **ha**| **initialization**| **interface**| **ipv4**| **ipv6**| **issu**| **lock**| **lookup**| **mpls**| **selection**}

Syntax Description

create	Specifies VRF creation debugging.
delete	Specifies VRF deletion debugging.
error	Specifies VRF error debugging.
ha	Specifies VRF high-availability debugging.
initialization	Specifies VRF subsystem initialization debugging.
interface	Specifies VRF interface assignment debugging.
ipv4	Specifies VRF IPv4 address family debugging.
ipv6	Specifies VRF IPv6 address family debugging.
issu	Specifies VRF in-service software upgrade debugging.
lock	Specifies VRF lock debugging.
lookup	Specifies VRF database lookup debugging.
mpls	Specifies VRF multiprotocol label switching debugging.
selection	Specifies VRF selection debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.2S	This command was introduced.

Usage Guidelines

Use this command to get debugging information on VRFs.

Examples

The following example shows how to turn on debugging of VRF interface assignment:

```
Router# debug vrf interface
```

Related Commands

Command	Description
vrf definition	Defines a virtual routing and forwarding instance.

debug vrrp all

To display debugging 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(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(31)SG	This command was integrated into Cisco IOS Release 12.2(31)SG.
12.2(17d)SXB	This command was integrated into Cisco IOS Release 12.2(17d)SXB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.
Cisco IOS XE Release 2.6	This command was modified. This output was modified to display VRRP debugging statements for Virtual Router Redundancy Service (VRRS).

Examples

The following is sample output from 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 debugging messages about VRRP error conditions.
debug vrrp events	Displays debugging messages about VRRP events.
debug vrrp state	Displays debugging messages about the VRRP state transitions.

debug vrrp authentication

To display debugging messages for Virtual Router Redundancy Protocol (VRRP) Message Digest 5 (MD5) authentication, use the **debug vrrp authentication**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vrrp authentication

no debug vrrp authentication

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(14)T	This command was introduced.

Examples

The following sample output shows that MD5 authentication is enabled on one router but not the other:

```
Router# debug vrrp authentication
VRRP: Grp 1 Adv from 172.24.1.2 has incorrect auth type 1 expected 0
```

The following sample output shows that the MD5 key IDs and key strings differ on each router:

```
Router# debug vrrp authentication
VRRP: Sent: 21016401FE050000AC1801FE0000000000000000
VRRP: HshC: B861CBF1B9026130DD34AED849BEC8A1
VRRP: Rcvd: 21016401FE050000AC1801FE0000000000000000
VRRP: HshC: B861CBF1B9026130DD34AED849BEC8A1
VRRP: HshR: C5E193C6D84533FDC750F85FCFB051E1
VRRP: Grp 1 Adv from 172.24.1.2 has failed MD5 auth
```

The following sample output shows that the text authentication strings differ on each router:

```
Router# debug vrrp authentication
VRRP: Grp 1 Adv from 172.24.1.2 has failed TEXT auth
```

Related Commands

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

debug vrrp error

To display debugging 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(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(31)SG	This command was integrated into Cisco IOS Release 12.2(31)SG.
12.2(17d)SXB	This command was integrated into Cisco IOS Release 12.2(17d)SXB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Examples

The following is sample output from the **debug vrrp error**command:

```
Router# debug vrrp error
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 debugging messages for VRRP errors, events, and state transitions.

debug vrrp events

To display debugging 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(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(31)SG	This command was integrated into Cisco IOS Release 12.2(31)SG.
12.2(17d)SXB	This command was integrated into Cisco IOS Release 12.2(17d)SXB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Examples

The following is sample output from 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 debugging messages for VRRP errors, events, and state transitions.

debug vrrp ha

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

debug vrrp ha

no debug vrrp ha

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRC	This command was introduced.
12.2(33)SB2	This command was integrated into Cisco IOS Release 12.2(33)SB2.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

Examples

The following examples for the **debug vrrp ha**command display the syncing of VRRP state information from the Active RP to the Standby RP.

The following sample output displays two VRRP state changes on the Active RP:

```
Router# debug vrrp ha
.
.
.
*Nov 14 11:36:50.272 UTC: VRRP: Gi3/2 Grp 42 RF Encode state Backup into sync buffer
*Nov 14 11:36:50.272 UTC: %VRRP-6-STATECHANGE: Gi3/2 Grp 42 state Init -> Backup
*Nov 14 11:36:53.884 UTC: VRRP: Gi3/2 Grp 42 RF Encode state Master into sync buffer
*Nov 14 11:36:53.884 UTC: %VRRP-6-STATECHANGE: Gi3/2 Grp 42 state Backup -> Master
```

The following sample output displays two VRRP state changes on the Standby RP:

```
Router# debug vrrp ha
.
.
.
*Nov 14 11:36:50.392 UTC: STDBY: VRRP: Gi3/2 Grp 42 RF sync state Init -> Backup
*Nov 14 11:36:53.984 UTC: STDBY: VRRP: Gi3/2 Grp 42 RF sync state Backup -> Master
```

Related Commands

Command	Description
debug vrrp error	Displays debugging messages about VRRP error conditions.
debug vrrp events	Displays debugging messages about VRRP events.
debug vrrp state	Displays debugging messages about the VRRP 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(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.

Command History

12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(31)SG	This command was integrated into Cisco IOS Release 12.2(31)SG.
12.2(17d)SXB	This command was integrated into Cisco IOS Release 12.2(17d)SXB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Examples

The following is sample output from 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 debugging 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(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(31)SG	This command was integrated into Cisco IOS Release 12.2(31)SG.
12.2(17d)SXB	This command was integrated into Cisco IOS Release 12.2(17d)SXB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Examples

The following is sample output from 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 debugging messages for VRRP errors, events, and state transitions.

debug vrrp vrrs

To enable Virtual Router Redundancy Protocol (VRRP) debugging statements for Virtual Router Redundancy Service (VRRS) interactions, use the **debug vrrp vrrs** command in privileged EXEC mode. To disable VRRP VRRS debugging statements, use the **no** form of this command.

debug vrrp vrrs

no debug vrrp vrrs

Syntax Description

This command has no arguments or keywords.

Command Default

VRRP debugging for VRRS interactions is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.6	This command was introduced.

Examples

The following is sample output from the **debug vrrp vrrs** command:

```
Router# debug vrrp vrrs
VRRP VRRS debugging is on
The following is sample output from the debug vrrp vrrs
 command when a VRRP group is configured with a name association to ‘name1’:
Router# configure termina
l
Router(config)# interface gigabitethernet0/0/0
Router(config-if)# ip address 10.0.0.1 255.0.0.0
Router(config-if)# vrrp 1 ip 10.0.0.7
Router(config-if)# vrrp 1 name name1
*Feb  5 09:29:47.005: VRRP: Registered VRRS group "name1"
```

The following is sample output when a VRRP group is brought up:

```
Router(config-if)# no shutdown
*Feb  5 09:29:53.237: VRRP: Updated info for VRRS group name1
```

The following is sample output when a name association is changed to a different name:

```
Router(config-if)# vrrp 1 name name2
*Feb  5 09:30:14.153: VRRP: Unregistered VRRS group "name1"
*Feb  5 09:30:14.153: VRRP: Registered VRRS group "name2"
```

The following is sample output when a name association for group is removed:

```
Router(config-if)# no vrrp 1 name
*Feb  5 09:30:22.689: VRRP: Unregistered VRRS group "name2"
```

Related Commands

Command	Description
debug vrrs accounting	Enables debug messages for VRRS accounting.
debug vrrs infra	Enables VRRS infrastructure debug messages.
debug vrrs plugin	Enables VRRS plug-in debug messages.

debug vrrs all

To enable debugging information associated with all elements of Virtual Router Redundancy Service (VRRS), use the **debug vrrs all** command in Privileged EXEC mode.

debug vrrs all [**detail**]

Syntax Description

detail	(Optional) Enables detailed debugging information associated with VRRS pathways and databases.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(1)S	This command was introduced.
Cisco IOS XE Release 3.8S	This command was introduced.

Usage Guidelines

You must configure the VRRS control groups using the **vrrs** command on interfaces that require a redundant virtual gateway.

Examples

The following example shows how to enable debugging information associated with all elements of VRRS using the **debug vrrs all** command:

```
Device# debug vrrs all

vrrs database client debugging is on
vrrs database error debugging is on
vrrs database event debugging is on
vrrs database server debugging is on
vrrs database tag debugging is on
vrrs pathway event debugging is on
vrrs pathway database debugging is on
vrrs pathway error debugging is on
vrrs pathway mac debugging is on
vrrs pathway address resolution protocol debugging is on
vrrs pathway process debugging is on
vrrs pathway state debugging is on
vrrs pathway address debugging is on
```

Related Commands

Command	Description
debug vrrs log	Enables debugging information associated with VRRS logs.
debug vrrs database	Enables debugging information associated with VRRS databases.
debug vrrs pathway	Enables debugging information associated with VRRS pathways.

debug vrrs accounting

To enable debug messages for Virtual Router Redundancy Service (VRRS) accounting, use the **debug vrrs accounting** command in privileged EXEC mode. To disable VRRS accounting debug messages, use the **no** form of this command.

debug vrrs accounting {**all**| **errors**| **events**}

no debug vrrs accounting command {**all**| **errors**| **events**}

Syntax Description

all	Enables all VRRS accounting debug messages.
errors	Enables VRRS accounting error debug messages.
events	Enables VRRS accounting event debug messages.

Command Default

VRRS accounting debug messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.6	This command was introduced.

Examples

The following example turns on all VRRS accounting debug messages:

```
Router# debug vrrs accounting all
00:16:13: VRRS/ACCT/EV: entry create for abc(0x4E8C1F0)
00:16:13: VRRS/ACCT/EV: abc(0x4E8C1F0 12000006) client add ok2(No group)
```

Related Commands

Command	Description
debug vrrp vrrs	Enables VRRP debugging statements for VRRS interactions.
debug vrrs accounting	Enables debug messages for VRRS accounting.
debug vrrs infra	Enables VRRS infrastructure debug messages.
debug vrrs plugin	Enables VRRS plug-in debug messages.

debug vrrs database

To enable debugging information associated with the Virtual Router Redundancy Services (VRRS) database, use the **debug vrrs database** command in Privileged EXEC mode.

debug vrrs database {**all** [**detail**]| {**client**| **error**| **event**| **server**| **tag**} [**Ethernet** *number* [**IPv4** [**verbose**]| **IPv6** [**verbose**]]| **IPv4** [**Ethernet** *number* [**verbose**]| **verbose** [**Ethernet** *number*]]| **IPv6** [**Ethernet** *number* [**verbose**]| **verbose** [**Ethernet** *number*]]] [**detail**]}

Syntax Description

all	Enables debugging information associated with all VRRS databases.
detail	(Optional) Enables detailed debugging information associated with all VRRS databases.
client	Enables debugging information associated with VRRS database clients.
error	Enables debugging information associated with VRRS database errors.
event	Enables debugging information associated with VRRS database events.
server	Enables debugging information associated with VRRS database servers.
tag	Enables debugging information associated with VRRS database tags.
Ethernet *number*	(Optional) Enables debugging information associated with VRRS database for ethernet interfaces.
IPv4	(Optional) Enables debugging information associated with VRRS database for VRRP groups adhering to IPv4 protocol.
verbose	(Optional) Enables debugging information associated with VRRS database for groups adhering to non-protocol events.
IPv6	(Optional) Enables debugging information associated with VRRS database for VRRP groups adhering to IPv6 protocol.

Command Modes Privileged EXEC (#)

Command History

Release	Modification
15.3(1)S	This command was introduced.
Cisco IOS XE Release 3.8S	This command was introduced.

Usage Guidelines You must configure the VRRS control groups using the **vrrs** command on interfaces that require a redundant virtual gateway.

Examples The following example shows how to enable debugging information associated with all elements of VRRS database using the **debug vrrs database** command with the **all** keyword:

```
Device# debug vrrs database all

vrrs database client debugging is on
vrrs database error debugging is on
vrrs database event debugging is on
vrrs database server debugging is on
vrrs database tag debugging is on
```

Related Commands

Command	Description
debug vrrs all	Enables debugging information associated with all elements of Virtual Router Redundancy Service (VRRS).
debug vrrs log	Enables debugging information associated with VRRS logs.
debug vrrs pathway	Enables debugging information associated with VRRS pathways.

debug vrrs infra

To enable Virtual Router Redundancy Service (VRRS) infrastructure debug messages, use the **debug vrrs infra** command in privileged EXEC mode. To turn off VRRS infrastructure debugging, use the **no** form of this command.

debug vrrs infra {**all**| **client**| **events**| **server**}

no debug vrrs infra {**all**| **client**| **events**| **server**}

Syntax Description

all	Enables all VRRS infrastructure debug messages.
client	Enables debugging for VRRS infrastructure to VRRS client interactions.
events	Enables debugging for VRRS infrastructure events.
server	Enables debugging for VRRS infrastructure to VRRS server interactions.

Command Default

VRRS debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.6	This command was introduced.

Examples

The following is sample output from the **debug vrrs infra**command:

```
Router# debug vrrs infra all
*Sep 9 16:09:53.848: VRRS: Client 21 is not registered
*Sep 9 16:09:53.848: VRRS: Client 21 unregister failed
*Sep 9 16:09:53.848: VRRS: Client VRRS TEST CLIENT registered, id 21
*Sep 9 16:09:53.848: VRRS: Client 21 add, group VRRP-TEST-1 does not exist, allocating...
*Sep 9 16:09:53.848: VRRS: Client 21 add to VRRP-TEST-1. Vrrs handle F7000001, client handle
 FE720
*Sep 9 16:09:53.848: VRRS: Server VRRP add, group VRRP-TEST-1, state INIT, vrrs handle
F7000001
*Sep 9 16:09:53.876: VRRS: VRRP-TEST-1 group added notification
*Sep 9 16:09:53.876: VRRS: Normal priority clients for group 200000, for all groups[4C0
*Sep 9 16:09:53.876: VRRS: Client 2 add to VRRP-TEST-1. Vrrs handle F7000001, client handle
 22766F0
*Sep 9 16:09:54.356: VRRS: Client 21 remove from VRRP-TEST-1. vrrs handle F7000001
*Sep 9 16:09:54.356: VRRS: Server VRRP delete, group VRRP-TEST-1 vrrs handle F7000001
*Sep 9 16:09:54.360: VRRS: VRRP-TEST-1 group deleted notification
```

```
*Sep 9 16:09:54.360: VRRS: Low priority clients 4
*Sep 9 16:09:54.360: VRRS: Client 2 remove from VRRP-TEST-1. vrrs handle F7000001
*Sep 9 16:09:54.360: VRRS: client remove, no more clients and no server for group VRRP-TEST-1.
 Remov
*Sep 9 16:09:54.860: VRRS: Client 22 is not registered
*Sep 9 16:09:54.860: VRRS: Client 22 unregister failed
*Sep 9 16:09:54.860: VRRS: Client VRRS TEST CLIENT registered, id 22
```

Related Commands

Command	Description
debug vrrp vrrs	Enables VRRP debugging statements for VRRS interactions.
debug vrrs accounting	Enables debug messages for VRRS accounting.
debug vrrs plugin	Enables VRRS plug-in debug messages.

debug vrrs log

debug vrrs log [**detail**]

Syntax Description

detail	(Optional) Enables detailed debugging information associated with VRRS logs.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(1)S	This command was introduced.
Cisco IOS XE Release 3.8S	This command was introduced.

Usage Guidelines

You must configure the VRRS control groups using the **vrrs** command on interfaces that require a redundant virtual gateway.

Examples

Related Commands

Command	Description
debug vrrs all	Enables debugging information associated with all elements of Virtual Router Redundancy Service (VRRS).
debug vrrs database	Enables debugging information associated with VRRS databases.
debug vrrs pathway	Enables debugging information associated with VRRS pathways.

debug vrrs pathway

debug vrrs pathway {**all** [**detail**]| **process** [**detail**]| **address** [*ipv4-address* [**Ethernet** *number*]| **Ethernet** *number* [*ipv4-address*| **IPv4**| **IPv6**| *ipv6-address*]| **IPv4** [**Ethernet** *number*]| **IPv6** [**Ethernet** *number*]| *ipv6-address* [**Ethernet** *number*]] [**detail**]| {**database**| **error**| **event**| **mac-address**| **protocol**| **state**} [**Ethernet** *number* [**IPv4** [**verbose**]| **IPv6** [**verbose**]]| **IPv4** [**Ethernet** *number* [**verbose**]| **verbose** [**Ethernet** *number*]]| **IPv6** [**Ethernet** *number* [**verbose**]| **verbose** [**Ethernet** *number*]]] [**detail**]}

Syntax Description

all	Enables debugging information associated with all VRRS pathways.
detail	(Optional) Enables detailed debugging information associated with all VRRS pathways.
process	Enables debugging information associated with VRRS pathway processes.
address	Enables debugging information associated with VRRS pathway addresses.
ipv4-address	Enables debugging information associated with IPv4 addresses on VRRS pathways.
Ethernet *number*	(Optional) Enables debugging information associated with VRRS pathways for ethernet interfaces.
IPv4	(Optional) Enables debugging information associated with VRRS pathways for VRRP groups adhering to IPv4 protocol.
IPv6	(Optional) Enables debugging information associated with VRRS pathways for VRRP groups adhering to IPv6 protocol.
ipv6-address	Enables debugging information associated with IPv6 addresses on VRRS pathways.
database	Enables debugging information associated with VRRS pathways for databases.
error	Enables debugging information associated with VRRS pathway errors.
event	Enables debugging information associated with VRRS pathway events.

mac-address	Enables debugging information associated with MAC addresses on VRRS pathways.
protocol	Enables debugging information associated with VRRS pathway protocols.
state	Enables debugging information associated with VRRS pathways for interface states.
verbose	Enables debugging information associated with VRRS pathways for non-protocol events.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(1)S	This command was introduced.
Cisco IOS XE Release 3.8S	This command was introduced.

Usage Guidelines

You must configure VRRS pathways by defining the First Hop Redundancy Protocol (FHRP) groups and configuring the interfaces that require redundant virtual gateway.

Examples

The following example shows how to enable debugging information associated with all elements of VRRS using the **debug vrrs platform** command:

```
Device# debug vrrs platform

vrrs pathway event debugging is on
vrrs pathway database debugging is on
vrrs pathway error debugging is on
vrrs pathway mac debugging is on
vrrs pathway address resolution protocol debugging is on
vrrs pathway process debugging is on
vrrs pathway state debugging is on
vrrs pathway address debugging is on
```

Related Commands

Command	Description
debug vrrs all	Enables debugging information associated with all elements of Virtual Router Redundancy Service (VRRS).
debug vrrs database	Enables debugging information associated with VRRS databases.

Command	Description
debug vrrs log	Enables debugging information associated with VRRS logs.

debug vrrs plugin

To enable Virtual Router Redundancy Service (VRRS) plug-in debug messages, use the **debug vrrs plugin**command in privileged EXEC mode. To disable VRRS plug-in debug messages, use the **no** form of this command.

debug vrrs plugin {**all**| **arp-packet**| **client**| **database**| **if-state**| **mac**| **process**| **sublock**| **test**}

no debug vrrs plugin {**all**| **arp-packet**| **client**| **database**| **if-state**| **mac**| **process**| **sublock**| **test**}

Syntax Description

all	Enables all VRRS debugs.
arp-packet	Enables debugging for VRRS mac-address gratuitous ARP messages.
client	Enables debugging for VRRS plug-in client interactions with VRRS.
database	Enables debugging for VRRS plug-in database management.
if-state	Enables VRRS events associated specifically with the VRRS interface-state plug-in.
mac	Enables VRRS events associated specifically with the VRRS mac-address plug-in.
process	Enables debugging for the VRRS plug-in events process.
sublock	Enables debugging for VRRS interface subblock management.
test	Enables VRRS plug-in test code monitoring.

Command Default

VRRS plug-in debug messages are not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.6	This command was introduced.

Examples

The following is sample output when a VRRS borrowed MAC address is added to the MAC address filter of an interface enables VRRS plug-in debug messages:

```
Router)# debug vrrs plugin all
Feb 17 19:15:38.052: VRRS-P(mac): GigEth0/0/0.1 Add 0000.12ad.0001 to MAC filter, using
(afilter_add)
Feb 17 19:15:38.053: VRRS-P(mac): Active count increase to (2) for MAC : 0000.12ad.0001
```

The table below describes the significant fields shown in the display.

Table 1: debug vrrs plugin Field Descriptions

Field	Description
VRRS-P	Specifies this debug is related to VRRS plug-ins.
(mac)	Specifies this debug is related to the VRRS mac-address plug-in. Alternately (if-state) may displayed to indicate the debug is related to the VRRS interface-state plugiplug-inn.

Related Commands

Command	Description
debug vrrp vrrs	Enables VRRP debugging statements for VRRS interactions.
debug vrrs accounting	Enables debug messages for VRRS accounting.
debug vrrs infra	Enables VRRS infrastructure debug messages.

debug vsi api

Note Effective with Cisco IOS Release 12.4(20)T, the **debug vsi api**command is not available in Cisco IOS software.

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.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.4(20)T	This command was removed.

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
```

The table below describes the significant fields shown in the display.

Table 2: 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.
->	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)	The status of a request. One of following status indications appears: OK INVALID_ARGS NONEXIST_INTF TIMEOUT NO_RESOURCES FAIL OK means only that the request is successfully queued for transmission to the switch; it does not indicate completion of the request.

debug vsi errors

Note Effective with Cisco IOS Release 12.4(20)T, the **debug vsi errors** command is not available in Cisco IOS software.

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 *interface*	(Optional) Specifies the interface number.
slave *number*	(Optional) Specifies the slave number (beginning with 0).

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.4(20)T	This command was removed.

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
```

The table below describes the significant fields shown in the display.

Table 3: debug vsi errors 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

Note Effective with Cisco IOS Release 12.4(20)T, the**debug vsi events** command is not available in Cisco IOS software.

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 *interface* >	(Optional) The interface number.
slave > *number*	(Optional) The slave number (beginning with zero).

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

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
```

The table below describes the significant fields shown in the display.

Table 4: debug vsi events 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

Note Effective with Cisco IOS Release 12.4(20)T, the **debug vsi packets**command is not available in Cisco IOS software.

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 *interface*	(Optional) The interface number.
slave > *number*	(Optional) The slave number (beginning with zero).

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.4(20)T	This command was removed.

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
```

The table below describes the significant fields shown in the display.

Table 5: debug vsi packets 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

Note Effective with Cisco IOS Release 12.4(20)T, the **debug vsi param-groups**command is not available in Cisco IOS software.

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**]]

Syntax Description

interface *interface*	(Optional) The interface number.
slave > *number*	(Optional) The slave number (beginning with zero).

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.4(20)T	This command was removed.

Usage Guidelines

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

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

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 encap):
 01.02.00.80 00.00.95.c2 00.00.00.00
Incoming VSI msg of 72 bytes (not including encap):
 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 encap):
 01.02.00.80 00.00.95.c3 00.00.00.00
Incoming VSI msg of 72 bytes (not including encap):
 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
```

The table below describes the significant fields shown in the display.

Table 6: debug vsi param-groups 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.

Command Modes

Privileged EXEC

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 encap 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
```

The table below describes the significant fields shown in the display.

Table 7: 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 encap 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.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to down	Line protocol status: The line protocol is down.

Field	Description
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 messages 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.

Command Default

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	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(15)B	This command was integrated into Cisco IOS Release 12.2(15)B.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(31)SB	This command was integrated into Cisco IOS Release 12.2(31)SB.

Usage Guidelines

The debug messages are displayed if you configure virtual templates with commands that are incompatible with virtual access subinterfaces.

Examples

The following example shows how to display 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
```

The table below describes the significant fields shown in the display.

Table 8: 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.
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

Note Effective with release 12.3(8)T, the **debug vtsp**command is replaced by the **debug voip dsm** and **debug voip vtsp**commands. See the **debug voip dsm** and **debug voip vtsp**commands for more information.

To display the state of the gateway and the call events, use the **debug vtsp**command in privileged EXEC mode. To display the machine state during voice telephony service provider (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**}

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 Cisco IOS Release 12.1(2)T.

Release	Modification
12.2(11)T	The enhancement of debug capabilities, which affects 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.
12.3(8)T	This command was replaced by the **debug voip vtsp** command.

Usage Guidelines

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

Examples

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

Examples

```
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-xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate:
00:11:09:
//-1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate:
00:11:09: //-1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_tsp_call_setup_ind:

00:11:09: //-1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_tsp_fill_setup_ind:
```

Examples

```
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-xxxxxxxxxxxx/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-xxxxxxxxxxxx/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-xxxxxxxxxxxx/VTSP:():-1:-1:-1/vtsp_do_regxrule_translate: .
20:58:16:
//-1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/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-xxxxxxxxxxxx/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-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
//-1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/VTSP:(3/0:23):-1:0:0/vtsp_do_call_setup_ind: Call
 ID=101123, guid=63EB9AC8
```

The table below describes the significant fields shown in the display.

Table 9: 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.

Command Default

Disabled

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, and 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 (xxxxxxxxxxxx).

```
*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],RTMu[0],RTMa[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_bdrop: .
*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]=32639c[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: .
*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.

Command Default

Disabled

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 implemented 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, and 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
```

The table below describes the significant fields shown in the display.

Table 10: 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.

Command Default

Disabled

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, and 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.

Command	Description
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.

Command Default

Disabled

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, and 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 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_CODEC_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*

slot/subunit/port	• *slot* specifies a router slot in which a voice network module (NM) is installed. Valid entries are router slot numbers for the specific platform. • *subunit* 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.) • *port* specifies an analog voice port number. Valid entries are 0 and 1.

Syntax Description

slot/port:ds0-group	Debugs the digital voice port you specify with the *slot/port:ds0-group* designation. • *slot* 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. • *port* 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.) • *ds0-group* specifies a T1 or E1 logical port number. Valid entries are 0 to 23 for T1 and 0 to 30 for E1.

Syntax Description

slot/port	Debugs the analog voice port you specify with the *slot/port*designation. • *slot* is the physical slot in which the analog voice module (AVM) is installed. The *slot* is always 1 for analog voice ports in the Cisco MC3810 series. • *port* specifies an analog voice port number. Valid entries are 1 to 6.

Syntax Description

slot:ds0-group	Debugs the digital voice port you specify with the *slot:ds0-group* designation. • *slot* specifies the module (and controller). Valid entries are 0 for the MFT (controller 0) and 1 for the DVM (controller 1). • *ds0-group* specifies a T1 or E1 logical voice port number. Valid entries are 0 to 23 for T1 and 0 to 30 for E1.

Command Default

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, and 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** command 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.
payload-type	(Optional) Valid payload types are as follows: • **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.

codec	(Optional) If a codec needs to be specified for the payload type, valid codecs are as follows: • **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.

Command Default

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, and 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.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was introduced.

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.

Command Default

Disabled

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 implemented on the Cisco 2600, Cisco 3600 and Cisco MC3810 series.
12.2(11)T	The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, and 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 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 digital signal processor (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.

Command Default

Disabled

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 implemented on the Cisco 2600, Cisco 3600 and Cisco MC3810 series.
12.2(11)T	The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, and 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.

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 tone

To display debugging 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 arguments or keywords.

Command Default

Disabled

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, and 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.

Command Default

Disabled

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(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, and 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 vwic-mft firmware controller

To display debug output from the multiflex (MFT) Voice/WAN interface card (VWIC) controller firmware, use the **debug vwic-mft firmware controller** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vwic-mft firmware controller {**t1**| **e1**} *slot*/*port* {**alarm**| **all**| **config**| **fdl**| **loopback**| **register display**| **status**}

no debug vwic-mft firmware controller {**t1**| **e1**} *slot*/*port* {**alarm**| **all**| **config**| **fdl**| **loopback**| **register display**| **status**}

Syntax Description

t1	Displays debugging messages for T1 channels.
e1	Displays debugging messages for E1 channels.
slot	Slot number. Refer to the appropriate hardware manual for slot information.
port	Port number. Refer to the appropriate hardware manual for port information. The slash mark is required between the *slot* argument and the *port*argument.
alarm	Displays firmware alarm messages.
all	Displays all debugging messages about the MFT VWIC.
config	Displays firmware output messages about configuration change messages sent by the Cisco IOS software.
fdl	Displays firmware output messages when select facilities data link (FDL) events occur.
loopback	Displays firmware output messages when select loopback events occur.
register display	Displays a full framer register value table.
status	Displays current attributes enabled for the specified controller.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(6)	This command was introduced.
12.3(7)T	This command was integrated into Cisco IOS Release 12.3(7)T.

Usage Guidelines

Use the **debug vwic-mft firmware controller** command in privileged EXEC mode to provide firmware-level information for VWICs when information is required beyond the Cisco IOS T1 and E1 controller statistics. The physical-layer information generated by this command includes alarm conditions, line status, controller issues, and register settings, all of which can be used to help troubleshoot MFT VWIC problems.

All the debugging keywords, except **register display**, enable debugging on both ports of a 2-port card. For example, if T1 0/0 and T1 0/1 are two ports on a 2-port MFT card and any of the keywords except **register display** is enabled, debugging output will be generated for both ports because they share a common firmware system.

The Cisco 1- and 2-port T1/E1 multiflex VWICs support voice and data applications in Cisco 2600, Cisco 3600, and Cisco 3700 series multiservice routers. The multiflex VWIC combines WAN interface card and voice interface card functionality.

Caution Use any debugging command with caution because the volume of output generated can slow or stop the router operations. We recommend that this command be used only under the supervision of a Cisco engineer.

Examples

The following sample output displays firmware output about alarm messages for an MFT VWIC installed in slot 0.

```
Router# debug vwic-mft firmware controller e1 0/0 alarm

vwic-mft firmware output messages for wic slot set to: Alarm
Router#
*Mar  4 13:58:14.702: E1T1 0/1  FW: alm1:0e p:01 ALOS LOS LOF
*Mar  4 13:58:15.194: E1T1 0/1  FW:  CERR: 00
*Mar  4 13:58:15.194: E1T1 0/1  FW:  MERR: 00
*Mar  4 13:58:15.194: E1T1 0/1  FW:  FERR: 00
```

Note The output will vary depending on what the router is configured to do after the **debug** command is entered.

The table below describes the significant fields shown in the display.

Table 11: debug vwic-mft firmware controller alarm Field Descriptions

Field	Description
vwic-mft firmware output messages for wic slot set to	Acknowledges that the command has been entered and indicates the current state.

Field	Description
*Mar 4 13:58:14.702: E1T1 0/1 FW	Time-stamp preface that shows that this is a firmware (FW) message. **Note** The port numbers reported here may differ from the numbers configured using the Cisco IOS software because the error is being reported from the second port where debugging has been enabled by the **alarm** keyword on a 2-port MFT card.
alm1:0e	Actual value of the alarm status register.
p:01	Port number of the local VWIC port that is reporting the condition. Value is either 0 or 1 for each port. **Note** The output shows two port numbers; this is an example of the debugging being enabled for both ports on a 2-port MFT card.
ALOS LOS LOF	Shorthand value of current alarm conditions defined in the register. One of the following: • AIS--Receive Alarm Indication Signal • ALOS--Receive Analog Loss of Signal • LOF--Receive Loss of Frame Alignment • LOS--Receive Loss of Signal • MYEL--Receive Multiframe Yellow Alarm • YEL--Receive Yellow Alarm Register value showing the actual value of the alarm status register.
CERR	Status of the error status register; cyclical redundancy check (CRC) block error.
MERR	Status of the error status register; multiframe alignment signal (MFAS) pattern error (E1 only).
FERR	Status of the error status register; framing error.

Related Commands

show controllers e1	Displays information about E1 links.
show controllers t1	Displays information about T1 links.

debug vxml

Note Effective with release 12.3(8)T, the **debug vxml**command is replaced by the **debug voip application vxml**command. See the **debug voip application vxml**command for more information.

To display debugging messages for VoiceXML features, use the **debug vxml** command in privileged EXEC mode. To disable 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.

Command Default

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.
12.3(8)T	This command was replaced by the **debug voip application vxml**command.

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#
1w5d: //-1//VAPP:/vapp_get_apphandler:
1w5d: vapp_get_apphandler: Script callme
1w5d: //-1//VAPP:/vapp_get_apphandler_core:
1w5d: //-1/000000000000/VAPP:/vapp_InterpInitConfigParams:
1w5d: //-1/000000000000/VAPP:/vapp_evt_handler: State VAPP_ACTIVE got event CC_D
1w5d: //-1/000000000000/VAPP:/vapp_driver: pInterp[660E10FC]:
1w5d: //-1/000000000000/VAPP:/vapp_driver: evtID: 28 vapp record state: 0
1w5d: //-1/000000000000/VAPP:/vapp_evt_setup:
1w5d: //-1//VAPP:/vapp_incoming_cal
doc-rtr54-01#lblock:
1w5d: vapp_incoming_callblock:
1w5d: //39/924083218026/VAPP:/vapp_load_or_run_script:
1w5d: //39/924083218026/VAPP:/vapp_load_or_run_script:
1w5d: 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
1w5d: //39/924083218026/VAPP:/vapp_media_play:
1w5d: //39/
Router#924083218026/VAPP:/vapp_media_play: prompt=flash:welcome_test.au:
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]:
1w5d: //39/924083218026/VAPP:/vapp_driver: evtID: 36 vapp record state: 0
1w5d: //39/924083218026/VAPP:/vapp_checksessionstate:
1w5d: //39/924083218026/VAPP:/vapp_checkifdo
Router#ne:
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_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: //39/9240832180 26/VAPP:/vapp_checksessionstate:
1w5d: //39/9240832180 26/VAPP:/vapp_checkifdone:
1w5d: //39/9240832180 26/VAPP:/vapp_evt_handler: State VAPP_CLEANING got event AE
1w5d: //39/9240832180 26/VAPP:/vapp_cleaner:
1w5d: //39/9240832180 26/VAPP:/vapp_cleaner: VxmlDialogDone event=vxml.session.c0
1w5d: //39/9240832180 26/VAPP:/vapp_popifdone:
1w5d: //39/9240832180 26/VAPP:/vapp_checkifdone:
1w5d: //39/9240832180 26/VAPP:/vapp_
Router#cleanup_apphandler:
1w5d: vapp_cleanup_apphandler: Terminate FALSE Terminated TRUE{HAN[VXML_HAN][NU}
1w5d: //39/9240832180 26/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.vxml
 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
vapphandle=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:/vxm
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
1w5d: //51/359408288031/VXML:/vxml_prompt_proc:
1w5d:    <prompt>: bargein=1 count=1 typeaheadflush=0
1w5d: //51/359408288031/VXML:/vxml_audio_proc:
1w5d:     <audio>: URI(abs):flash:enter_dest.au scheme=flash path=enter_dest.au0
1w5d: //51/359408288031/VXML:/vxml_vapp_media_play: bargein=1 timeout=0 typeahe0
1w5d: //51/359408288031/VXML:/vxml_vapp_media_play:
1w5d: //51/359408288031/VXML:/vxml_vapp_media_play: audio
Router#=flash:enter_dest.au cachable=1 timeout20
1w5d: //51/359408288031/VXML:/vxml_vapp_vcr_control_disable:
1w5d: //51/359408288031/VXML:/vxml_vapp_digit_collect: termchar # maxDigits 0 t0
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(abs1
Router#.0
1w5d: //51/359408288031/VXML:/vxml_field_proc:
1w5d:   <field>: type=number
1w5d: //51/359408288031/VXML:/vxml_item_attrs_proc:  name=get_phone_num modal=a2
1w5d: //51/359408288031/VXML:/vxml_filled_proc:
1w5d:
1w5d: <filled>: mode=all
1w5d: //51/359408288031/VXML:/vxml_assign_proc:
1w5d:    <assign>: namep=phone_num expr=get_phone_num
1w5d: //51/359408288031/VXML:/vxml_goto_proc:
1w5d:    <goto>: caching=fast fetchhint=invalid fetchtimeout=20 URI:#transfer_mm
Router#entp=transfer_me
1w5d:    vxml_dialog_reset:
1w5d: //51/359408288031/VXML:/vxml_leave_scope: scope=110
1w5d: //51/359408288031/VXML:/vxml_leave_scope: scope=8
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_form_proc:
1w5d:  <form>: id=transfer_me   scope=dialog
1w5d: vxml_form_init current scope: dialog
1w5d:  <var>: namep=myd
Router#ur
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_transfer_proc:
1w5d:   <transfer>:
1w5d: //51/359408288031/VXML:/vxml_item_attrs_proc:  name=mycall dest_expr='phoe
Router#ctreason=-1
1w5d: //51/359408288031/VXML:/vxml_vapp_placecall: dest 5551234 timeout 15 max10
1w5d: //51/359408288031/VXML:/vxml_vapp_gain_control_default:
1w5d: //51/359408288031/VXML:/vxml_expr_eval: expr=dialog.mycall = 'far_end_dis'
1w5d: //51/359408288031/VXML:/vxml_expr_eval: expr=dialog.mycall$.duration = 2
1w5d: //51/359408288031/VXML:/vxml_start: vxmlhandle=65350A7C vapphandle=660E100
1w5d: //51/359408288031/VXML:/vxml_vxml
Router#_proc:
1w5d: <vxml> URI(abs):flash:call.vxml scheme=flash path=call.vxml base= URI(abs0
1w5d: //51/359408288031/VXML:/vxml_transfer_proc:
1w5d:   <transfer>:
1w5d: //51/359408288031/VXML:/vxml_item_attrs_proc:  name=mycall URI(abs):phone-
Router#1, redirectreason=-1
1w5d: //51/359408288031/VXML:/vxml_form_proc:
1w5d:  <form>: id=transfer_me   scope=dialog
1w5d: //51/359408288031/VXML:/vxml_filled_proc:
1w5d:
1w5d: <filled>: mode=all
1w5d: //51/359408288031/VXML:/vxml_assign_proc:
1w5d:   <assign>: namep=mydur expr=mycall$.duration
1w5d: //51/359408288031/VXML:/vxml_if_proc:
1w5d:   <if>: cond=mycall == 'busy'
1w5d: //51/359408288031/VXML:/vxml_leave_scope: scope=8
1w5d: //51/359408288031/VXML:/vxml_formitem_select: Status=VXML_ST
Router#ATUS_OK,
1w5d: //51/359408288031/VXML:/vxml_formitem_select:  AsyncStatus=VXML_STATUS_OK
1w5d: //51/359408288031/VXML:/vxml_formitem_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.

debug waas

To enable debugging for WAAS Express modules, use the **debug waas** command in privileged EXEC mode. To disable WAAS Express debugging, use the **no** form of this command.

debug waas {{**auto-discovery**| **aoim**| **cce**| **infrastructure**| **lz**| **memory**| **tfo**} {**events**| **errors**| **operations**}| **api**| **mibs**| **dre** {**events**| **errors**| **operations [brief]**| **uplink**}| **management** {**events**| **errors**}}

no debug waas {{**auto-discovery**| **aoim**| **cce**| **infrastructure**| **lz**| **memory**| **tfo**} {**events**| **errors**| **operations**}| **api**| **mibs**| **dre** {**events**| **errors**| **operations [brief]**| **uplink**}| **management** {**events**| **errors**}}

Syntax Description

auto-discovery	Enables debugging for WAAS Express autodiscovery information.
aoim	Enables debugging for peer information and negotiated capabilities information.
cce	Enables debugging for Common Classification Engine (CCE).
infrastructure	Enables debugging for WAAS Express infrastructure.
lz	Enables debugging for Lempel-Ziv (LZ) optimization.
memory	Enables debugging for WAAS Express internal memory usage.
tfo	Enables debugging for Transport Flow Optimization (TFO).
events	Enables debugging for WAAS Express events.
errors	Enables debugging for WAAS Express errors.
operations	Enables debugging for WAAS Express operations.
brief	Displays WAAS connection operations in brief.
api	Enables debugging for WAAS Express public application programming interfaces (APIs).
mibs	Enables debugging for WAAS Express MIBs.
dre	Enables debugging for Data Redundancy Elimination (DRE) optimization.
uplink	Enables debugging for DRE upload.
management	Enables debugging for error and event management.

Command Default

Debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(2)T	This command was introduced.
15.2(3)T	This command was modified. The **api** and **mibs** keywords were added, and the **brief** keyword was removed.

Examples

The following example shows how to enable debugging output for WAAS Express infrastructure operations:

```
Device> enable
Device# debug waas infrastructure operations
```

Related Commands

Command	Description
clear waas	Clears WAAS Express statistics and closed connections information.
show waas alarms	Displays WAAS Express status and alarms.
show waas auto-discovery	Displays information about WAAS Express autodiscovery.
show waas connection	Displays information about WAAS Express connections.
show waas statistics aoim	Displays WAAS Express peer information and negotiated capabilities.
show waas statistics application	Displays WAAS Express policy application statistics.
show waas statistics auto-discovery	Displays WAAS Express autodiscovery statistics.
show waas statistics class	Displays statistics for the WAAS Express class map.
show waas statistics dre	Displays WAAS Express DRE statistics.
show waas statistics errors	Displays WAAS Express error statistics.
show waas statistics global	Displays global WAAS Express statistics.
show waas statistics lz	Displays WAAS Express LZ statistics.
show waas statistics pass-through	Displays WAAS Express connections placed in a pass-through mode.

Command	Description
show waas statistics peer	Displays inbound and outbound statistics for peer WAAS Express devices.
show waas status	Displays the status of WAAS Express.
show waas token	Displays the value of the configuration token used by the WAAS Central Manager.
waas cm-register url	Registers a device with the WAAS Central Manager.

debug waas accelerator cifs-express

To enable debugging for the Common Internet File System (CIFS)-Express accelerator module of WAAS Express, use the **debug waas accelerator cifs-express** command in privileged EXEC mode. To disable CIFS-Express accelerator debugging, use the **no** form of this command.

debug waas accelerator cifs-express [**ads-negative-cache**| **async-write**| **infra**| **read-ahead**] {**debug**| **events**| **errors**| **file remote-file** *file-URL*| **operations**}

no debug waas accelerator cifs-express [**ads-negative-cache**| **async-write**| **infra**| **read-ahead**] {**debug**| **events**| **errors**| **file remote-file** *file-URL*| **operations**}

Syntax Description

ads-negative-cache	(Optional) Enables debugging of alternate data stream negative caching.
async-write	(Optional) Enables debugging of async write operations.
infra	(Optional) Enables debugging of CIFS-Express accelerator infrastructure.
read-ahead	(Optional) Enables debugging of read ahead operations.
debug	Enables debugging of a specific CIFS-Express parameter, such as async write or read ahead.
events	Enables debugging of CIFS-Express parameter events.
errors	Enables debugging of CIFS-Express parameter errors.
file remote-file *file-URL*	Enables debugging of the CIFS-Express accelerator log file. The format to specify the file URL is *ftp://user:pass@remote_ip/filepathname* and can have up to 500 characters.
operations	Enables debugging of CIFS-Express parameter operations.

Command Default

CIFS-Express accelerator debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(3)T	This command was introduced.

Examples

The following example shows how to enable debugging of CIFS-Express accelerator read ahead errors:

```
Device> enable
Device# debug waas accelerator cifs-express read-ahead errors
```

Related Commands

Command	Description
accelerator	Enters a specific WAAS Express accelerator configuration mode based on the accelerator being configured.
debug waas	Enables debugging for WAAS Express modules.
debug waas accelerator http-express	Enables debugging for the HTTP-Express accelerator module of WAAS Express.
debug waas accelerator ssl-express	Enables debugging for the SSL-Express accelerator module of WAAS Express.
show waas accelerator	Displays information about WAAS Express accelerators.

debug waas accelerator http-express

To enable debugging for the HTTP-Express accelerator module of WAAS Express, use the **debug waas accelerator http-express** command in privileged EXEC mode. To disable HTTP-Express accelerator debugging, use the **no** form of this command.

debug waas accelerator http-express {**infrastructure**| **metadatacache**| **parser**| **transaction**} {**events**| **errors**| **operations**}

no debug waas accelerator http-express {**infrastructure**| **metadatacache**| **parser**| **transaction**} {**events**| **errors**| **operations**}

Syntax Description

infrastructure	Enables debugging of HTTP-Express accelerator infrastructure.
metadatacache	Enables debugging of HTTP metadata cache.
parser	Enables debugging of the HTTP-Express accelerator parser.
transaction	Enables debugging of HTTP-Express accelerator transactions.
events	Enables debugging of HTTP-Express parameter events.
errors	Enables debugging of HTTP-Express parameter errors.
operations	Enables debugging of HTTP-Express parameter operations.

Command Default

HTTP-Express accelerator debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(3)T	This command was introduced.

Examples

The following example shows how to enable debugging of HTTP-Express accelerator parser events:

```
Device> enable
Device(config)# debug waas accelerator http-express parser events
```

Related Commands

Command	Description
accelerator	Enters a specific WAAS Express accelerator configuration mode based on the accelerator being configured.
debug waas	Enables debugging for WAAS Express modules.
debug waas accelerator cifs-express	Enables debugging for the CIFS-Express accelerator module of WAAS Express.
debug waas accelerator ssl-express	Enables debugging for the SSL-Express accelerator module of WAAS Express.
show waas accelerator	Displays information about WAAS Express accelerators.

debug waas accelerator ssl-express

To enable debugging for the Secure Sockets Layer (SSL)-Express accelerator module of WAAS Express, use the **debug waas accelerator ssl-express** command in privileged EXEC mode. To disable SSL-Express accelerator debugging, use the **no** form of this command.

debug waas accelerator ssl-express {**events**| **errors**| **operations**| **messages**}

no debug waas accelerator ssl-express {**events**| **errors**| **operations**| **messages**}

Syntax Description

events	Enables debugging of SSL-Express events.
errors	Enables debugging of SSL-Express errors.
operations	Enables debugging of SSL-Express operations.
messages	Enables debugging of SSL protocol messages.

Command Default

Debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(3)T	This command was introduced.

Examples

The following example shows how to enable debugging of SSL-Express accelerator operations:

```
Device> enable
Device(config)# debug waas accelerator ssl-express operations
```

Related Commands

Command	Description
accelerator	Enters a specific WAAS Express accelerator configuration mode based on the accelerator being configured.
debug waas	Enables debugging for WAAS Express modules.
debug waas accelerator cifs-express	Enables debugging for the CIFS-Express accelerator module of WAAS Express.

Command	Description
debug waas accelerator http-express	Enables debugging for the HTTP-Express accelerator module of WAAS Express.
show waas accelerator	Displays information about WAAS Express accelerators.

debug warm-reboot

To display warm reload debug information, use the **debug warm-reboot** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug warm-reboot

no debug warm-reboot

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.

Examples

The following is sample output from the **reload warm file** *url* command when the **debug warm-reboot** command is enabled:

```
Router# debug warm-reboot
Router# reload warm file tftp://9.1.0.1/c7200-p-mz.port
Proceed with reload? [confirm]
Loading c7200-p-mz.port from 9.1.0.1 (via Ethernet5/0):!!!
00:05:43:ptr   :63B978E0
00:05:43:magic :A457272
00:05:43:ptr   :63B98020
00:05:43:magic :0
00:05:43:ptr   :63B98380
00:05:43:magic :0
00:05:43:ptr   :63B983A0
00:05:43:magic :FEEDFACE
00:05:43:uncomp_size         :2749E7C
00:05:43:comp_size           :E966F0
00:05:43:comp_checksum       :9BB36053
00:05:43:uncomp_checksum     :56F1754B!!!
[OK - 15323964 bytes]
Decompressing the image :###
00:06:22:Image checksum correct -1#682743213
00:06:22:Compressed Image checksum correct### [OK]
Number 0     source 0x63BD17C4
Number 1     source 0x63C43AD0
Number 2     source 0x63C83AFC
Number 3     source 0x63CC3B28
.
.
.
Number 156   source 0x66384074
Number 157   source 0x663C40A0
Number 158   source 0x664040CC
wrb_copy_and_launch location = 0x664040CC
00:06:39:Found elf header at the expected location
00:06:39:Source elf_hdr->e_shnum = A
00:06:39:Setting up to copy ELF section 1
00:06:39: to image_info section 0
```

```
00:06:39: sh_name = B
00:06:39: sh_type = 1
00:06:39: sh_flags = 7
00:06:39: sh_addr = 80008000
00:06:39: sh_offset = 60
00:06:39: sh_size = 186C000
00:06:39: sh_link = 0
00:06:39: sh_info = 0
00:06:39: sh_addralign = 20
00:06:39: sh_entsize = 0
.
.
.
00:06:40:Setting up to copy ELF section 4
00:06:40: to image_info section A0
00:06:40: sh_name = 1F
00:06:40: sh_type = 1
00:06:40: sh_flags = 10000003
00:06:40: sh_addr = 82750380
00:06:40: sh_offset = 27483E0
00:06:40: sh_size = 18A0
00:06:40: sh_link = 0
00:06:40: sh_info = 0
00:06:40: sh_addralign = 10
00:06:40: sh_entsize = 0
00:06:40:cpu type                   :19
00:06:40:image_info->entry_point   = 80008000
00:06:40:image_info->section_count = A1
00:06:40:image_info->monstack      = 80007FC0
00:06:40:image_info->monra         = BFC014E4
00:06:40:image_info->param0        = 2
00:06:40:image_info->param1        = 0
00:06:40:image_info->param2        = 80005998
00:06:40:image_info->param3        = 80008000
00:06:40:Section
00:06:40:Section
Decompressed Image checksum correct
             Restricted Rights Legend
.
.
.
```

debug wccp

To display information about all (IPv4 and IPv6) Web Cache Communication Protocol (WCCP) services, use the **debug wccp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug wccp {**default**| **vrf** *vrf-name* {**events**| **packets [control]**}| **events**| **packets** [**bypass**| **control**| **redirect**]| **platform**| **subblocks**}

no debug wccp {**default**| **vrf** *vrf-name* {**events**| **packets [control]**}| **events**| **packets** [**bypass**| **control**| **redirect**]| **platform**| **subblocks**}

Syntax Description

default	Displays information about default WCCP services.
vrf *vrf-name*	Specifies a virtual routing and forwarding (VRF) instance to associate with a service group.
events	Displays information about significant WCCP events.
packets	Displays information about every WCCP packet received or sent by the router.
control	(Optional) Displays information about WCCP control packets.
bypass	(Optional) Displays information about WCCP bypass packets.
redirect	(Optional) Displays information about WCCP redirect packets.
platform	Displays information about the WCCP platform application programming interface (API).
subblocks	Displays information about WCCP subblocks.

Command Default

Debug information is not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(3)T	This command was introduced.

Release	Modification
15.1(1)SY1	This command was integrated into Cisco IOS Release 15.1(1)SY1.

Usage Guidelines

When the **vrf** keyword is not used, the command displays debug information about all WCCP services on the router. The **default** keyword is used to specify default WCCP services.

Examples

The following is sample output from the **debug wccp events** command when a Cisco Cache Engine is added to the list of available Web caches:

```
Router# debug wccp events
WCCP-EVNT: Built I_See_You msg body w/1 usable web caches, change # 0000000A
WCCP-EVNT: Web Cache 192.168.25.3 added
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000B
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000C
```

The following is sample output from the **debug wccp packets** command. The router is sending keepalive packets to the Cisco Cache Engines at 192.168.25.4 and 192.168.25.3. Each keepalive packet has an identification number associated with it. When the Cisco Cache Engine receives a keepalive packet from the router, it sends a reply with the identification number back to the router.

```
Router# debug wccp packets
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003532
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003534
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003533
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003535
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003534
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003536
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003535
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003537
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003536
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003538
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003537
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003539
```

Related Commands

Command	Description
clear wccp	Clears the counter for packets redirected using WCCP.
ip wccp	Enables support of the specified WCCP service for participation in a service group.
ipv6 wccp	Enables support of the specified WCCP service for participation in a service group.
ip wccp redirect	Enables packet redirection on an outbound or inbound interface using WCCP.
ipv6 wccp redirect	Enables packet redirection on an outbound or inbound interface using WCCP.

Command	Description
show ip interface	Lists a summary of the IP information and status of an interface.
show ipv6 interface	Lists a summary of the IP information and status of an interface.

debug webvpn

To enable the display of debug information for SSL VPN applications and network activity, use the **debug webvpn**command in privileged EXEC mode. To stop debugging messages from being processed and displayed, use the **no** form of this command.

debug webvpn [verbose] [**aaa**| **acl**| **cifs**| **citrix [verbose]**| **cookie [verbose]**| **count**| **csd**| **data**| **dns**| **emweb [state]**| **entry** *context-name* [**source** *ip* [*network-mask*]| **user** *username*]| **http** [**authentication**| **trace**| **verbose**]| **package**| **sdps** [**level** *number*]| **sock [flow]**| **sso**| **timer**| **trie**| **tunnel** [**traffic** *acl-number*| **verbose**]| **url-disp**| **webservice [verbose]**]

no debug webvpn [verbose] [**aaa**| **acl**| **cifs**| **citrix [verbose]**| **cookie [verbose]**| **count**| **csd**| **data**| **dns**| **emweb [state]**| **entry** *context-name* [**source** *ip* [*network-mask*]| **user** *username*]| **http** [**authentication**| **trace**| **verbose**]| **package**| **sdps** [**level** *number*]| **sock [flow]**| **sso**| **timer**| **trie**| **tunnel** [**traffic** *acl-number*| **verbose**]| **url-disp**| **webservice [verbose]**]

Syntax Description

verbose	(Optional) Detailed information about SSL VPN applications and network activity is displayed in addition to the nondetailed information.
aaa	(Optional) Displays authentication, authorization, and accounting (AAA) event and error messages.
acl	(Optional) Displays information about the Application Layer access control list (ACL).
cifs	(Optional) Displays Microsoft Windows file share access event and error messages.
citrix [**verbose**	(Optional) Displays Citrix application event and error messages. • **verbose** (Optional)--All detailed and nondetailed citrix messages are displayed. If the **verbose** keyword is not used, only the nondetailed messages are displayed.
cookie [**verbose**	(Optional) Displays event and error messages that relate to the cookie that is pushed to the browser of the end user. • **verbose** (Optional)--All detailed and nondetailed cookie messages are displayed. If the **verbose** keyword is not used, only the nondetailed messages are displayed.
count	(Optional) Displays reference count information for a context.

csd	(Optional) Displays Cisco Secure Desktop (CSD) event and error messages.
data	(Optional) Displays data debug messages.
dns	(Optional) Displays domain name system (DNS) event and error messages.
emweb [**state**	(Optional) Displays emweb state debug messages.
entry *context-name* [**source** *ip* [*network-mask*] | **user** *username*	(Optional) Displays information for a specific user or group. • *context-name*-- SSL VPN context name. • **source** *ip* (Optional)--IP address of the user or group. The *network-mask*argument is optional. If not specified, 255.255.255.255 is used. • **user** *username* (Optional)-- Username of the user. **Note** The **entry** keyword can be used with other **debug** commands to single out the debug messages for a particular user or group. If the **debug webvpn** entry is not defined, the debug messages of the feature or function that are turned on are printed for every user.
http [**authentication**| **trace**| **verbose**	(Optional) Displays HTTP debug messages. • **authentication** (Optional)--Displays information for HTTP authentication, such as NT LAN Manager (NTLM). • **trace** (Optional)--Displays HTTP information that involves EmWeb processing. • **verbose** (Optional)--All detailed and nondetailed HTTP messages are displayed. If the **verbose** keyword is not used, only the nondetailed messages are displayed.
package	(Optional) Deploys event and error messages for the software packages that are pushed to the end user.
sdps [**level** *number*]	(Optional) Displays SDPS debug messages. The level is entered as a number from 1 to 5.
sock [**flow**]	(Optional) Displays socket debug messages.

sso	(Optional) Displays information about Single SignOn (SSO) ticket creation, session setup, and response handling.
timer	(Optional) Displays timer debug messages.
trie	(Optional) Displays trie debug messages.
tunnel [**traffic** *acl-number* | **verbose**]	(Optional) Displays tunnel debug messages. • **traffic** *acl-number* (Optional)--Access control list number of the traffic to be displayed. • **verbose** (Optional)--All detailed and nondetailed tunnel messages are displayed. If the **verbose** keyword is not used, only the nondetailed messages are displayed.
url-disp	(Optional) Displays URL debug messages.
webservice [**verbose**]	(Optional) Displays web service event and error messages. • **verbose** (Optional)--All detailed and nondetailed web service messages are displayed. If the **verbose** keyword is not used, only the nondetailed messages are displayed.

Command Default

None

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.
12.4(6)T	Support for the SSL VPN enhancements feature was added.

Release	Modification
12.4(11)T	The following keywords were deleted effective with Cisco IOS Release 12.4(11)T: • **port-forward** • **detail** keyword option for the **tunnel** keyword The following keywords and arguments were added effective with Cisco IOS Release 12.4(11)T: • **verbose** • **acl** • **entry** *context-name* [**source** *ip* [*network-mask*] | **user** *username* • **authentication** , **trace**, and **verbose** keyword options for the **http** keyword • **sso** • **verbose** keyword option for the **citrix**, **cookie**, **tunnel**, and **webservice** keywords

Usage Guidelines

This command should be used with caution on a production router or networking device. It is recommended that debugging is enabled only for individual components as necessary. This restriction is intended to prevent the console session from be overwhelmed by large numbers of messages.

The **no** form of this command turns off feature debugging. It does not matter if the **verbose** keyword has been used or not.

If the **no** form of this command is used with the **verbose** keyword option for any keyword, all keyword and argument fields must be an exact match.

Examples

Examples

The following example displays **debug webvpn** output for various SSL VPN sessions:

```
Router# debug webvpn
*Dec 23 07:47:41.368: WV: Entering APPL with Context: 0x64C5F270,
      Data buffer(buffer: 0x64C877D0, data: 0x4F27B638, len: 272,
      offset: 0, domain: 0)
*Dec 23 07:47:41.368: WV: http request: /sslvpn with domain cookie
*Dec 23 07:47:41.368: WV: Client side Chunk data written..
 buffer=0x64C877B0 total_len=189 bytes=189 tcb=0x6442FCE0
*Dec 23 07:47:41.368: WV: sslvpn process rcvd context queue event
*Dec 23 07:47:41.372: WV: sslvpn process rcvd context queue event
*Dec 23 07:47:41.372: WV: Entering APPL with Context: 0x64C5F270,
      Data buffer(buffer: 0x64C877D0, data: 0x4F26D018, len: 277,
      offset: 0, domain: 0)
*Dec 23 07:47:41.372: WV: http request: /webvpn.html with domain cookie
*Dec 23 07:47:41.372: WV: [Q]Client side Chunk data written..
 buffer=0x64C877B0 total_len=2033 bytes=2033 tcb=0x6442FCE0
*Dec 23 07:47:41.372: WV: Client side Chunk data written..
 buffer=0x64C87710 total_len=1117 bytes=1117 tcb=0x6442FCE0
```

Examples

The following example displays information for a specific user (user1 under the context “mycontext”) and for a feature or function:

```
Router# debug webvpn entry mycontext_user_user1
! The above line turns debugging on for user1.
! The following line turns on debugging for a feature (or features) or function (or
functions)--in this case; for authentication, authorization, and accounting (AAA).
Router# debug webvpn aaa
```

The actual output is as follows:

```
*Dec 23 07:56:41.351: WV-AAA: AAA authentication request sent for user: "user1"
*Dec 23 07:56:41.351: WV-AAA: AAA Authentication Passed!
*Dec 23 07:56:41.351: WV-AAA: User "user1" has logged in from "10.107.163.147" to gateway
"sslvpn" context "mycontext"
*Dec 23 07:59:01.535: WV-AAA: User "user1" has logged out from gateway "sslvpn" context
"mycontext"
```

Examples

The following example displays cookie and HTTP information for a group of users under the context “mycontext” having a source IP range from 192.168.1.1. to 192.168.1.255:

```
Router# debug webvpn entry mycontext source 192.168.1.0 255.255.255.0
! The above command line sets up debugging for the group.
!The following command lines turn on debugging for cookie and HTTP information.
Router# debug webvpn cookie
Router# debug webvpn http
```

The actual output is as follows:

```
*Dec 23 08:10:11.191: WV-HTTP: Original client request
GET /webvpn.html HTTP/1.1

*Dec 23 08:10:11.191: WV-HTTP: HTTP Header parsing complete
*Dec 23 08:10:11.191: WV-HTTP: * HTTP request complete
*Dec 23 08:10:11.191: WV-COOKIE: Enter VW context cookie check with Context:0x64C5F470,
          buffer: 0x64C87710, buffer->data: 0x4F26D018, buffer->len: 277,
          cookie: 0x4F26D10A, length: 33
*Dec 23 08:10:11.191: WV-COOKIE: webvpn context cookie received is webvpncontext=00@mycontext

*Dec 23 08:10:11.191: WV-COOKIE: context portion in context cookie is: mycontext

*Dec 23 08:10:11.327: WV-HTTP: Original client request
GET /paramdef.js HTTP/1.1

*Dec 23 08:10:11.327: WV-HTTP: HTTP Header parsing complete
*Dec 23 08:10:11.327: WV-HTTP: * HTTP request complete
```

Examples

The following output example displays information about SSO ticket creation, session setup, and response handling:

```
Router# debug webvpn sso
*Jun 12 20:37:01.052: WV-SSO: Redirect to SSO web agent URL -
http://example.examplecompany.com/vpnauth/
*Jun 12 20:37:01.052: WV_SSO: Set session cookie with SSO redirect
*Jun 12 20:37:01.056: WV-SSO: Set SSO auth flag
*Jun 12 20:37:01.056: WV-SSO: Attach credentials - building auth ticket
*Jun 12 20:37:01.060: WV-SSO: user: [user11], secret: [example123], version: [1.0], login
time: [BCEFC86D], session key: [C077F97A], SHA1 hash :
[B07D0A924DB33988D423AE9F937C1C5A66404819]
*Jun 12 20:37:01.060: WV-SSO: auth_ticket :
user11:1.0@C077F97A@BCEFC86D@B07D0A924DB33988D423AE9F937C1C5A66404819
*Jun 12 20:37:01.060: WV-SSO: Base64 credentials for the auth_ticket:
```

```
dXNlcjExOjEuMEBDMDc3Rjk3QUBCQ0VGQzg2REBCMDdEMEE5MjREQjMzOTg4RDQyM0FFOUY5MzdDMUM1QTY2NDA0ODE5
*Jun 12 20:37:01.060: WV-SSO: Decoded credentials =
user11:1.0@C077F97A@BCEFC86D@B07D0A924DB33988D423AE9F937C1C5A66404819
*Jun 12 20:37:01.060: WV-SSO: Starting SSO request timer for 15-second
*Jun 12 20:37:01.572: WV-SSO: SSO auth response rcvd - status[200]
*Jun 12 20:37:01.572: WV-SSO: Parsed non-SM cookie: SMCHALLENGE
*Jun 12 20:37:01.576: WV-SSO: Parsed SMSESSION cookie
*Jun 12 20:37:01.576: WV-SSO: Sending logon page after SSO auth success
```

debug webvpn dtls

To enable the display of Secure Socket Layer Virtual Private Network (SSL VPN) Datagram Transport Layer Security (DTLS) debug information, use the **debug webvpn dtls** command in privileged EXEC mode. To stop debugging messages from being processed and displayed, use the **no** form of this command.

debug webvpn dtls [**errors**| **events**| **packets**]

no debug webvpn dtls [**errors**| **events**| **packets**]

Syntax Description

errors	(Optional) Displays errors that might have occurred while setting up DTLS tunnel or during data transfer.
events	(Optional) Displays DTLS event messages. Displays events like encryption, decryption, switching, and so on.
packets	(Optional) Displays DTLS packet dump.

Command Default

If no keyword is specified, then all the SSL VPN DTLS debug information displays are enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(2)T	This command was introduced.

Usage Guidelines

You can use the **debug webvpn dtls** command to debug any issues related to WebVPN DTLS.

This debug information provides information about the packets that are being processed by WebVPN DTLS and indicates if there are any errors.

Examples

The following example displays the SSL VPN DTLS packet dump information:

```
Router# debug webvpn dtls packets

*Jun 15 10:23:04.495: WV-DTLS: pak (0x67EEF474), dgram (109), length (0) encsize(0)
2E6FBD10:                   17010000 01000000          ........
2E6FBD20: 00004C00 6057A7E2 399F19CF 9915D3F4  ..L.`W'b9..O..St
2E6FBD30: 4FBA7F24 8AEC4EFC 9F4192B5 D334F471  O:.$.lN|.A.5S4tq
2E6FBD40: 02232ADF BB248C8B 54E197F5 713D7886  .#*_;$..Ta.uq=x.
2E6FBD50: 4F71398D 993342BA 90D2A677 96A6ABB9  Oq9..3B:.R&w.&+9
2E6FBD60: 8B72F19C 4D454CBB A74D2342 B643FA74  .rq.MEL;'M#B6Czt
2E6FBD70: A627656A E1DDF0A9 ABDAC6FC 7986FC52  &'eja]p)+ZF|y.|R
```

```
2E6FBD80: AD9AF67D C5                           -.v}E
*Jun 15 10:23:04.499: WV-DTLS: pak (0x67EEB7A8), dgram (137), length (0) encsize(0)
2E6FA4D0:                   45000089 FCE80000          E...|h..
2E6FA4E0: FF11761F 1E010132 28010128 01BB0CBA  ..v....2(..(.;.:
2E6FA4F0: 0075ECF8 17010000 01000000 00004C00  .ulx..........L.
2E6FA500: 6057A7E2 399F19CF 9915D3F4 4FBA7F24  `W'b9..O..StO:.$
2E6FA510: 8AEC4EFC 9F4192B5 D334F471 02232ADF  .lN|.A.5S4tq.#*_
2E6FA520: BB248C8B 54E197F5 713D7886 4F71398D  ;$..Ta.uq=x.Oq9.
2E6FA530: 993342BA 90D2A677 96A6ABB9 8B72F19C  .3B:.R&w.&+9.rq.
2E6FA540: 4D454CBB A74D2342 B643FA74 A627656A  MEL;'M#B6Czt&'ej
2E6FA550: E1DDF0A9 ABDAC6FC 7986FC52 AD9AF67D  a]p)+ZF|y.|R-.v}
2E6FA560: C5
```

The following example displays the SSL VPN DTLS event information:

```
Router# debug webvpn dtls events

*Jun 15 10:28:13.731: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:14.575: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:14.575: WV-DTLS-3 Encryption done: context (0x67BF9BA0) pak (0x65EF4778),
ce_status = (1)
*Jun 15 10:28:14.575: WV-DTLS-2 DTLS: Switching cont pak in process path
*Jun 15 10:28:15.575: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:15.579: WV-DTLS-3 Encryption done: context (0x67BF9BA0) pak (0x66B2AAD4),
ce_status = (1)
*Jun 15 10:28:15.579: WV-DTLS-2 DTLS: Switching cont pak in process path
*Jun 15 10:28:16.575: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:16.575: WV-DTLS-3 Encryption done: context (0x67BF9BA0) pak (0x65EF4C04),
ce_status = (1)
*Jun 15 10:28:16.575: WV-DTLS-2 DTLS: Switching cont pak in process path
*Jun 15 10:28:17.579: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:17.579: WV-DTLS-3 Encryption done: context (0x67BF9BA0) pak (0x66B298A4),
ce_status = (1)
*Jun 15 10:28:17.579: WV-DTLS-2 DTLS: Switching cont pak in process path
*Jun 15 10:28:18.579: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:18.579: WV-DTLS-3 Encryption done: context (0x67BF9BA0) pak (0x65EF74F0),
ce_status = (1)
*Jun 15 10:28:18.579: WV-DTLS-2 DTLS: Switching cont pak in process path
*Jun 15 10:28:19.579: WV-DTLS-3 Decryption done: context (0x67BF9BA0) pak (0x67634074),
ce_status = (1)
*Jun 15 10:28:19.583: WV-DTLS-3 Encryption done: context (0x67BF9BA0) pak (0x65EF6BD8),
ce_status = (1)
*Jun 15 10:28:19.583: WV-DTLS-2 DTLS: Switching cont pak in process path
```

Related Commands

Command	Description
dtls port	Configures a desired port for the DTLS to listen.
svc dtls	Enables DTLS support on the Cisco IOS SSL VPN.

debug webvpn license

To display information related to license operations, events, and errors, use the **debug webvpn license**command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

debug webvpn license

no debug webvpn license

Syntax Description

This command has no arguments or keywords.

Command Default

Debug messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.0(1)M	This command was introduced.

Examples

The following is sample output from the **debug webvpn license** command when there is no valid license, and a user tries to log in to SSL VPN:

```
*Sep 17 09:36:21.091: %SSLVPN-3-LICENSE_NO_LICENSE: No valid license is available to use
IOS SSLVPN service
*Sep 17 09:36:21.091: WV-License: no valid reserve handle exists, request is not made
*Sep 17 09:36:21.091: WV-AAA: Error! No valid SSLVPN license exists
```

The following is sample output from the **debug webvpn license** command when there is a valid license, and a user tries to log in:

```
*Sep 17 09:40:15.535: WV-License: requested 1 count, granted 1 count, status is : No Error
```

The following is sample output from the **debug webvpn license** command when a user logs out and closes his or her session:

```
*Sep 17 09:41:48.143: WV-License: trying to release 1 count, released 1 count, status is :
 No Error
```

The following is sample output from the **debug webvpn license** command when the currently active license is a temporary (nonpermanent) license, and it has expired; some sessions are still active:

```
*Sep 18 00:28:19.018: WV-License: received licensing event for handle 0x1000004
*Sep 18 00:28:19.018:        Event type : LICENSE_CLIENT_EXPIRED
*Sep 18 00:28:19.018:        Count [usage/max(new max)]: 0/0(0)
*Sep 18 00:28:19.018: WV-License: setting lic expired flag!
*Sep 18 00:28:19.018: %SSLVPN-3-LICENSE_EXPIRED: IOS SSLVPN evaluation/extension license
has expired
*Sep 18 00:28:19.018: WV-License: event handling completed
*Sep 18 00:28:19.078: %LICENSE-2-EXPIRED: License for feature SSL_VPN_Test_Feature 1.0 has
 expired now.. UDI=CISCO2821:FHK1110F0PF
```

The following is sample output from the **debug webvpn license** command when the currently active license is a temporary (nonpermanent) license, and it has expired; some sessions are still active and a new user tries to log in:

```
*Sep 18 00:29:18.078: WV-AAA: AAA authentication request sent for user: "lab"
*Sep 18 00:29:18.078: WV-AAA: AAA Authentication Passed!
*Sep 18 00:29:18.078: %SSLVPN-3-LICENSE_EXPIRED: IOS SSLVPN evaluation/extension license
has expired
*Sep 18 00:29:18.078: WV-License: License expired, no more counts can be requested!
*Sep 18 00:29:18.078: WV-AAA: Error! No valid SSLVPN license exists
```

The following is sample output from the **debug webvpn license** command when a new license having a count higher than the currently active license is installed:

```
*Sep 18 00:39:12.658: WV-License: received licensing event
*Sep 18 00:39:12.658:        Event type : LICENSE_CLIENT_COUNT_CHANGED
*Sep 18 00:39:12.658:        Count [usage/max(new max)]: 0/0(169)
*Sep 18 00:39:12.770: WV-License: reserved extra count (158): No Error
*Sep 18 00:39:12.770: WV-License: reserved count now is 169
*Sep 18 00:39:12.774: WV-License: event handling completed
*Sep 18 00:39:12.774: WV-License: received licensing event for handle 0x1000004
*Sep 18 00:39:12.774:        Event type : LICENSE_CLIENT_COUNT_CHANGED
*Sep 18 00:39:12.774:        Count [usage/max(new max)]: 0/0(169)
```

The above outputs are self-explanatory.

Related Commands

Command	Description
show webvpn license	Displays the available count and the current usage.

debug wlccp ap

Use the debug wlccp ap privileged EXEC command to enable debugging for devices that interact with the access point that provides wireless domain services (WDS).

debug wlccp ap {**mn**| **rm** [**statistics**| **context**| **packet**]| **state**| **wds-discovery**}

Syntax Description

Command	Description
mn	(Optional) Activates display of debug messages related to client devices
rm [**statistics** | **context** | **packet**]	(Optional) Activates display of debug messages related to radio management • **statistics** --shows statistics related to radio management • **context** --shows the radio management contexts • **packet** --shows output related to packet flow
state	(Optional) Activates display of debug messages related to access point authentication to the WDS access point
wds-discovery	(Optional) Activates display of debug messages related to the WDS discovery process

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(11)JA	This command was first introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

This command is not supported on bridges.

Examples

This example shows how to begin debugging for LEAP-enabled client devices participating in Cisco Centralized Key Management (CCKM):

```
SOAP-AP# debug wlccp ap mn
```

Related Commands

Command	Description
show debugging	Displays all debug settings and the debug packet headers
show wlccp	Displays WLCCP information

debug wlccp ap rm enhanced-neighbor-list

Use the **debug wlccp ap rm enhanced-neighbor-list** privileged EXEC command to enable internal debugging information and error messages of the Enhanced Neighbor List feature. Use the **no** form of the command to disable the debugging and error messages.

[no] debug wlccp ap rm enhanced-neighbor-list

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)JA	This command was first introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

This command is not supported on bridges.

Examples

This example shows how to activate debugging and error messages of the Enhanced Neighbor List feature on the access point:

```
SOAP-AP# debug wlccp ap rm enhanced-neighbor-list
```

Related Commands

Command	Description
show debugging	Displays all debug settings and the debug packet headers
show wlccp	Displays WLCCP information
show wlccp ap rm enhanced-neighbor-list	Displays Enhanced Neighbor List feature related information.

debug wlccp packet

To display the packets being delivered to and from the wireless domain services (WDS) device, use the **debug wlccp packet**command in privileged EXEC mode. To disable the display of packets, use the **no** form of this command.

debug wlccp packet

no debug wlccp packet

Syntax Description

This command has no arguments of keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)JA	This command was introduced on Cisco Aironet access points.
12.3(11)T	This command was implemented on the following platforms: Cisco 2600XM, Cisco 2691, Cisco 2811, Cisco 2821, Cisco 2851, Cisco 3700, and Cisco 3800 series routers.

Related Commands

Command	Description
debug wlccp wds	Displays either WDS debug state or WDS statistics messages.
show wlccp wds	Shows information about access points and client devices on the WDS router.
wlccp authentication-server client	Configures the list of servers to be used for 802.1X authentication.
wlccp authentication-server infrastructure	Configures the list of servers to be used for 802.1X authentication for the wireless infrastructure devices.
wlccp wds priority interface	Enables a wireless device such as an access point or a wireless-aware router to be a WDS candidate.

debug wlccp rmlib

Use the debug wlccp rmlib privileged EXEC command to activate display of radio management library functions on the access point that provides wireless domain services (WDS).

debug wlccp rmlib

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)JA	This command was first introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

This command is not supported on bridges.

Examples

This example shows how to activate display of radio management library functions on the access point that provides WDS:

```
SOAP-AP# debug wlccp rmlib
```

Related Commands

Command	Description
show debugging	Displays all debug settings and the debug packet headers
show wlccp	Displays WLCCP information

debug wlccp wds

To display wireless domain services (WDS) debug messages, state messages, and failure statistics, use the **debug wlccp wds**command in privileged EXEC mode. To disable debug output, use the **no** form of this command.

debug wlccp wds {**authenticator**| **state**| **statistics**}

no debug wlccp wds

Syntax Description

authenticator	MAC and Extensible Authentication Protocol (EAP) authentication.
state	WDS state and debug messages.
statistics	WDS failure statistics.

Command Default

No default behavior or values

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(11)JA	This command was introduced.
12.3(11)T	This command was implemented on the following platforms: Cisco 2600XM, Cisco 2691, Cisco 2811, Cisco 2821, Cisco 2851, Cisco 3700, and Cisco 3800 series routers.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Examples

The following command displays WDS failure statistics:

```
Router# debug wlccp wds
statistics
```

Related Commands

Command	Description
debug wlccp packet	Displays packet traffic to and from the WDS router.

Command	Description
show wlccp wds	Shows information about access points and client devices on the WDS router.
wlccp authentication-server client	Configures the list of servers to be used for 802.1X authentication.
wlccp authentication-server infrastructure	Configures the list of servers to be used for 802.1X authentication for the wireless infrastructure devices.
wlccp wds priority interface	Enables a wireless device such as an access point or a wireless-aware router to be a WDS candidate.

debug wsma agent

To display debugging information on all Web Services Management Agents (WSMAs), use the **debug wsma agent** command in privileged EXEC mode. To disable the debugging information on all WSMAs, use the **no** form of this command.

debug wsma agent [**config**| **exec**| **filesys**| **notify**]

no debug wsma agent

Syntax Description

config	(Optional) Displays debugging information for the configuration agent.
exec	(Optional) Displays debugging information for the executive agent.
filesys	(Optional) Displays debugging information for the file system agent.
notify	(Optional) Displays debugging information for the notify agent.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(24)T	This command was introduced.
12.2(50)SY	This command was integrated into Cisco IOS Release 12.2(50)SY.
15.1(1)SG	This command was integrated into Cisco IOS Release 15.1(1)SG.
IOS XE Release 3.3SG	This command was integrated into Cisco IOS XE Release 3.3SG.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Examples

The following example shows how to display debugging information for a WSMA listener profile:

```
Router# debug wsma agent config
WSMA agent config debugging is on
```

Related Commands

Command	Description
debug wsma profile	Displays debugging information for all WSMA profiles.

debug wsma profile

To display debugging information on all Web Services Management Agent (WSMA) profiles, use the **debug wsma profile** command in privileged EXEC mode. To disable the debugging information on all WSMA profiles, use the **no** form of this command.

debug wsma profile [**listener**| **initiator**]

no debug wsma profile

Syntax Description

listener	Displays debugging information for the listener profile.
initiator	Displays debugging information for the initiator profile.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(24)T	This command was introduced.
15.1(1)T	This command was modified. The **initiator** keyword was added.
12.2(50)SY	This command was integrated into Cisco IOS Release 12.2(50)SY.
15.1(1)SG	This command was integrated into Cisco IOS Release 15.1(1)SG.
IOS XE Release 3.3SG	This command was integrated into Cisco IOS XE Release 3.3SG.
15.1(1)SY	This command was integrated into Cisco IOS Release 15.1(1)SY.

Examples

The following example shows how to display debugging information for a WSMA listener profile:

```
Router# debug wsma profile listener

WSMA profile listener debugging is on
```

The following example shows how to display debugging information for a WSMA initiator profile:

```
Router# debug wsma profile initiator

WSMA profile initiator debugging is on
```

Related Commands

Command	Description
debug wsma agent	Displays debugging information for all WSMAs.

debug wsapi

To collect and display traces for the Cisco Unified Communication IOS services application programming interface, use the **debug wsapi** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug wsapiinfrastructure| xcc| xcdr| xsvcall| default| detail| error| event| function| inout| messages

no debug wsapiinfrastructure| xcc| xcdr| xsvcall| default| detail| error| event| function| inout| messages

Syntax Description

infrastructure	Enables debugging traces on the infrastructure.
xcc	Enables debugging traces on the xcc provider.
xcdr	Enables debugging traces on the xcdr provider.
xsvc	Enables debugging traces on the xsvc provider
all	Enables all debugging traces.
default	Enables default debugging traces.
detail	Enables detailed debugging traces.
error	Enables error debugging traces.
event	Enables event debugging traces.
function	Enables function debugging traces.
inout	Enables inout debugging traces.
messages	Enables API message traces.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(2)T	This command was introduced.

Usage Guidelines

Use this command to enable debugging traces for the Cisco Unified Communicaion IOS services subsystems.

Examples

The following is a sample output from the **debug wsapi infrastructure** command for an XCC registration.

```
Router# debug wsapi infrastructure

23:25:09: //WSAPI/INFRA/wsapi_https_urlhook:
23:25:09: //WSAPI/INFRA: app_name cisco_xcc in url /cisco_xcc in port 8090
23:25:09: //WSAPI/INFRA/wsapi_https_urlhook: Exit
23:25:09: //WSAPI/INFRA/wsapi_https_post_action:
23:25:09: wsapi_https_data_read: <soapenv:Envelope
xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"><soapenv:Body><RequestXccRegister
[illegible]
 AUTHORIZE_CALL REDIRECTED ALERTING CONNECTED TRANSFERRED CALL_DELIVERY DISCONNECTED
HANDOFFLEAVE
HANDOFFJOIN</connectionEventsFilter><mediaEventsFilter>MODE_CHANGE DTMF TONE_BUSY TONE_DIAL
 TONE_SECOND_DIAL TONE_RINGBACK TONE_OUT_OF_SERVICE
MEDIA_ACTIVITY</mediaEventsFilter><msgHeader><transactionID>txID001</transactionID></msgHeader><providerData><url>http://10.1.1.1:8090/cisco_xcc</url></providerData></RequestXccRegister></soapenv:Body></soapenv:Envelope>
23:25:09: //WSAPI/INFRA/27/0/wsapi_https_recv:
23:25:09: //WSAPI/INFRA/27/0/txID001/wsapi_ph_request_msg_handle:
23:25:09: //WSAPI/INFRA/27/0/txID001: prov_type 0 msg_type 6 prov_state 1
23:25:09: //WSAPI/INFRA/wsapi_create_common_msg:
23:25:09: //WSAPI/INFRA/wsapi_create_common_msg: Exit
23:25:09: //WSAPI/INFRA/27/0/txID001/wsapi_send_outbound_response:
23:25:09: wsapi_dump_msg: type 8
23:25:09: transactionID txID001
23:25:09: registrationID 50674FC:XCC:myapp:9
23:25:09: ResponseXccRegister:
23:25:09: providerStatus 1
23:25:09: //WSAPI/INFRA/27/0/txID001/wsapi_send_outbound_response: Exit
23:25:09: wsapi_send_ResponseRegister:mem_mgr_mempool_free: mem_refcnt(3CA18B8)=0 - mempool
 cleanup
23:25:09: //WSAPI/INFRA/27/0/txID001/wsapi_https_recv: Exit
23:25:09: wsapi_https_data_write: <?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"><SOAP:Body><ResponseXccRegister
xmlns="http://www.cisco.com/schema/cisco_xcc/v1_0"><msgHeader><transactionID>txID001</transactionID><registrationID>50674FC:XCC:myapp:9</registrationID></msgHeader><providerStatus>IN_SERVICE</providerStatus></ResponseXccRegister></SOAP:Body></SOAP:Envelope>
23:25:09: //WSAPI/INFRA/wsapi_https_post_action: Exit
```

The following is a partial debug log from the **debug wsapi xcc all** command for a call..

```
Router# debug wsapi xcc all

23:27:20: //WSAPI/XCC/check_xccp_active:177:
23:27:20: //WSAPI/XCC/provider_base_get_state:248:
23:27:20: //WSAPI/XCC/provider_base_get_registration_count:212:
23:27:20: //WSAPI/XCC/check_xccp_active:177:
23:27:20: //WSAPI/XCC/provider_base_get_state:248:
23:27:20: //WSAPI/XCC/provider_base_get_registration_count:212:
23:27:20: //WSAPI/XCC/xccp_sessStore_call_add:271:
23:27:20: //WSAPI/XCC/xccp_sessStore_get_db:145:
23:27:20: //WSAPI/XCC/xccp_session_call_add:353: xcc session successfully added
23:27:20: //WSAPI/XCC/xccp_sessStore_call_add:285: xcc call successfully added
23:27:20: //WSAPI/XCC/check_xccp_active:177:
23:27:20: //WSAPI/XCC/provider_base_get_state:248:
23:27:20: //WSAPI/XCC/provider_base_get_registration_count:212:
23:27:20: //WSAPI/XCC/xccp_create_outbound_msg_space:677:
23:27:20: //WSAPI/XCC/xccp_sessStore_get_callData:225:
23:27:20: //WSAPI/XCC/xccp_sessStore_get_db:145:
23:27:20: //WSAPI/XCC/xccp_session_get_callData:445:
23:27:20: //WSAPI/XCC/check_xccp_active:177:
23:27:20: //WSAPI/XCC/provider_base_get_state:248:
23:27:20: //WSAPI/XCC/provider_base_get_registration_count:212:
23:27:20: //WSAPI/XCC/xccp_notify_events:434:
23:27:20: //WSAPI/XCC/xccp_queue_events:304:
23:27:20: //WSAPI/XCC/provider_base_event_new:335:
23:27:20: //WSAPI/UNKNOWN/event_base_new:267:
23:27:20: //WSAPI/XCC: magic [0xBABE] state[EVENT_STATE_ACTIVE] owner [0x1148C178] evSize[56]
```

```
 debFlag[3] evHdlr[0x894D834] evHdlFree[0x894DB00]
23:27:20: //WSAPI/UNKNOWN/event_base_new:292: event base new succ
23:27:20: //WSAPI/XCC/provider_base_event_new:360: provider base eventNew success
23:27:20: //WSAPI/XCC/provider_base_add_ev_to_q:393:
23:27:20: //WSAPI/XCC/check_xccp_active:177:
23:27:20: //WSAPI/XCC/provider_base_get_state:248:
23:27:20: //WSAPI/XCC/provider_base_get_registration_count:212:
23:27:20: //WSAPI/XCC/xccp_create_outbound_msg_space:677:
23:27:20: //WSAPI/XCC/xccp_sessStore_get_callData:225:
23:27:20: //WSAPI/XCC/xccp_sessStore_get_db:145:
23:27:20: //WSAPI/XCC/xccp_session_get_callData:445:
23:27:20: //WSAPI/XCC/check_xccp_active:177:
23:27:20: //WSAPI/XCC/provider_base_get_state:248:
23:27:20: //WSAPI/XCC/provider_base_get_registration_count:212:
23:27:20: //WSAPI/XCC/xccp_solicit_events:359:
23:27:20: //WSAPI/XCC/xccp_queue_events:304:
23:27:20: //WSAPI/XCC/provider_base_event_new:335:
23:27:20: //WSAPI/UNKNOWN/event_base_new:267:
23:27:20: //WSAPI/XCC: magic [0xBABE] state[EVENT_STATE_ACTIVE] owner [0x1148C178] evSize[56]
 debFlag[3] evHdlr[0x894D834] evHdlFree[0x894DB00]
23:27:20: //WSAPI/UNKNOWN/event_base_new:292: event base new succ
23:27:20: //WSAPI/XCC/provider_base_event_new:360: provider base eventNew success
23:27:20: //WSAPI/XCC/provider_base_add_ev_to_q:393:
23:27:20: //WSAPI/XCC/provider_base_process_events:444:
23:27:20: //WSAPI/XCC/xccp_handle_events:153:
23:27:20: //WSAPI/INFRA/wsapi_send_outbound_message:
23:27:20: //WSAPI/INFRA/wsapi_send_outbound_message_by_provider_info:
23:27:20: //WSAPI/XCC/wsapi_xcc_encode_outbound_msg:
23:27:20: //WSAPI/XCC/wsapi_xcc_encode_outbound_msg: Exit
23:27:20: //WSAPI/INFRA/0/1527/50875A4:319:out_url http://sj22lab-as2:8090/xcc
23:27:20: wsapi_send_outbound_message_by_provider_info: <?xml version="1.0"
encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"><SOAP:Body><NotifyXccConnectionData
[redacted]
23:27:20: //WSAPI/INFRA/0/1527/50875A4:319/wsapi_send_outbound_message_by_provider_info:
Exit
.
.
.
```

debug x25

To display information about all X.25 traffic or a specific X.25 service class, use the **debug x25**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug x25 [**only**| **cmns**| **xot**] [**events**| **all**] **[dump]**

no debug x25 [**only**| **cmns**] [**events**| **all**] **[dump]**

Syntax Description

only	(Optional) Displays information about X.25 services only.
cmns	(Optional) Displays information about CMNS services only.
xot	(Optional) Displays information about XOT services only.
events	(Optional) Displays all traffic except Data and Receiver Ready (RR) packets.
all	(Optional) Displays all traffic. This is the default.
dump	(Optional) Displays the encoded packet contents in hexadecimal and ASCII formats.

Command Default

All traffic is displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.0(5)T	For Domain Name System (DNS)-based X.25 routing, additional functionality was added to the **debug x25 events** command to describe the events that occur while the X.25 address is being resolved to an IP address using a DNS server. The **debug domain** command can be used along with **debug x25 events** to observe the whole DNS-based X.25 routing data flow.
12.0(7)T	For the X.25 Closed User Groups (CUGs) feature, functionality was added to the **debug x25 events** command to describe events that occur during CUG activity.

Release	Modification
12.2(8)T	The **debug x25 events** command was enhanced to display events specific to Record Boundary Preservation protocol.
12.3(2)T	The **dump** keyword was added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelin

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

The **debug x25**, **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands all generate the same basic output. The **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands narrow the scope of the **debug x25**output to specific virtual circuits or types of traffic.

This command is particularly useful for diagnosing problems encountered when placing calls. The **debug x25 all** output includes data, control messages, and flow control packets for all virtual circuits of the router.

All **debug x25** commands can take either the **events** or the **all** keyword. The keyword **all** is the default and causes all packets meeting the other debug criteria to be reported. The keyword **events** omits reports of any Data or RR flow control packets; the normal flow of data and RR packets is commonly large and less interesting to the user, so event reporting can significantly decrease the processor load induced by debug reporting.

Use the **dump** keyword to display the entire contents, including user data, of X.25 packets. The encoded X.25 packet contents are displayed after the standard packet description. The output includes the offset into the packet and the display of the data in both hexadecimal and ASCII formats.

Caution The X.25 packet information that is reported by using the **dump** keyword may contain sensitive data; for example, clear-text account identities and passwords. The network access policies and router configuration should be controlled appropriately to address this risk.

Examples

The following is sample output from the **debug x25**command, displaying output concerning the functions X.25 restart, call setup, data exchange, and clear:

```
Router# debug x25
Serial0: X.25 I R/Inactive Restart (5) 8 lci 0
  Cause 7, Diag 0 (Network operational/No additional information)
Serial0: X.25 O R3 Restart Confirm (3) 8 lci 0
Serial0: X.25 I P1 Call (15) 8 lci 1
From(6): 170091 To(6): 170090
   Facilities: (0)
   Call User Data (4): 0xCC000000 (ip)
Serial0: X.25 O P3 Call Confirm (3) 8 lci 1
Serial0: X.25 I D1 Data (103) 8 lci 1 PS 0 PR 0
Serial0: X.25 O D1 Data (103) 8 lci 1 PS 0 PR 1
Serial0: X.25 I P4 Clear (5) 8 lci 1
  Cause 9, Diag 122 (Out of order/Maintenance action)
Serial0: X.25 O P7 Clear Confirm (3) 8 lci 1
```

Examples

The following example of the **debug x25**command with the **events** keyword shows output related to the DNS-Based X.25 Routing feature. It shows messages concerning access to the DNS server. In the example, nine alternate addresses for one XOT path are entered into the DNS server database. All nine addresses are returned to the host cache of the router by the DNS server. However, only six addresses will be used during the XOT switch attempt because this is the limit that XOT allows.

```
Router# debug x25 events
00:18:25:Serial1:X.25 I R1 Call (11) 8 lci 1024
00:18:25: From (0): To (4):444
00:18:25: Facilities:(0)
00:18:25: Call User Data (4):0x01000000 (pad)
00:18:25:X.25 host name sent for DNS lookup is "444"
00:18:26:%3-TRUNCATE_ALT_XOT_DNS_DEST:Truncating excess XOT addresses (3)
returned by DNS
00:18:26:DNS got X.25 host mapping for "444" via network
00:18:32:[10.1.1.8 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:38:[10.1.1.7 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:44:[10.1.1.6 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:50:[10.1.1.5 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:56:[10.1.1.4 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:20:04:[10.1.1.3,1998/10.1.1.3,11007]:XOT O P2 Call (17) 8 lci 1
00:20:04: From (0): To (4):444
00:20:04: Facilities:(6)
00:20:04:  Packet sizes:128 128
00:20:04:  Window sizes:2 2
00:20:04: Call User Data (4):0x01000000 (pad)
00:20:04:[10.1.1.3,1998/10.1.1.3,11007]:XOT I P2 Call Confirm (11) 8 lci 1
00:20:04: From (0): To (0):
00:20:04: Facilities:(6)
00:20:04:  Packet sizes:128 128
00:20:04:  Window sizes:2 2
00:20:04:Serial1:X.25 O R1 Call Confirm (5) 8 lci 1024
00:20:04: From (0): To (0):
00:20:04: Facilities:(0)
```

Examples

The following examples show output for the **x25 debug**command with the **events** keyword when record boundary preservation (RBP) has been configured using the **x25 map rbp local** command.

The following display shows establishment of connection:

```
X25 RBP:Incoming connection for port 9999 from 10.0.155.30 port 11001
Serial0/1:X.25 O R1 Call (10) 8 lci 64
  From (5):13133 To (5):12131
  Facilities:(0)
Serial0/1:X.25 I R1 Call Confirm (3) 8 lci 64
```

The following display shows that the X.25 call was cleared by the X.25 host:

```
Serial0/1:X.25 I R1 Clear (5) 8 lci 64
  Cause 0, Diag 122 (DTE originated/Maintenance action)
X25 RBP:X.25 circuit cleared
Serial0/1:X.25 O R1 Clear Confirm (3) 8 lci 64
```

The following display shows that the TCP session has terminated:

```
[10.0.155.30,11000/10.0.155.33,9999]:TCP receive error, End of data transfer
X25 RBP:End of data transfer
Serial0/1:X.25 O R1 Clear (5) 8 lci 64
```

```
Cause 9, Diag 122 (Out of order/Maintenance action)
Serial0/1:X.25 I R1 Clear Confirm (3) 8 lci 64
```

The following examples show output of the **x25 debug**command with the **events** keyword when RBP has been configured using the **x25 pvc rbp local** command.

The following display shows data on the permanent virtual circuit (PVC) before the TCP session has been established:

```
X25 RBP:Data on unconnected PVC
Serial1/0:X.25 O D1 Reset (5) 8 lci 1
  Cause 0, Diag 113 (DTE originated/Remote network problem)
Serial1/0:X.25 I D2 Reset Confirm (3) 8 lci 1
```

The following display shows establishment of connection:

```
X25 RBP:Incoming connection for port 9998 from 2.30.0.30 port 11002
Serial1/0:X.25 O D1 Reset (5) 8 lci 1
  Cause 0, Diag 0 (DTE originated/No additional information)
Serial1/0:X.25 I D2 Reset Confirm (3) 8 lci 1
```

The following display shows termination of connection when the X.25 PVC was reset:

```
Serial1/0:X.25 I D1 Reset (5) 8 lci 1
  Cause 15, Diag 122 (Network operational (PVC)/Maintenance action)
X25 RBP:Reset packet received
Serial1/0:X.25 O D3 Reset Confirm (3) 8 lci 1
```

The following display shows that the TCP session has terminated:

```
[2.30.0.30,11003/2.30.0.33,9998]:TCP receive error, End of data transfer
X25 RBP:End of data transfer
Serial1/0:X.25 O D1 Reset (5) 8 lci 1
  Cause 0, Diag 113 (DTE originated/Remote network problem)
Serial1/0:X.25 I D2 Reset Confirm (3) 8 lci 1
```

The following examples show output of the **x25 debug**command with the **events** keyword when RBP has been configured using the **x25 map rbp remote** command.

The following display shows that the X.25 call was cleared:

```
Serial0/1:X.25 I R1 Clear (5) 8 lci 1024
  Cause 0, Diag 122 (DTE originated/Maintenance action)
X25 RBP:X.25 circuit cleared
Serial0/1:X.25 O R1 Clear Confirm (3) 8 lci 1024
```

The following display shows that the X.25 call was reset:

```
Serial0/1:X.25 I D1 Reset (5) 8 lci 1024
  Cause 0, Diag 122 (DTE originated/Maintenance action)
X25 RBP:Reset packet received
Serial0/1:X.25 O R1 Clear (5) 8 lci 1024
  Cause 9, Diag 122 (Out of order/Maintenance action)
Serial0/1:X.25 I R1 Clear Confirm (3) 8 lci 1024
```

The following examples show output of the **x25 debug**command with the **events** keyword when RBP has been configured using the **x25 pvc rbp remote** command.

The following display shows that the X.25 PVC has been reset:

```
Serial0/0:X.25 I D1 Reset (5) 8 lci 1
  Cause 0, Diag 122 (DTE originated/Maintenance action)
X25 RBP:Reset packet received
Serial0/0:X.25 O D2 Reset Confirm (3) 8 lci 1
```

The following display shows that the connection was terminated when the X.25 interface was restarted:

```
Serial0/0:X.25 I R1 Restart (5) 8 lci 0
  Cause 0, Diag 122 (DTE originated/Maintenance action)
```

```
X25 RBP:X.25 PVC inactive
Serial0/0:X.25 O R2 Restart Confirm (3) 8 lci 0
Serial0/0:X.25 O D1 Reset (5) 8 lci 1
  Cause 1, Diag 113 (Out of order (PVC)/Remote network problem)
Serial0/0:X.25 I D3 Reset Confirm (3) 8 lci 1
```

Examples

The following is sample output for the **debug x25 dump** command. The encoded X.25 packet contents are displayed after the standard packet description. The output includes the offset into the packet and the display of the data in both hexadecimal and ASCII formats.

```
Router# debug x25 dump
Serial1: X.25 O R/Inactive Restart (5) 8 lci 0
   Cause 0, Diag 0 (DTE originated/No additional information)
       0: 1000FB00 00                          ..{..
Serial1: X.25 I R2 Restart (5) 8 lci 0
   Cause 7, Diag 0 (Network operational/No additional information)
       0:                                1000FB              ..{
       3: 0700                                 ..
Serial1: X.25 I R1 Call (13) 8 lci 1
   From (4): 2501 To (4): 2502
   Facilities: (0)
   Call User Data (4): 0xCC000000 (ip)
       0:            10010B 44250225 0100CC00       ...D%.%..L.
      11: 0000                                 ..
Serial1: X.25 O R1 Call Confirm (3) 8 lci 1
       0: 10010F                               ...
Serial1: X.25 I D1 Data (103) 8 lci 1 PS 0 PR 0
       0:            100100 45000064 00000000       ...E..d....
      11: FF01A764 0A190001 0A190002 0800CBFB  ..'d.........K{
      27: 0B1E22CA 00000000 00028464 ABCDABCD  .."J.......d+M+M
      43: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
      59: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
      75: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
      91: ABCDABCD ABCDABCD ABCDABCD           +M+M+M+M+M+M
Serial1: X.25 O D1 Data (103) 8 lci 1 PS 0 PR 1
       0:            100120 45000064 00000000       .. E..d....
      11: FF01A764 0A190002 0A190001 0000D3FB  ..'d.........S{
      27: 0B1E22CA 00000000 00028464 ABCDABCD  .."J.......d+M+M
      43: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
      59: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
      75: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
      91: ABCDABCD ABCDABCD ABCDABCD           +M+M+M+M+M+M
 Serial1: X.25 I R1 Clear (5) 8 lci 1
   Cause 9, Diag 122 (Out of order/Maintenance action)
       0:            100113 097A                   ....z
 Serial1: X.25 O R1 Clear Confirm (3) 8 lci 1
       0:                  100117                  .
```

The table below describes significant fields shown in the displays.

Table 12: debug x25 Field Descriptions

Field	Description
Serial0	Interface on which the X.25 event occurred.
X.25	Type of event this message describes.
I	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.

Field	Description
R3	State of the service or virtual circuit (VC). Possible values follow: R/Inactive--Packet layer awaiting link layer service R1--Packet layer ready R2--Data terminal equipment (DTE) restart request R3--DCE restart indication P/Inactive--VC awaiting packet layer service P1--Idle P2--DTE waiting for DCE to connect CALL P3--DCE waiting for DTE to accept CALL P4--Data transfer P5--CALL collision P6--DTE clear request P7--DCE clear indication D/Inactive--VC awaiting setup D1--Flow control ready D2--DTE reset request D3--DCE reset indication Refer to Annex B of the *ITU-T Recommendation X.25* for more information on these states.

Field	Description
Restart	The type of X.25 packet. Possible values follow: R Events • Restart • Restart Confirm • Diagnostic P Events • Call • Call Confirm • Clear • Clear Confirm D Events • Reset • Reset Confirm D1 Events • Data • Receiver Not Ready (RNR) • RR (Receiver Ready) • Interrupt • Interrupt Confirm XOT Overhead • PVC Setup Refer to RFC 1613 *Cisco Systems X.25 over TCP (XOT)* for information about the XOT PVC Setup packet type.
(5)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 and 128.
lci 0	VC number. Refer to Annex A of the *ITU-T Recommendation X.25* for information on VC assignment.

Field	Description
Cause 7	Code indicating the event that triggered the packet. The Cause field can appear only in entries for Clear, Reset, and Restart packets. Possible values for the Cause field can vary, depending on the type of packet. Refer to the appendix "X.25 Cause and Diagnostic Codes" for an explanation of these codes.
Diag 0	Code providing an additional hint of what, if anything, went wrong. The Diag field can appear only in entries for Clear, Diagnostic (as "error 0"), Reset, and Restart packets. Refer to the appendix "X.25 Cause and Diagnostic Codes" for an explanation of these codes.
(Network operational/ No additional information)	The standard explanations of the Cause and Diagnostic codes (*cause*/*diag*).
From (6):170091	Source address. (6) indicates the number of digits in the address that follows. The source address is part of the address block that may be encoded in Call Setup packets.
To (6): 170090	Destination address. (6) indicates the number of digits in the address that follows. The destination address is part of the address block that may be encoded in Call Setup packets.
Facilities:(0)	Indicates that a facilities block is encoded and that it consists of 0 bytes. A breakdown of the encoded facilities (if any) follows.
Call User Data (4):	Indicates that the Call User Data (CUD) field is present and consists of 4 bytes.
0xCC000000 (ip)	Protocol identifier (PID). This subfield of the CUD field is presented in the output as a hexadecimal string followed by the name of the protocol (in this case, IP) that the string represents. Any bytes following the PID are designated "user data" and may be used by an application separately from the PID.

Related Commands

Command	Description
debug x25 interface	Displays information about a specific X.25 or CMNS context or virtual circuit.

Command	Description
debug x25 vc	Displays information about traffic for all virtual circuits that use a given number.
debug x25 xot	Displays information about traffic to or from a specific XOT host.

debug x25 annexg

To display information about Annex G (X.25 over Frame Relay) events, use the **debug x25 annexg**command. To disable debugging output, use the **no** form of this command.

debug x25 annexg

no debug x25 annexg

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0 T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

It is generally recommended that the **debug x25 annexg** command be used only when specifically requested by Cisco TAC to obtain information about a problem with an Annex G configuration. The messages displayed by the **debug x25 annexg** command are meant to aid in the diagnosing of internal errors.

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

Examples

The following shows sample output from the **debug x25 annexg** command for a Frame Relay data-link connection identifier (DLCI) configured for Annex G operation:

```
Router# debug x25 annexg

Jul 31 05:23:20.316:annexg_process_events:DLCI 18 attached to interface Serial2/0:0 is
ACTIVE
Jul 31 05:23:20.316:annexg_ctxt_create:Creating X.25 context over Serial2/0:0 (DLCI:18 using
 X.25 profile:OMC), type 10, len 2, addr 00 12
Jul 31 05:23:20.316:annexg_create_lower_layer:Se2/0:0 DLCI 18, payload 1606, overhead 2
Jul 31 05:23:20.320:annexg_restart_tx:sending pak to Serial2/0:0
Jul 31 05:23:23.320:annexg_restart_tx:sending pak to Serial2/0:0
```

The table below describes significant fields shown in the display.

Table 13: debug x25 annexg Field Descriptions

Field	Description
payload	Amount of buffer space available per message before adding Frame Relay and device-specific headers.
overhead	The length of the Frame Relay header and any device-specific header that may be needed.

Related Commands

Command	Description
debug x25	Displays information about all X.25 traffic or a specific X.25 service class.
debug x25 interface	Displays information about specific X.25, Annex G or CMN contexts or virtual circuits that occur on the identified interface.
debug x25 vc	Displays information about traffic for all virtual circuits that have a given number.

debug x25 aodi

To display information about an interface running PPP over an X.25 session, use the **debug x25 aodi**command. To disable debugging output, use the **no** form of this command.

debug x25 aodi

no debug x25 aodi

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Use the **debug x25 aodi** command to display interface PPP events running over an X.25 session and to debug X.25 connections between a client and server configured for Always On/Dynamic ISDN (AO/DI).

Examples

The following examples show the normal sequence of events for both the AO/DI client and the server sides:

Examples

```
Router# debug x25 aodi
PPP-X25: Virtual-Access1: Initiating AODI call request
PPP-X25: Bringing UP X.25 AODI VC
PPP-X25: AODI Client Call Confirm Event Received
PPP-X25: Cloning interface for AODI is Di1
PPP-X25: Queuing AODI Client Map Event
PPP-X25: Event:AODI Client Map
PPP-X25: Created interface Vi2 for AODI service
PPP-X25: Attaching primary link Vi2 to Di1
PPP-X25: Cloning Vi2 for AODI service using Di1
PPP-X25: Vi2: Setting the PPP call direction as OUT
PPP-X25: Vi2: Setting vectors for RFC1598 operation on BRI3/0:0 VC 0
PPP-X25: Vi2: Setting the interface default bandwidth to 10 Kbps
PPP-X25: Virtual-Access2: Initiating AODI call request
PPP-X25: Bringing UP X.25 AODI VC
PPP-X25: AODI Client Call Confirm Event Received
```

Examples

```
Router# debug x25 aodi
PPP-X25: AODI Call Request Event Received
PPP-X25: Event:AODI Incoming Call Request
PPP-X25: Created interface Vi1 for AODI service
PPP-X25: Attaching primary link Vi1 to Di1
PPP-X25: Cloning Vi1 for AODI service using Di1
```

```
PPP-X25: Vi1: Setting vectors for RFC1598 operation on BRI3/0:0 VC 1
PPP-X25: Vi1: Setting the interface default bandwidth to 10 Kbps
PPP-X25: Binding X.25 VC 1 on BRI3/0:0 to Vi1
```

debug x25 interface

To display information about the specific X.25, Annex G or Connection Mode Network Service (CMN) contexts or virtual circuits that occur on the identified interface, use the **debug x25 interface**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug x25 interface {*serial-interface*| *cmns-interface* [**mac** *mac-address*]} [**vc** *number*] [**events**| **all**] **[dump]**

no debug x25 interface {*serial-interface*| *cmns-interface* [**mac** *mac-address*]} [**vc** *number*] [**events**| **all**] **[dump]**

Syntax Description

serial-interface	Serial interface number that is configured for X.25 or Annex G service.
cmns-interface	Interface supporting CMNS traffic and, if specified, the MAC address of a remote host. The interface type can be Ethernet, Token Ring, or FDDI.
mac *mac-address*	(Optional) MAC address of the CMNS interface and remote host.
vc *number*	(Optional) Virtual circuit number. Range is from 1 to 4095.
events	(Optional) Displays all traffic except Data and Receiver Ready (RR) packets.
all	(Optional) Displays all traffic. This is the default.
dump	(Optional) Displays the encoded packet contents in hexadecimal and ASCII formats.

Command Default

All traffic is displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.3(2)T	The **dump** keyword was added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelin

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

The **debug x25**, **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands all generate the same basic output. The **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands narrow the scope of the **debug x25** output to specific virtual circuits or types of traffic.

The **debug x25 interface** command is useful for diagnosing problems encountered with a single X.25 or CMNS host or virtual circuit.

The keyword **all** is the default and causes all packets meeting the other debug criteria to be reported. The keyword **events** omits reports of any Data or RR flow control packets; the normal flow of data and RR packets is commonly large and less interesting to the user, so event reporting can significantly decrease the processor load induced by debug reporting.

Use the **dump** keyword to display the entire contents, including user data, of X.25 packets. The encoded X.25 packet contents are displayed after the standard packet description. The output includes the offset into the packet and the display of the data in both hexadecimal and ASCII formats.

Caution The X.25 packet information that is reported by using the **dump** keyword may contain sensitive data; for example, clear-text account identities and passwords. The network access policies and router configuration should be controlled appropriately to address this risk.

Examples

The following is sample output from the **debug x25 interface**command:

```
Router# debug x25 interface serial 0
X.25 packet debugging is on
X.25 packet debugging is restricted to interface serial0
Serial0: X.25 I R/Inactive Restart (5) 8 lci 0
  Cause 7, Diag 0 (Network operational/No additional information)
Serial0: X.25 O R3 Restart Confirm (3) 8 lci 0
Serial0: X.25 I P1 Call (15) 8 lci 1
From(6): 170091 To(6): 170090
   Facilities: (0)
   Call User Data (4): 0xCC000000 (ip)
Serial0: X.25 O P3 Call Confirm (3) 8 lci 1
Serial0: X.25 I D1 Data (103) 8 lci 1 PS 0 PR 0
Serial0: X.25 O D1 Data (103) 8 lci 1 PS 0 PR 1
Serial0: X.25 I P4 Clear (5) 8 lci 1
  Cause 9, Diag 122 (Out of order/Maintenance action)
Serial0: X.25 O P7 Clear Confirm (3) 8 lci 1
```

The table below describes the significant fields shown in the display.

Table 14: debug x25 interface Field Descriptions

Field	Description
Serial0	Interface on which the X.25 event occurred.
X.25	Type of event this message describes.

Field	Description
I	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.
R3	State of the service or virtual circuit (VC). Possible values follow: R/Inactive--Packet layer awaiting link layer service R1--Packet layer ready R2--Data terminal equipment (DTE) restart request R3--DCE restart indication P/Inactive--VC awaiting packet layer service P1--Idle P2--DTE waiting for DCE to connect CALL P3--DCE waiting for DTE to accept CALL P4--Data transfer P5--CALL collision P6--DTE clear request P7--DCE clear indication D/Inactive--VC awaiting setup D1--Flow control ready D2--DTE reset request D3--DCE reset indication Refer to Annex B of the *ITU-T Recommendation X.25* for more information on these states.

Field	Description
Restart	The type of X.25 packet. Possible values follow: R Events • Restart • Restart Confirm • Diagnostic P Events • Call • Call Confirm • Clear • Clear Confirm D Events • Reset • Reset Confirm D1 Events • Data • Receiver Not Ready (RNR) • RR (Receiver Ready) • Interrupt • Interrupt Confirm XOT Overhead • PVC Setup
(5)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 and 128.
lci 0	VC number. Refer to Annex A of the *ITU-T Recommendation X.25* for information on VC assignment.

Field	Description
Cause 7	Code indicating the event that triggered the packet. The Cause field can appear only in entries for Clear, Reset, and Restart packets. Possible values for the Cause field can vary, depending on the type of packet. Refer to the appendix "X.25 Cause and Diagnostic Codes" for an explanation of these codes.
Diag 0	Code providing an additional hint of what, if anything, went wrong. The Diag field can appear only in entries for Clear, Diagnostic (as "error 0"), Reset, and Restart packets. Refer to the appendix "X.25 Cause and Diagnostic Codes" for an explanation of these codes.
(Network operational/ No additional information)	The standard explanations of the Cause and Diagnostic codes (*cause*/*diag*).
From (6):170091	Source address. (6) indicates the number of digits in the address that follows. The source address is part of the address block that may be encoded in Call Setup packets.
To (6): 170090	Destination address. (6) indicates the number of digits in the address that follows. The destination address is part of the address block that may be encoded in Call Setup packets.
Facilities:(0)	Indicates that a facilities block is encoded and that it consists of 0 bytes. A breakdown of the encoded facilities (if any) follows.
Call User Data (4):	Indicates that the Call User Data (CUD) field is present and consists of 4 bytes.
0xCC000000 (ip)	Protocol identifier (PID). This subfield of the CUD field is presented in the output as a hexadecimal string followed by the name of the protocol (in this case, IP) that the string represents. Any bytes following the PID are designated "user data" and may be used by an application separately from the PID.

Related Commands

Command	Description
debug x25	Displays information about all X.25 traffic or a specific X.25 service class.

Command	Description
debug x25 vc	Displays information about traffic for all virtual circuits that use a given number.
debug x25 xot	Displays information about traffic to or from a specific XOT host.

debug x25 vc

To display information about traffic for all virtual circuits that have a given number, use the **debug x25 vc**command. To disable debugging output, use the **no**form of this command.

debug x25 vc *number* [**events**| **all**] **[dump]**

no debug x25 vc *number* [**events**| **all**] **[dump]**

Syntax Description

number	Virtual circuit number. Range is from 1 to 4095.
events	(Optional) Displays all traffic except Data and Receiver Ready (RR) packets.
all	(Optional) Displays all traffic. This is the default.
dump	(Optional) Displays the encoded packet contents in hexadecimal and ASCII formats.

Command Default

All traffic is displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.3(2)T	The **dump** keyword was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelin

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

The **debug x25, debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands all generate the same basic output. The **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands narrow the scope of the **debug x25** output to specific virtual circuits or types of traffic.

Because no interface is specified by the **debug x25 vc** command, traffic on any virtual circuit that has the specified number is reported.

Virtual circuit (VC) zero (vc 0) cannot be specified. It is used for X.25 service messages, such as RESTART packets, not virtual circuit traffic. Service messages can be monitored only when no virtual circuit filter is used.

Use the **dump** keyword to display the entire contents, including user data, of X.25 packets. The encoded X.25 packet contents are displayed after the standard packet description. The output includes the offset into the packet and the display of the data in both hexadecimal and ASCII formats.

Caution The X.25 packet information that is reported by using the **dump** keyword may contain sensitive data; for example, clear-text account identities and passwords. The network access policies and router configuration should be controlled appropriately to address this risk.

Examples

The following shows sample output from the **debug x25 vc** command:

```
Router# debug x25 vc 1 events
X.25 special event debugging is on
X.25 debug output restricted to VC number 1
Router# show debug
X.25 (filtered for VC 1):
  X.25 special event debugging is on
*Jun 18 20:22:29.735 UTC:Serial0:X.25 O R1 Call (13) 8 lci 1
*Jun 18 20:22:29.735 UTC:  From (4):2501 To (4):2502
*Jun 18 20:22:29.735 UTC:  Facilities:(0)
*Jun 18 20:22:29.735 UTC:  Call User Data (4):0xCC000000 (ip)
*Jun 18 20:22:29.739 UTC:Serial0:X.25 I R1 Call Confirm (3) 8 lci 1
*Jun 18 20:22:36.651 UTC:Serial0:X.25 O R1 Clear (5) 8 lci 1
*Jun 18 20:22:36.651 UTC:  Cause 9, Diag 122 (Out of order/Maintenance action)
*Jun 18 20:22:36.655 UTC:Serial0:X.25 I R1 Clear Confirm (3) 8 lci 1
```

The table below describes significant fields shown in the display.

Table 15: debug x25 vc Field Descriptions

Field	Description
Serial0	Interface on which the X.25 event occurred.
X.25	Type of event this message describes.
O	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.

Field	Description
R1	State of the service or virtual circuit (VC). Possible values follow: R/Inactive--Packet layer awaiting link layer service R1--Packet layer ready R2--Data terminal equipment (DTE) restart request R3--DCE restart indication P/Inactive--VC awaiting packet layer service P1--Idle P2--DTE waiting for DCE to connect CALL P3--DCE waiting for DTE to accept CALL P4--Data transfer P5--CALL collision P6--DTE clear request P7--DCE clear indication D/Inactive--VC awaiting setup D1--Flow control ready D2--DTE reset request D3--DCE reset indication Refer to Annex B of the *ITU-T Recommendation X.25* for more information on these states.

Field	Description
Call	The type of X.25 packet. Possible values follow: R Events • Restart • Restart Confirm • Diagnostic P Events • Call • Call Confirm • Clear • Clear Confirm D Events • Reset • Reset Confirm D1 Events • Data • Receiver Not Ready (RNR) • RR (Receiver Ready) • Interrupt • Interrupt Confirm XOT Overhead • PVC Setup
(5)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 and 128.
lci 0	VC number. Refer to Annex A of the *ITU-T Recommendation X.25* for information on VC assignment.
From (4):2501	Source address. (4) indicates the number of digits in the address that follows. The source address is part of the address block that may be encoded in Call Setup packets.

Field	Description
To (4): 2502	Destination address. (4) indicates the number of digits in the address that follows. The destination address is part of the address block that may be encoded in Call Setup packets.
Facilities:(0)	Indicates that 0 bytes are being used to encode facilities.
Call User Data (4):	Indicates that the Call User Data (CUD) field is present and consists of 4 bytes.
0xCC000000 (ip)	Protocol identifier (PID). This subfield of the CUD field is presented in the output as a hexadecimal string followed by the name of the protocol (in this case, IP) that the string represents. Any bytes following the PID are designated "user data" and may be used by an application separately from the PID.
Cause 7	Code indicating the event that triggered the packet. The Cause field can appear only in entries for Clear, Reset, and Restart packets. Possible values for the Cause field can vary, depending on the type of packet. Refer to the appendix "X.25 Cause and Diagnostic Codes" for an explanation of these codes.
Diag 0	Code providing an additional hint of what, if anything, went wrong. The Diag field can appear only in entries for Clear, Diagnostic (as "error 0"), Reset, and Restart packets. Refer to the appendix "X.25 Cause and Diagnostic Codes" for an explanation of these codes.
(Network operational/ No additional information)	The standard explanations of the Cause and Diagnostic codes (*cause/diag*).

Related Commands

Command	Description
debug x25	Displays information about all X.25 traffic or a specific X.25 service class.
debug x25 interface	Displays information about a specific X.25 or CMNS context or virtual circuit.
debug x25 xot	Displays information about traffic to or from a specific XOT host.

debug x25 xot

To display information about traffic to or from a specific X.25 over TCP (XOT) host, use the **debug x25 xot** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug x25 xot [**remote** *ip-address* [**port** *number*]] [**local** *ip-address* [**port** *number*]] [**events**| **all**] **[dump]**

no debug x25 xot [**remote** *ip-address* [**port** *number*]] [**local** *ip-address* [**port** *number*]] [**events**| **all**] **[dump]**

Syntax Description

remote *ip-address* [**port** *number*]	(Optional) Remote IP address and, optionally, a port number. Range is from 1 to 65535.
local *ip-address* [**port** *number*]	(Optional) Local host IP address and, optionally, a port number. Range is from 1 to 65535.
events	(Optional) Displays all traffic except Data and Receiver Ready (RR) packets.
all	(Optional) Displays all traffic. This is the default.
dump	(Optional) Displays the encoded packet contents in hexadecimal and ASCII formats.

Command Default

All traffic is displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.3(2)T	The **dump** keyword was added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelin

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

The **debug x25**, **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands all generate the same basic output. The **debug x25 interface**, **debug x25 vc**, and **debug x25 xot** commands narrow the scope of the **debug x25** output to specific virtual circuits or types of traffic.

The **debug x25 xot** output allows you to restrict the debug output reporting to XOT traffic for one or both hosts or host/port combinations. Because each XOT virtual circuit uses a unique TCP connection, an XOT debug request that specifies both host addresses and ports will report traffic only for that virtual circuit. Also, you can restrict reporting to sessions initiated by the local or remote router by specifying 1998 for the remote or local port. (XOT connections are received on port 1998.)

Use the **dump** keyword to display the entire contents, including user data, of X.25 packets. The encoded X.25 packet contents are displayed after the standard packet description. The output includes the offset into the packet and the display of the data in both hexadecimal and ASCII formats.

Caution The X.25 packet information that is reported by using the **dump** keyword may contain sensitive data; for example, clear-text account identities and passwords. The network access policies and router configuration should be controlled appropriately to address this risk.

Examples

The following shows sample output from the **debug x25 xot** command:

```
Router# debug x25 xot
X.25 packet debugging is on
X.25 debug output restricted to protocol XOT
Router# show debug
X.25 (filtered for XOT):
  X.25 packet debugging is on
*Jun 18 20:32:34.699 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT I P/Inactive Call (19) 8
 lci 1
*Jun 18 20:32:34.699 UTC:  From (4):2501 To (4):2502
*Jun 18 20:32:34.699 UTC:  Facilities:(6)
*Jun 18 20:32:34.699 UTC:    Packet sizes:128 128
*Jun 18 20:32:34.699 UTC:    Window sizes:2 2
*Jun 18 20:32:34.699 UTC:  Call User Data (4):0xCC000000 (ip)
*Jun 18 20:32:34.707 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT O P3 Call Confirm (11) 8
 lci 1
*Jun 18 20:32:34.707 UTC:  From (0): To (0):
*Jun 18 20:32:34.707 UTC:  Facilities:(6)
*Jun 18 20:32:34.707 UTC:    Packet sizes:128 128
*Jun 18 20:32:34.707 UTC:    Window sizes:2 2
*Jun 18 20:32:34.715 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT I D1 Data (103) 8 lci 1
PS 0 PR 0
*Jun 18 20:32:34.723 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT O D1 Data (103) 8 lci 1
PS 0 PR 1
*Jun 18 20:32:34.731 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT I D1 Data (103) 8 lci 1
PS 1 PR 1
*Jun 18 20:32:34.739 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT O D1 Data (103) 8 lci 1
PS 1 PR 2
*Jun 18 20:32:34.747 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT I D1 Data (103) 8 lci 1
PS 2 PR 2
*Jun 18 20:32:34.755 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT O D1 Data (103) 8 lci 1
PS 2 PR 3
*Jun 18 20:32:34.763 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT I D1 Data (103) 8 lci 1
PS 3 PR 3
*Jun 18 20:32:34.771 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT O D1 Data (103) 8 lci 1
PS 3 PR 4
*Jun 18 20:32:34.779 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT I D1 Data (103) 8 lci 1
PS 4 PR 4
*Jun 18 20:32:34.787 UTC:[10.0.155.71,11001/10.0.155.70,1998]:XOT O D1 Data (103) 8 lci 1
PS 4 PR 5
```

The table below describes the significant fields shown in the display.

Table 16: debug x25 xot Field Descriptions

Field	Description
[10.0.155.71,11001/10.0.155.70,1998]	TCP connection identified by the remote IP address, remote TCP port/local IP address, local TCP port. An XOT connection is always placed to port ID 1998, so a remote port ID of 1998 implies that the router initiated the TCP connection, whereas a local port ID of 1998 implies that the router received the TCP connection.
XOT	Type of event this message describes.
I	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.
P/Inactive	State of the service or virtual circuit (VC). Possible values follow: R/Inactive--Packet layer awaiting link layer service R1--Packet layer ready R2--Data terminal equipment (DTE) restart request R3--DCE restart indication P/Inactive--VC awaiting packet layer service P1--Idle P2--DTE waiting for DCE to connect CALL P3--DCE waiting for DTE to accept CALL P4--Data transfer P5--CALL collision P6--DTE clear request P7--DCE clear indication D/Inactive--VC awaiting setup D1--Flow control ready D2--DTE reset request D3--DCE reset indication Refer to Annex B of the *ITU-T Recommendation X.25* for more information on these states.

Field	Description
Call	The type of X.25 packet. Possible values follow: R Events • Restart • Restart Confirm • Diagnostic P Events • Call • Call Confirm • Clear • Clear Confirm D Events • Reset • Reset Confirm D1 Events • Data • Receiver Not Ready (RNR) • RR (Receiver Ready) • Interrupt • Interrupt Confirm XOT Overhead • PVC Setup
(19)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 and 128.
lci 1	VC number. Refer to Annex A of the *ITU-T Recommendation X.25* for information on VC assignment.
From (4):2501	Source address. (4) indicates the number of digits in the address that follows. The source address is part of the address block that may be encoded in Call Setup packets.

Field	Description
To (4): 2502	Destination address. (4) indicates the number of digits in the address that follows. The destination address is part of the address block that may be encoded in Call Setup packets.
Facilities:(6)	Indicates that a facilities block is encoded and that it consists of 6 bytes. A breakdown of the encoded facilities follows.
Packet sizes	Encoded packet size facility settings.
Window sizes	Encoded window size facility settings.
Call User Data (4):	Indicates that the Call User Data (CUD) field is present and consists of 4 bytes.
0xCC000000 (ip)	Protocol identifier (PID). This subfield of the CUD field is presented in the output as a hexadecimal string followed by the name of the protocol (in this case, IP) that the string represents. Any bytes following the PID are designated “user data” and may be used by an application separately from the PID.
Cause 7	Code indicating the event that triggered the packet. The Cause field can appear only in entries for Clear, Reset, and Restart packets. Possible values for the Cause field can vary, depending on the type of packet. Refer to the appendix “X.25 Cause and Diagnostic Codes” for an explanation of these codes.
Diag 0	Code providing an additional hint of what, if anything, went wrong. The Diag field can appear only in entries for Clear, Diagnostic (as “error 0”), Reset, and Restart packets. Refer to the appendix “X.25 Cause and Diagnostic Codes” for an explanation of these codes.
(Network operational/ No additional information)	The standard explanations of the Cause and Diagnostic codes (*cause/diag*).

Related Commands

Command	Description
debug x25	Displays information about all X.25 traffic or a specific X.25 service class.

Command	Description
debug x25 interface	Displays information about a specific X.25 or CMNS context or virtual circuit.
debug x25 vc	Displays information about traffic for all virtual circuits that use a given number.

debug x28

To monitor error information and X.28 connection activity, use the **debug x28**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug x28

no debug x28

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output while the packet assembler/disassembler (PAD) initiates an X.28 outgoing call:

```
Router# debug x28
X28 MODE debugging is on
Router# x28
*
03:30:43: X.28 mode session started
03:30:43: X28 escape is exit
03:30:43: Speed for console & vty lines :9600
*call 123456
COM
03:39:04: address ="123456", cud="[none]" 03:39:04: Setting X.3 Parameters for this call...1:1
 2:1 3:126 4:0 5:1 6:2 7:2 8:0 9:0 10:0 11:14 12:1 13:0 14:0 15:0 16:127 17:24 18:18 19:2
20:0 21:0 22:0
Router> exit
CLR CONF
*
*03:40:50: Session ended
* exit
Router#
*03:40:51: Exiting X.28 mode
```

debug xcctsp all

To debug External Call Control Telephony Service Provider (TSP) information, use the **debug xcctsp all**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xcctsp all

no debug xcctsp all

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

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

Examples

See the following examples to turn on and off external call control debugging:

```
AS5300-TGW# debug xcctsp all
External call control all debugging is on
AS5300-TGW# no debug xcctsp all
External call control all debugging is off
AS5300-TGW#
```

Related Commands

Command	Description
debug xcctsp error	Enables debugging on external call control errors.
debug xcctsp session	Enables debugging on external call control sessions.

debug xcctsp error

To debug External Call Control Telephony Service Provider (TSP) error information, use the **debug xcctsp error**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xcctsp error

no debug xcctsp error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was integrated on the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off error-level debugging:

```
AS5300-TGW# debug xcctsp error
External call control error debugging is on
AS5300-TGW# no debug xcctsp error
External call control error debugging is off
```

Related Commands

Command	Description
debug xcctsp all	Enables debugging on all external call control levels.
debug xcctsp session	Enables debugging on external call control sessions.

debug xcctsp session

To debug External Call Control Telephony Service Provider (TSP) session information, use the **debug xcctsp session**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xcctsp session

no debug xcctsp session

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was integrated on the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off session-level debugging:

```
AS5300-TGW# debug xcctsp session
External call control session debugging is on
AS5300-TGW# no debug xcctsp session
External call control session debugging is off
AS5300-TGW#
```

Related Commands

Command	Description
debug xcctsp all	Enables debugging on external call control levels.
debug xcctsp error	Enables debugging on external call control errors.

debug xconnect

To debug a problem related to the xconnect configuration, use the **debug xconnect** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xconnect [rib] {**error**| **event**| **checkpoint**}

no debug xconnect [rib] {**error**| **event**| **checkpoint**}

Syntax Description

rib	(Optional) Displays events related to the pseudowire Routing Information Base (RIB).
error	Displays errors related to an xconnect configuration.
event	Displays events related to an xconnect configuration processing.
checkpoint	Displays the autodiscovered pseudowire information that is checkpointed to the standby Route Processor (RP).

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(23)S	This command was introduced.
12.3(2)T	This command was integrated into Cisco IOS Release 12.3(2)T.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(27)SBC	This command was integrated into Cisco IOS Release 12.2(27)SBC.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
15.1(1)S	This command was integrated into Cisco IOS Release 15.1(1)S. The **rib** and **checkpoint** keywords were added.
Cisco IOS XE Release 3.8S	This command was integrated into Cisco IOS XE Release 3.8S.

Usage Guidelines

Use this command to display debugging information about xconnect sessions.

Examples

The following is sample output from the **debug xconnect** command for an xconnect session on an Ethernet interface:

```
Router# debug xconnect event
00:01:16: XC AUTH [Et2/1, 5]: Event: start xconnect authorization, state changed from IDLE
 to AUTHORIZING
00:01:16: XC AUTH [Et2/1, 5]: Event: found xconnect authorization, state changed from
AUTHORIZING to DONE
00:01:16: XC AUTH [Et2/1, 5]: Event: free xconnect authorization request, state changed
from DONE to END
```

Related Commands

Command	Description
debug acircuit	Displays events and failures related to attachment circuits.
debug vpdn	Displays errors and events relating to L2TP configuration and the surrounding Layer 2 tunneling infrastructure.

debug xcsp

To display the debugging messages for the External Control Service Provider (XCSP) subsystem, use the **debug xcsp** command in privilegedEXEC mode. To disable debugging output, use the **no** form of this command.

debug xcsp {**all**| **cot**| **event**}

no debug xcsp {**all**| **cot**| **event**}

Syntax Description

all	Provides debug information about XCSP events and continuity testing (COT).
cot	Provides debug information about XCSP and COT. The **cot** keyword is not used with the NAS Package for Media Gateway Control Protocol (MGCP) feature.
event	Provides debug information about XCSP events.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)XB	This command was introduced.
12.2(11)T	The command was integrated into Cisco IOS Release 12.2(11)T for the Cisco AS5850.

Usage Guidelines

This command is used with the Network Access Server Package for MGCP. The XCSP subsystem is not configured directly, but information about it may be useful in troubleshooting. The **debug xcsp** command is used to display the exchange of signaling information between the MGCP protocol stack and end applications such as call switching module (CSM) or dialer.

The **cot** keyword is not used with the Network Access Server Package for MGCP feature.

Examples

The following shows sample output from the **debug xcsp all** command and keyword and the **debug xcsp event** command and keyword:

```
Router# debug xcsp all
xcsp all debugging is on
```

```
Router# debug xcsp event
xcsp events debugging is on
01:49:14:xcsp_call_msg:Event  Call Indication , channel state =  Idle  for
slot port channel 7
c5400# 0 23
01:49:14:xcsp_process_sig_fsm:state/event  Idle / Call Indication
01:49:14:xcsp_incall:
01:49:14:xcsp_incall CONNECT_IND:cdn=3000 cgn=1000
01:49:14:xcsp:START guard TIMER
01:49:14:xcsp_fsm:slot 7 port 0 chan 23 oldstate =  Idle  newstate= Connection
in progress mgcpapp_process_mgcp_msg PROCESSED NAS PACKAGE EVENT
01:49:14:Received message on XCSP_CDAPI
01:49:14:process_cdapi_msg :slot/port/channel 7/0/23
01:49:14: process_cdapi_msg:new  slot/port/channel 7/0/23
01:49:14:
c5400#Received CONN_RESP:callid=0x7016
01:49:14:process_cdapi:Event CONN_RESP, channel state = 8 for slot port
channel 7 0 23
01:49:14:xcsp_process_sig_fsm:state/event  Connection in progress / In Call
accept
 mgcpapp_xcsp_alert:
 mgcpapp_xcsp_get_chan_cb -Found - Channel state  Connection in progress
200 58  Alert
I:630AED90
<---:Ack send SUCCESSFUL
01:49:14:xcsp_fsm:slot 7 p
c5400#ort 0 chan 23 oldstate =  Connection in progress  newstate= Connection in
progress
01:49:14:Received message on XCSP_CDAPI
01:49:14:process_cdapi_msg :slot/port/channel 7/0/23
01:49:14: process_cdapi_msg:new  slot/port/channel 7/0/23
01:49:14: Received CALL_CONN:callid=0x7016
01:49:14:process_cdapi:Event CONN_, channel state = 8 for slot port channel 7
0 23
01:49:14:xcsp_process_sig_fsm:state/event  Connection in progress / in call
connect
 mgcpapp_xcsp_connect:
 mgcpapp_xc
c5400#sp_get_chan_cb -Found - Channel state In Use
01:49:14:STOP TIMER
01:49:14:xcsp_fsm:slot 7 port 0 chan 23 oldstate =  Connection in progress
newstate=In Use
c5400#
01:50:23:Received message on XCSP_CDAPI
01:50:23:process_cdapi_msg :slot/port/channel 7/0/23
01:50:23: process_cdapi_msg:new  slot/port/channel 7/0/23
01:50:23: Received CALL_DISC_REQ:callid=0x7016
01:50:23:process_cdapi:Event DISC_CONN_REQ, channel state = 7 for slot port
channel 7 0 23
01:50:23:xcsp_process_sig_fsm:state/event In Use / release Request
 mgcpapp_xcsp_disconnect
 mgcpapp_xcsp_get_chan_cb -Fou
c5400#nd - Channel state In Use
01:50:23:send_mgcp_msg, MGCP Packet sent --->
01:50:23:RSIP 1 *@c5400 MGCP 1.0
RM:restart
DLCX 4 S7/DS1-0/23 MGCP 1.0
C:3
I:630AED90
E:801 /NAS User request
01:50:23:xcsp_fsm:slot 7 port 0 chan 23 oldstate = In Use  newstate=Out
Release in progress
 xcsp_restart Serial7/0:22 vc = 22
 xcsp_restart Put idb Serial7/0:22 in down state
01:50:23:MGCP Packet received -
200 4 bye
 Data call ack received callp=0x62AEEA70mgcpapp_xcsp
c5400#_ack_recv:mgcpapp_xcsp_get_chan_cb -Found - Channel state Out Release in
progress
mgcpapp_xcsp_ack_recv ACK 200 rcvd:transaction id = 4 endpt=S7/DS1-0/23
01:50:23:xcsp_call_msg:Event  Release confirm , channel state = Out Release in
progress for slot port channel 7 0 23
01:50:23:xcsp_process_sig_fsm:state/event Out Release in progress/ Release
```

```
confirm
01:50:23:STOP TIMER
01:50:23:xcsp_fsm:slot 7 port 0 chan 23 oldstate = Out Release in progress
newstate= Idle
```

Related Commands

Command	Description
show vrm vdevices	Displays the status of a router port under the control of the XCSP subsystem.
show xcsp slot	Displays the status of a router slot under the control of the XCSP subsystem.

debug xdsl application

To monitor the xDSL if the digital subscriber line (DSL) does not come up, use the debug xdsl application command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xdsl application

no debug xdsl application

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)XD	This command was introduced on Cisco 2600 series and Cisco 3700 series routers.
12.3(4)XG	Support was added for the Cisco 1700 series routers.
12.3(7)T	This command was integrated into Cisco IOS Release 12.3(7)T on Cisco 2600 series, Cisco 3631, and Cisco 3700 series routers.
12.3(11)T	Support was added for the Cisco 2800 and Cisco 3800 series routers.
12.3(14)T	Support was added for the Cisco 1800 series routers.

Usage Guidelines

The debug xdsl application command details what occurs during the Cisco IOS SHDSL process events and signal-to-noise ratio sampling of the SHDSL chip. This information can be used more for software debugging in analyzing the internal events.

Examples

The following is sample output from the **debug xdsl application** command:

```
Router# debug xdsl application

xDSL application debugging is on
Router#
```

The following lines show that the application is starting on the router and waiting for a response:

```
00:47:40: DSL 0/0 process_get_wakeup
00:47:41: DSL 0/0 process_get_wakeup
00:47:42: DSL 0/0 process_get_wakeup
00:47:43: DSL 0/0 process_get_wakeup
00:47:44: DSL 0/0 process_get_wakeup
00:47:45: DSL 0/0 process_get_wakeup
00:47:46: DSL 0/0 process_get_wakeup
00:47:47: DSL 0/0 process_get_wakeup
00:47:48: DSL 0/0 process_get_wakeup
```

```
00:47:49: DSL 0/0 process_get_wakeup
00:47:49: DSL 0/0 process_get_wakeup
```

The following lines show that the controller link comes up:

```
00:47:49: DSL 0/0 xdsl_background_process: XDSL link up boolean event received
00:47:49:  DSL 0/0 controller Link up! line rate: 1600 Kbps
```

The following lines show that the DSL controller comes up:

```
00:47:49: DSL 0/0 xdsl_controller_reset: cdb-state=up
00:47:49: %CONTROLLER-5-UPDOWN: Controller DSL 0/0, changed state to up
00:47:49:  Dslsar data rate 1600
00:47:49: DSL 0/0 TipRing 1, Xmit_Power Val 75, xmit_power 7.5
00:47:49: DSL 0/0 Mode 2, BW 1600, power_base_value 135, power_backoff 6
00:47:50: DSL 0/0 process_get_wakeup
00:47:51: DSL 0/0 process_get_wakeup
00:47:52: DSL 0/0 process_get_wakeup
00:47:53: DSL 0/0 process_get_wakeup
00:47:54: DSL 0/0 process_get_wakeup
00:47:55: DSL 0/0 process_get_wakeup
00:47:56: DSL 0/0 process_get_wakeup
```

The following lines show signal-to-noise ratio sampling:

```
00:47:56: DSL 0/0   SNR Sampling: 42 dB
00:47:57: DSL 0/0 process_get_wakeup
00:47:57: DSL 0/0   SNR Sampling: 41 dB
00:47:58: DSL 0/0 process_get_wakeup
00:47:58: DSL 0/0   SNR Sampling: 40 dB
00:47:59: DSL 0/0 process_get_wakeup
00:47:59: DSL 0/0   SNR Sampling: 40 dB
00:48:00: DSL 0/0 process_get_wakeup
00:48:00: DSL 0/0   SNR Sampling: 39 dB
00:48:01: DSL 0/0 process_get_wakeup
00:48:01: DSL 0/0   SNR Sampling: 39 dB
00:48:02: DSL 0/0 process_get_wakeup
00:48:02: DSL 0/0   SNR Sampling: 38 dB
00:48:03: DSL 0/0 process_get_wakeup
00:48:03: DSL 0/0   SNR Sampling: 38 dB
00:48:04: DSL 0/0 process_get_wakeup
00:48:04: DSL 0/0   SNR Sampling: 38 dB
00:48:05: DSL 0/0 process_get_wakeup
00:48:05: DSL 0/0   SNR Sampling: 37 dB
00:48:06: DSL 0/0 process_get_wakeup
00:48:06: DSL 0/0   SNR Sampling: 37 dB
00:48:07: DSL 0/0 process_get_wakeup
00:48:07: DSL 0/0   SNR Sampling: 36 dB
```

The following lines show that the link comes up:

```
00:48:07: %LINK-3-UPDOWN: Interface ATM0/0, changed state to up
00:48:08: DSL 0/0 process_get_wakeup
00:48:08: DSL 0/0   SNR Sampling: 36 dB
```

The following lines show that the line protocol comes up:

```
00:48:08: %LINEPROTO-5-UPDOWN: Line protocol on Interface ATM0/0, changed state to up
00:48:09: DSL 0/0 process_get_wakeup
00:48:09: DSL 0/0   SNR Sampling: 36 dB
00:48:10: DSL 0/0 process_get_wakeup
00:48:10: DSL 0/0   SNR Sampling: 36 dB
00:48:11: DSL 0/0 process_get_wakeup
00:48:11: DSL 0/0   SNR Sampling: 35 dB
00:48:12: DSL 0/0 process_get_wakeup
00:48:12: DSL 0/0   SNR Sampling: 36 dB
00:48:13: DSL 0/0 process_get_wakeup
00:48:13: DSL 0/0   SNR Sampling: 36 dB
00:48:14: DSL 0/0 process_get_wakeup
00:48:14: DSL 0/0   SNR Sampling: 36 dB
00:48:15: DSL 0/0 process_get_wakeup
00:48:15: DSL 0/0   SNR Sampling: 36 dB
00:48:16: DSL 0/0 process_get_wakeup
```

```
00:48:16: DSL 0/0   SNR Sampling: 36 dB
00:48:17: DSL 0/0 process_get_wakeup
00:48:17: DSL 0/0   SNR Sampling: 35 dB
00:48:18: DSL 0/0 process_get_wakeup
00:48:18: DSL 0/0   SNR Sampling: 35 dB
00:48:19: DSL 0/0 process_get_wakeup
```

Related Commands

Command	Description
debug xdsl driver	Monitors what is happening when downloading and installing the drivers.
debug xdsl eoc	Monitors what is in the embedded operations channel messages.
debug xdsl error	Monitors the errors of the xDSL process and firmware.

debug xdsl driver

To display what is happening when the drivers are downloaded and installed, use the **debug xdsl driver** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xdsl driver

no debug xdsl driver

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)XD	This command was introduced on Cisco 2600 series and Cisco 3700 series routers.
12.3(4)XG	Support was added for Cisco 1700 series routers.
12.3(7)T	This command was integrated into Cisco IOS Release 12.3(7)T on Cisco 2600 series, Cisco 3631, and Cisco 3700 series routers.
12.3(11)T	Support was added for the Cisco 2800 and Cisco 3800 series routers.
12.3(14)T	Support was added for Cisco 1800 series routers.

Usage Guidelines

Use the **debug xdsl driver** command to monitor what is happening when downloading the firmware. This debugging command displays the Globespan DSL Driver details and provides framer interrupt information and line training failure information. This information can help you understand the problems faced while downloading the firmware, why the line went down, and so forth.

Examples

The following is sample output from the **debug xdsl driver**command:

```
Router# debug xdsl driver
xDSL driver debugging is on
```

The following lines show that the DSP interrupt download is running:

```
*Mar 12 08:01:04.772: DSL 0/2  dsp interrupt-download next block for line-0
*Mar 12 08:01:04.780: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:05.072: DSL 0/2  dsp interrupt-download next block for line-0
*Mar 12 08:01:05.080: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:06.484: DSL 0/2  dsp interrupt-download next block for line-0
*Mar 12 08:01:06.492: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:08.092: DSL 0/2  dsp interrupt-download next block for line-0
*Mar 12 08:01:08.096: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:19.180: DSL 0/2  dsp interrupt-download next block for line-0
```

```
*Mar 12 08:01:19.184: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:19.480: DSL 0/2  dsp interrupt-download next block for line-0
*Mar 12 08:01:19.484: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:19.680: DSL 0/2  dsp interrupt-download next block for line-0
```

The following lines show that the DSP interrupt has been disabled and that the framer interrupt has been enabled:

```
*Mar 12 08:01:19.680: DSL 0/2 DSP interrupt disabled
*Mar 12 08:01:19.680: DSL 0/2  Download completed for line-0
*Mar 12 08:01:19.680: DSL 0/2 Framer interrupt enabled
*Mar 12 08:01:19.680: DSL 0/2 framer intr_status 0xC0
*Mar 12 08:01:19.680:  DSL 0/2 controller Link up! line rate: 2304 Kbps
```

The following lines show that the digital subscriber line (DSL) controller has come up on slot 0 and port 2:

```
*Mar 12 08:01:19.680: %CONTROLLER-5-UPDOWN: Controller DSL 0/2, changed state to up
*Mar 12 08:01:19.680:  Dslsar data rate 2304
*Mar 12 08:01:22.528: %LINK-3-UPDOWN: Interface ATM0/2, changed state to up
*Mar 12 08:01:23.528: %LINEPROTO-5-UPDOWN: Line protocol on Interface ATM0/2, changed state
 to up
```

The following lines show that the framer interrupt status is running:

```
*Mar 12 08:01:23.812: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:23.816: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:23.904: DSL 0/2 framer intr_status 0xC1
*Mar 12 08:01:28.612: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:28.616: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:28.708: DSL 0/2 framer intr_status 0xC1
*Mar 12 08:01:28.804: DSL 0/2 framer intr_status 0xC1
*Mar 12 08:01:33.412: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:33.420: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:33.508: DSL 0/2 framer intr_status 0xC1
*Mar 12 08:01:33.604: DSL 0/2 framer intr_status 0xC1
*Mar 12 08:01:33.700: DSL 0/2 framer intr_status 0xC1
*Mar 12 08:01:38.212: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:38.220: DSL 0/2 framer intr_status 0xC4
*Mar 12 08:01:38.308: DSL 0/2 framer intr_status 0xC1
```

Related Commands

Command	Description
debug xdsl application	Monitors the xDSL if the DSL does not come up.
debug xdsl eoc	Monitors what is in the embedded operations channel messages.
debug xdsl error	Monitors the errors of the xDSL process and firmware.

debug xdsl eoc

To display the flow of the embedded operations channel (EOC) messages received, processed, and transmitted, use the **debug xdsl eoc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xdsl eoc

no debug xdsl eoc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)XD	This command was introduced on Cisco 2600 series and Cisco 3700 series routers.
12.3(4)XG	This command was integrated into the Cisco IOS Release 12.3(4)XG on the Cisco 1700 series routers.
12.3(7)T	This command was integrated into Cisco IOS Release 12.3(7)T on Cisco 2600 series, Cisco 3631, and Cisco 3700 series routers.
12.3(11)T	This command was implemented on Cisco 2800 series and Cisco 3800 series routers.
12.3(14)T	This command was implemented on Cisco 1800 series routers.

Usage Guidelines

Use the **debug xdsl eoc** command to review the contents of the embedded operations channel messages.

Examples

The following is sample output from the **debug xdsl eoc** command:

```
Router# debug xdsl eoc

xDSL EOC debugging is on
Router#
```

The following lines show the embedded operations channel message being received and copied to the buffer. The xdsl_background_process is performed. The data_transparency_remove is performed.

```
00:02:55:   Incoming EOC received
00:02:55:   Copy the EOC to buffer
00:02:55:   Incoming EOC received
00:02:55:   Copy the EOC to buffer
00:02:55:  End of EOC received, Notify task
00:02:55: xdsl_background_process:
```

```
00:02:55:  Rx EOC remove transparency:: 12 C  A  63
00:02:55: data_transparency_remove: Done, eoc packet size = 4
```

The following lines show that the packet of the embedded operations channel messages was received and verified as good. The data_transparency_add is performed.

```
00:02:55:   Good eoc packet received
00:02:55:  incoming request eocmsgid: 12
00:02:55:  Tx Converted EOC message:: 21 8C 0  28 0  0  0  0  0  0  0  1  1  713
00:02:55: data_transparency_add: eoc packet size - before 15, after 15
```

The following lines show another embedded operations channel message coming in and being copied to the buffer. The xdsl_background_process is run on this message as before.

```
00:02:55:  size of eoc status response :: 13
00:02:56:   Incoming EOC received
00:02:56:   Copy the EOC to buffer
00:02:56:   Incoming EOC received
00:02:56:   Copy the EOC to buffer
00:02:56:  End of EOC received, Notify task
00:02:56: xdsl_background_process:
00:02:56:  Rx EOC remove transparency:: 12 C  A  63
00:02:56: data_transparency_remove: Done, eoc packet size = 4
```

Related Commands

Command	Description
debug xdsl application	Displays status of the xDSL if the DSL does not activate as expected.
debug xdsl driver	Diaplays status when the drivers are downloaded and installed.
debug xdsl error	Displays the errors of the xDSL process and firmware.

debug xdsl error

To display the errors of xDSL process and firmware, use the **debug xdsl error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug xdsl error

no debug xdsl error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)XD	This command was introduced on Cisco 2600 series and Cisco 3700 series routers.
12.3(4)XG	Support was added for the Cisco 1700 series routers.
12.3(7)T	This command was integrated into Cisco IOS Release 12.3(7)T on Cisco 2600 series, Cisco 3631, and Cisco 3700 series routers.
12.3(11)T	Support was added for the Cisco 2800 and Cisco 3800 series routers.
12.3(14)T	Support was added for Cisco 1800 series routers.

Usage Guidelines

Use the **debug xdsl error** command to display the errors during driver initialization and any Globespan firmware API failures.

Examples

The following is sample output from the **debug xdsl error** command. When the debug is enabled, a message indicates that DSL error debugging is on.

```
Router# debug xdsl error

xDSL error debugging is on
Router#
```

Related Commands

Command	Description
debug xdsl application	Monitors the xDSL if the DSL does not come up.
debug xdsl driver	Monitors what is happening when downloading and installing the drivers.

Command	Description
debug xdsl eoc	Monitors what is in the embedded operations channel messages.

debug zone

To display zone security event debugs, use the **debug zone**command in privileged EXEC mode. To disable the debugging messages, use the **no** form of this command.

debug zone security {**events**| **object-creation**| **object-deletion**}

no debug zone security {**events**| **object-creation**| **object-deletion**}

Syntax Description

security	Displays security events debug messages.
events	Displays zone security events debug messages.
object-creation	Displays zone security object creation debug messages.
object-deletion	Displays zone security object deletion debug messages.

Command Default

By default, debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(6)T	This command was introduced.

Examples

If the **debug zone security events**command is enabled and a zone event occurs, firewall generates debug messages. An event can be a zone or zone pair creation and deletion.

```
Router# show debug
zone:
Zone security Events debugging is on
Router# configure terminal
Router(config)# zone security public
Router(config-sec-zone)#
*Jan 29 05:04:52.967: ZONE_SEC:zone added
Router(config-sec-zone)# zone security private
Router(config-sec-zone)#
*Jan 29 05:05:02.999: ZONE_SEC:zone added
Router(config-sec-zone)# exit

Router(config)# zone-pair security pu2pr source public destination private

Router(config-sec-zone-pair)#
*Jan 29 05:05:37.575: ZONE_SEC:zone-pair added
```

```
*Jan 29 05:05:37.575: ZONE_SEC:allocating zone-pair
Router(config)# no zone-pair security pu2pr source public destination private
Router(config)#
*Jan 29 05:08:00.667: ZONE_SEC:zone-pair deleting...
Router(config)# no zone security public
Router(config)#
*Jan 29 05:08:12.135: ZONE_SEC:zone deleting..
Router(config)# no zone security private
Router(config)#
*Jan 29 05:08:18.243: ZONE_SEC:zone deleting..
```

If the **debug zone security object-creation** and the **debug zone security object-deletion** commands are enabled and when zones or zone pairs are created or deleted, firewall generates debug messages.

```
Router# show debugging

zone:
Zone security Object Creations debugging is on
Zone security Object Deletions debugging is on
Router# configure terminal
Router(config)# zone security public

Router(config-sec-zone)#
*Jan 29 05:09:28.207: ZONE_SEC: zone public created
Router(config-sec-zone)# exit
Router(config)# zone security private
Router(config-sec-zone)#
*Jan 29 05:09:50.831: ZONE_SEC: zone private created
Router(config-sec-zone)# exit
Router(config)# zone-pair security zp source public destination private
Router(config-sec-zone-pair)#
*Jan 29 05:10:22.063: ZONE_SEC: zone-pair zp created
Router(config-sec-zone-pair)# service-policy type inspect pmap
Router(config-sec-zone-pair)#
*Jan 29 05:10:36.787: ZONE_SEC: zone-pair FW_INT_REV_zp_3748291079 created
Router(config-sec-zone-pair)# no service-policy type inspect pmap
Router(config-sec-zone-pair)# exit
Router(config)# no zone-pair security zp source public destination private
Router(config)#
*Jan 29 05:11:04.043: ZONE_SEC: zone-pair zp deleted
Router(config)# no zone security public
Router(config)#
*Jan 29 05:11:10.875: ZONE_SEC: zone public deleted
Router(config)# no zone security private
Router(config)#
*Jan 29 05:11:16.931: ZONE_SEC: zone private deleted
Router(config)# end
Router#
```

Related Commands

Command	Description
zone security	Creates a security zone.
zone-pair security	Creates a zone pair.

show memory debug incremental

To display information about memory leaks after a starting time has been established, use the **show memory debug incremental** command in privileged EXEC mode.

show memory debug incremental {**allocations**| **leaks**[**lowmem**| **summary**]| **status**}

Syntax Description

allocations	Displays all memory blocks that were allocated after issuing the **set memory debug incremental starting-time** command.
leaks	Displays only memory that was leaked after issuing the **set memory debug incremental starting-time** command.
lowmem	(Optional) Forces the memory leak detector to work in low memory mode, making no memory allocations.
summary	(Optional) Reports summarized memory leaks based on allocator_pc and size of the memory block.
status	Displays all memory blocks that were allocated after issuing the **set memory debug incremental starting-time** command.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(7)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.4T	The summary keyword was added.

Usage Guidelines

The **show memory debug incremental allocations** command displays all the memory blocks that were allocated after the **set memory debug incremental starting-time** command was entered. The displayed memory blocks are just memory allocations, they are not necessarily leaks.

The **show memory debug incremental leaks** command provides output similar to the **show memory debug leaks** command, except that it displays only memory that was leaked after the **set memory debug incremental starting-time** command was entered.

The **show memory debug incremental leaks lowmem** command forces memory leak detection to work in low memory mode. The amount of time taken for analysis is considerably greater than that of normal mode. The output for this command is similar to the **show memory debug leaks** command, except that it displays only memory that was leaked after the **set memory debug incremental starting-time** command was entered. You can use this command when you already know that normal mode memory leak detection will fail (perhaps by an unsuccessful previous attempt to invoke normal mode memory leak detection).

The **show memory debug incremental leaks summary** command displays a summarized report of the memory that was leaked after the **set memory debug incremental starting-time** command was entered, ordered by allocator process call address (Alloc_pc) and by memory block size.

The **show memory debug incremental status** command displays whether a starting point for incremental analysis has been set and the elapsed time since then.

Note All show memory debug commands must be used on customer networks only to diagnose the router for memory leaks when memory depletion is observed. These CLI's will have high CPU utilization and might result in time sensitive protocols to flap. These CLI's are recommended for customer use, only in the maintenance window when the router is not in a scaled condition.

Note All memory leak detection commands invoke normal mode memory leak detection, except when the low memory option is specifically invoked by use of the **lowmem** keyword. In normal mode, if memory leak detection determines that there is insufficient memory to proceed in normal mode, it will display an appropriate message and switch to low memory mode.

Examples

Examples

The following example shows output from the **show memory debug incremental**command when entered with the **allocations** keyword:

```
Router# show memory debug incremental allocations
Address    Size   Alloc_pc  PID  Name
62DA4E98      176 608CDC7C  44   CDP Protocol
62DA4F48       88 608CCCC8  44   CDP Protocol
62DA4FA0       88 606224A0  3    Exec
62DA4FF8       96 606224A0  3    Exec
635BF040       96 606224A0  3    Exec
63905E50      200 606A4DA4  69   Process Events
```

Examples

The following example shows output from the **show memory debug incremental** command when entered with the **leaks** and **summary** keywords:

```
Router# show memory debug incremental leaks summary
Adding blocks for GD...
                PCI memory
Alloc PC     Size     Blocks      Bytes    What
                I/O memory
Alloc PC     Size     Blocks      Bytes    What
                Processor memory
Alloc PC     Size          Blocks       Bytes      What
0x60874198  0000000052  0000000001  0000000052    Exec
0x60874198  0000000060  0000000001  0000000060    Exec
0x60874198  0000000100  0000000001  0000000100    Exec
```

```
0x60874228  0000000052  0000000004  0000000208    Exec
0x60874228  0000000060  0000000002  0000000120    Exec
0x60874228  0000000100  0000000004  0000000400   Exec
```

Examples

The following example shows output from the **show memory debug incremental** command entered with the **status** keyword:

```
Router# show memory debug incremental status
Incremental debugging is enabled
Time elapsed since start of incremental debugging: 00:00:10
```

Related Commands

Command	Description
set memory debug incremental starting-time	Sets the current time as the starting time for incremental analysis.
show memory debug leaks	Displays detected memory leaks.

set memory debug incremental starting-time

To set the current time as the starting time for incremental analysis, use the **set memory debug incremental starting-time** command in privileged EXEC mode.

set memory debug incremental starting-time [none]

Syntax Description

none	(Optional) Resets the defined start time for incremental analysis.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T1	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

For incremental analysis, a starting point can be defined by using the **set memory debug incremental starting-time** command. When a starting time is set, only memory allocated after that starting time will be considered for reporting as leaks.

Examples

The following example shows the command used to set the starting time for incremental analysis to the time when the command was issued:

```
Router# set memory debug incremental starting-time
```

Related Commands

Command	Description
show memory debug incremental allocation	Displays all memory blocks that were allocated after the issue of the **set memory debug incremental starting-time** command.

Command	Description
show memory debug incremental leaks	Displays only memory that was leaked after the issue of the **set memory debug incremental starting-time** command.
show memory debug incremental leaks lowmem	Forces incremental memory leak detection to work in low memory mode. Displays only memory that was leaked after the issue of the **set memory debug incremental starting-time** command.
show memory debug incremental status	Displays if the starting point of incremental analysis has been defined and the time elapsed since then.
show memory debug leaks	Displays detected memory leaks.

show memory debug leaks

To display detected memory leaks, use the **show memory debug leaks** command in privileged EXEC mode.

Cisco IOS software

show memory debug leaks [**chunks**| **largest**| **lowmem**| **summary**]

Cisco Catalyst 4500e Series Switches running IOS XE software

show memory debug leak

Syntax Description

chunks	(Optional) Displays the memory leaks in chunks.
largest	(Optional) Displays the top ten leaking allocator_pcs based on size, and the total amount of memory they have leaked.
lowmem	(Optional) Forces the memory leak detector to work in low memory mode, making no memory allocations.
summary	(Optional) Reports summarized memory leaks based on allocator_pc and size of the memory block.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T1	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
Cisco IOS XE Release 3.1.0.SG	This command was introduced on the Cisco Catalyst 4500e Serfies Switches to display per-process memory leak ammounts.

Usage Guidelines

If no optional keywords are specified, the **show memory debug leaks** command invokes normal mode memory leak detection and does not look for memory leaks in chunks.

The **show memory debug leaks chunks** command invokes normal mode memory leak detection and looks for leaks in chunks as well.

The **show memory debug leaks largest** command displays the top ten leaking allocator_pcs and the total amount of memory that they have leaked. Additionally, each time this command is invoked it remembers the previous invocation's report and compares it to the current invocation's report. If there are new entries in the current report they are tagged as “inconclusive.” If the same entry appears in the previous invocation's report and the current invocation's report, the inconclusive tag is not added. It would be beneficial to run memory leak detection more than once and to consider only the consistently reported leaks.

The **show memory debug leaks lowmem** command forces memory leak detection to work in low memory mode. The amount of time taken for analysis is considerably greater than that of normal mode. The output for this command is similar to the **show memory debug leaks** command. You can use this command when you already know that normal mode memory leak detection will fail (perhaps by an unsuccessful previous attempt to invoke normal mode memory leak detection).

The **show memory debug leaks summary** command reports memory leaks based on allocator_pc and then on the size of the block.

Note All show memory debug commands must be used on customer networks only to diagnose the router for memory leaks when memory depletion is observed. These CLI’s will have high CPU utilization and might result in time sensitive protocols to flap. These CLI’s are recommended for customer use, only in the maintenance window when the router is not in a scaled condition.

Note The command **show memory debug leak lowmem** is extremely CPU intensive and can result in CPUHOG/WATCHDOG crash. This command must be used only when the router has reached an unusable state due to memory exhaustion. Its use on high end platforms such as ISR and above can potentially crash the box. Use outside of these limitations can cause a console hang of 1 hour in some cases. As an alternative, use the **show memory debug leak** command.

Examples

Example output varies between Cisco IOS software images and Cisco IOS Software Modularity software images. To view the appropriate output, choose one of the following sections:

- show memory debug leaks
- show memory debug leaks

Examples

Examples

The following example shows output from the **show memory debug leaks** command:

```
Router# show memory debug leaks
Adding blocks for GD...
                PCI memory
Address    Size   Alloc_pc  PID  Name
                I/O memory
Address    Size   Alloc_pc  PID  Name
                Processor memory
Address    Size   Alloc_pc  PID  Name
62DABD28       80 60616750  -2   Init
62DABD78       80 606167A0  -2   Init
62DCF240       88 605B7E70  -2   Init
62DCF298       96 605B7E98  -2   Init
```

```
62DCF2F8       88 605B7EB4  -2   Init
62DCF350       96 605B7EDC  -2   Init
63336C28      104 60C67D74  -2   Init
63370D58       96 60C656AC  -2   Init
633710A0      304 60C656AC  -2   Init
63B2BF68       96 60C659D4  -2   Init
63BA3FE0    32832 608D2848  104  Audit Process
63BB4020    32832 608D2FD8  104  Audit Process
```

The table below describes the significant fields shown in the display.

Table 17: show memory debug leaks Field Descriptions

Field	Description
Address	Hexadecimal address of the leaked block.
Size	Size of the leaked block (in bytes).
Alloc_pc	Address of the system call that allocated the block.
PID	The process identifier of the process that allocated the block.
Name	The name of the process that allocated the block.

Examples

The following example shows output from the **show memory debug leaks chunks** command:

```
Router# show memory debug leaks chunks
Adding blocks for GD...
                PCI memory
Address    Size   Alloc_pc  PID  Name
Chunk Elements:
Address  Size  Parent   Name
                I/O memory
Address    Size   Alloc_pc  PID  Name
Chunk Elements:
Address  Size  Parent   Name
                Processor memory
Address    Size   Alloc_pc  PID  Name
62DABD28       80 60616750  -2   Init
62DABD78       80 606167A0  -2   Init
62DCF240       88 605B7E70  -2   Init
62DCF298       96 605B7E98  -2   Init
62DCF2F8       88 605B7EB4  -2   Init
62DCF350       96 605B7EDC  -2   Init
63336C28      104 60C67D74  -2   Init
63370D58       96 60C656AC  -2   Init
633710A0      304 60C656AC  -2   Init
63B2BF68       96 60C659D4  -2   Init
63BA3FE0    32832 608D2848  104  Audit Process
63BB4020    32832 608D2FD8  104  Audit Process
Chunk Elements:
Address  Size  Parent   Name
62D80DA8    16 62D7BFD0 (Managed Chunk )
62D80DB8    16 62D7BFD0 (Managed Chunk )
62D80DC8    16 62D7BFD0 (Managed Chunk )
62D80DD8    16 62D7BFD0 (Managed Chunk )
62D80DE8    16 62D7BFD0 (Managed Chunk )
62E8FD60   216 62E8F888 (IPC Message He)
```

The table below describes the significant fields shown in the display.

Table 18: show memory debug leaks chunks Field Descriptions

Field	Description
Address	Hexadecimal address of the leaked block.
Size	Size of the leaked block (in bytes).
Alloc_pc	Address of the system call that allocated the block.
PID	The process identifier of the process that allocated the block.
Name	The name of the process that allocated the block.
Size	(Chunk Elements) Size of the leaked element (bytes).
Parent	(Chunk Elements) Parent chunk of the leaked chunk.
Name	(Chunk Elements) The name of the leaked chunk.

Examples

The following example shows output from the **show memory debug leaks largest** command:

```
Router# show memory debug leaks largest
Adding blocks for GD...
                PCI memory
Alloc_pc    total leak size
                I/O memory
Alloc_pc    total leak size
                Processor memory
Alloc_pc    total leak size
608D2848    32776     inconclusive
608D2FD8    32776     inconclusive
60C656AC    288       inconclusive
60C67D74    48        inconclusive
605B7E98    40        inconclusive
605B7EDC    40        inconclusive
60C659D4    40        inconclusive
605B7E70    32        inconclusive
605B7EB4    32        inconclusive
60616750    24        inconclusive
```

The following example shows output from the second invocation of the **show memory debug leaks largest** command:

```
Router# show memory debug leaks largest
Adding blocks for GD...
                PCI memory
Alloc_pc    total leak size
                I/O memory
Alloc_pc    total leak size
                Processor memory
Alloc_pc    total leak size
608D2848    32776
608D2FD8    32776
60C656AC    288
60C67D74    48
605B7E98    40
605B7EDC    40
```

```
60C659D4    40
605B7E70    32
605B7EB4    32
60616750    24
```

Examples The following example shows output from the **show memory debug leaks summary** command:

```
Router# show memory debug leaks summary
Adding blocks for GD...
                PCI memory
Alloc PC        Size     Blocks      Bytes    What
                I/O memory
Alloc PC        Size     Blocks      Bytes    What
                Processor memory
Alloc PC        Size     Blocks      Bytes    What
0x605B7E70 0000000032 0000000001 0000000032    Init
0x605B7E98 0000000040 0000000001 0000000040    Init
0x605B7EB4 0000000032 0000000001 0000000032    Init
0x605B7EDC 0000000040 0000000001 0000000040    Init
0x60616750 0000000024 0000000001 0000000024    Init
0x606167A0 0000000024 0000000001 0000000024    Init
0x608D2848 0000032776 0000000001 0000032776    Audit Process
0x608D2FD8 0000032776 0000000001 0000032776    Audit Process
0x60C656AC 0000000040 0000000001 0000000040    Init
0x60C656AC 0000000248 0000000001 0000000248    Init
0x60C659D4 0000000040 0000000001 0000000040    Init
0x60C67D74 0000000048 0000000001 0000000048    Init
```

The table below describes the significant fields shown in the display.

Table 19: show memory debug leaks summary Field Descriptions

Field	Description
Alloc_pc	Address of the system call that allocated the block.
Size	Size of the leaked block.
Blocks	Number of blocks leaked.
Bytes	Total amount of memory leaked.
What	Name of the process that owns the block.

Examples

Examples The following example shows output from the **show memory debug leak** command on command on a Cisco Catalyst 4500e switch, using a Cisco IOS image from Cisco IOS XE Release 3.1.0.SG and later releases:

```
Switch#show memory debug leak
System memory  : 1943928K total, 735154K used, 1208774K free, 153224K kernel reserved
Lowest(b)      : 641564672
Process iosd, type L, PID = 10319
    1012856K total, 67716K text, 798420K data, 84K stack, 252K dynamic
    252 heapsize, 252 allocated, 0 free
Adding blocks for GD...
Leak(b)               PID               Name
368                   10319             iosd
Switch#
```

The table below describes the significant fields shown in the display.

Table 20: show memory debug leaks summary Field Descriptions

Field	Description
Leak	Size of the leaked block.
PID	The process identifier of the process that allocated the block.
Name	Name of the process that owns the block.

Related Commands

Command	Description
set memory debug incremental starting-time	Sets the current time as the starting time for incremental analysis.
show memory debug incremental allocation	Displays all memory blocks that were allocated after the issue of the **set memory debug incremental starting-time** command.
show memory debug incremental leaks	Displays only memory that was leaked after the issue of the **set memory debug incremental starting-time** command.
show memory debug incremental leaks lowmem	Forces incremental memory leak detection to work in low memory mode. Displays only memory that was leaked after the issue of the **set memory debug incremental starting-time** command.
show memory debug incremental status	Displays if the starting point of incremental analysis has been defined and the time elapsed since then.

show memory debug references

To display debug information on references, use the **show memory debug references**command in user EXEC or privileged EXEC mode.

show memory debug references [**dangling** [*start-address start-address*]]

Syntax Description

dangling	(Optional) Displays the possible references to free memory.
start-address	(Optional) Address numbers <0-4294967295> that determine the address range.

Command Modes

User EXEC Privileged EXEC

Command History

Release	Modification
12.0	This command was introduced.

Usage Guidelines

All show memory debug commands must be used on customer networks only to diagnose the router for memory leaks when memory depletion is observed. These CLI's will have high CPU utilization and might result in time sensitive protocols to flap. These CLI's are recommended for customer use, only in the maintenance window when the router is not in a scaled condition.

Examples

The following is sample output from the **show memory debug references**command:

```
Router# show memory debug references 2 3
Address  Reference Cont_block Cont_block_name
442850BC        2  44284960   bss
44285110        3  44284960   bss
4429C33C        2  44284960   bss
4429C34C        2  44284960   bss
4429C35C        3  44284960   bss
.
.
.
```

The following is sample output from the **show memory debug references dangling**command:

```
Router# show memory debug references dangling
Address  Reference Free_block Cont_block Cont_block_name
442D5774 458CE5EC  458CE5BC   44284960   bss
442D578C 46602998  46602958   44284960   bss
442D58A0 465F9BC4  465F9B94   44284960   bss
442D58B8 4656785C  4656781C   44284960   bss
442D5954 45901E7C  45901E4C   44284960   bss
.
```

.
.
The table below describes the significant fields shown in the displays.

Table 21: show memory debug references Field Descriptions

Field	Description
Address	Hexadecimal address of the block having the given or dangling reference.
Reference	Address which is given or dangling.
Free_block	Address of the free block which now contains the memory referenced by the dangling reference.
Cont_block	Address of the control block which contains the block having the reference.
Cont_block_name	Name of the control block.

show memory debug unused

To display debug information on leaks that are accessible, but are no longer needed, use the **show memory debug unused**command in user EXEC or privileged EXEC mode.

show memory debug unused

Syntax Description

This command has no arguments or keywords.

Command Modes

User EXEC Privileged EXEC

Command History

Release	Modification
12.0	This command was introduced.

Examples

The following is sample output from the **show memory debug unused**command:

```
Router# show memory debug unused
Address  Alloc_pc PID  size     Name
654894B8 62BF31DC -2   44       *Init*
6549A074 601F7A84 -2   4464     XDI data
6549B218 601F7274 -2   4500     XDI data
6549DFB0 6089DDA4 42   84       Init
65509160 6089DDA4 1    84       *Init*
6550A260 6089DDA4 2    84       *Init*
6551FDB4 6089DDA4 4    84       *Init*
6551FF34 627EFA2C -2   24       *Init*
65520B3C 6078B1A4 -2   24       Parser Mode Q1
65520B88 6078B1C8 -2   24       Parser Mode Q2
65520C40 6078B1A4 -2   24       Parser Mode Q1
65520C8C 6078B1C8 -2   24       Parser Mode Q2
65520D44 6078B1A4 -2   24       Parser Mode Q1
65520D90 6078B1C8 -2   24       Parser Mode Q2
65520E48 6078B1A4 -2   24       Parser Mode Q1
65520E94 6078B1C8 -2   24       Parser Mode Q2
65520F4C 6078B1A4 -2   24       Parser Mode Q1
65520F98 6078B1C8 -2   24       Parser Mode Q2
65521050 6078B1A4 -2   24       Parser Mode Q1
6552109C 6078B1C8 -2   24       Parser Mode Q2
65521154 6078B1A4 -2   24       Parser Mode Q1
655211A0 6078B1C8 -2   24       Parser Mode Q2
.
.
.
```

The table below describes the significant fields shown in the display.

Table 22: show memory debug unused Field Descriptions

Field	Description
Address	Hexadecimal address of the block.

Field	Description
Alloc_pc	Address of the program counter that allocated the block.
PID	Process identifier of the process that allocated the block.
size	Size of the unused block (in bytes).
Name	Name of the process that owns the block.

show crypto debug-condition

To display crypto debug conditions that have already been enabled in the router, use the **show crypto debug-condition**command in privileged EXEC mode.

show crypto debug-condition [peer] [connid] [spi] [fvrf] [**gdoi-group** *groupname*] [**isakmp profile** *profile-name*] **[ivrf]** [**local** *ip-address*] **[unmatched]** [**username** *username*]

Syntax Description

peer	(Optional) Displays debug conditions related to the peer. Possible conditions can include peer IP address, subnet mask, hostname, username, and group key.
connid	(Optional) Displays debug conditions related to the connection ID.
spi	(Optional) Displays debug conditions related to the security parameter index (SPI).
fvrf	(Optional) Displays debug conditions related to the front-door virtual private network (VPN) routing and forwarding (FVRF) instance.
gdoi-group *groupname*	(Optional) Displays debug conditions related to the Group Domain of Interpretation (GDOI) group filter. • The *groupname*value is the name of the GDOI group.
isakmp profile *profile-name*	(Optional) Displays debug conditions related to the Internet Security Association Key Management Protocol (ISAKMP) profile filter. • The *profile-name*value is the name of the profile filter.
ivrf	(Optional) Displays debug conditions related to the inside VRF (IVRF) instance.
local *ip-address*	(Optional) Displays debug conditions related to the local address debug condition filters. • The *ip-address* is the IP address of the local crypto endpoint.

unmatched	(Optional) Displays debug messages related to the Internet Key Exchange (IKE), IP Security (IPsec), or the crypto engine, depending on what was specified via the **debug crypto condition unmatched** [**engine** | **gdoi-group**| **ipsec** | **isakmp**] command.
username *username*	(Optional) Displays debug messages related to the AAA Authentication (Xauth) or public key infrastructure (PKI) and authentication, authorization, and accounting (AAA) username filter.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)SXD	This command was integrated into Cisco IOS Release 12.2(18)SXD.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.4(11)T	The **gdoi-group** *groupname* **, isakmp profile** *profile-name* **, local** *ip-address* **,**and **username** *username* keywords and arguments were added.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

You can specify as many filter values as specified via the **debug crypto condition** command. (You cannot specify a filter value that you did not use in the **debug crypto condition** command.)

Examples

The following example shows how to display debug messages when the peer IP address is 10.1.1.1, 10.1.1.2, or 10.1.1.3 and when the connection ID 2000 of crypto engine 0 is used. This example also shows how to enable global debug crypto CLIs and enable the **show crypto debug-condition** command to verify conditional settings.

```
Router#
debug crypto condition connid 2000 engine-id 1
Router#
debug crypto condition peer ipv4 10.1.1.1
Router#
debug crypto condition peer ipv4 10.1.1.2
Router#
debug crypto condition peer ipv4 10.1.1.3
Router#
debug crypto condition unmatched
! Verify crypto conditional settings.
```

```
Router#
show crypto debug-condition
Crypto conditional debug currently is turned ON
IKE debug context unmatched flag:ON
IPsec debug context unmatched flag:ON
Crypto Engine debug context unmatched flag:ON
IKE peer IP address filters:
10.1.1.1  10.1.1.2   10.1.1.3
Connection-id filters:[connid:engine_id]2000:1,
! Enable global crypto CLIs to start conditional debugging.
Router#
debug crypto isakmp
Router#
debug crypto ipsec
Router#
debug crypto engine
```

The following example shows how to disable all crypto conditional settings via the **reset** keyword:

```
Router#
debug crypto condition reset
! Verify that all crypto conditional settings have been disabled.
Router#
show crypto debug-condition
Crypto conditional debug currently is turned OFF
IKE debug context unmatched flag:OFF
IPsec debug context unmatched flag:OFF
Crypto Engine debug context unmatched flag:OFF
```

Related Commands

Command	Description
debug crypto condition	Defines conditional debug filters.
debug crypto condition unmatched	Displays crypto conditional debug messages when context information is unavailable to check against debug conditions.

show debugging

To display information about the types of debugging that are enabled for your router, use the **show debugging** command in privileged EXEC mode.

show debugging

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
11.1	This command was introduced.
12.3(7)T	The output of this command was enhanced to show TCP Explicit Congestion Notification (ECN) configuration.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.4(20)T	The output of this command was enhanced to show the user-group debugging configuration.

Examples

The following is sample output from the **show debugging** command. In this example, the remote host is not configured or connected.

```
Router# show debugging
!
TCP:
  TCP Packet debugging is on
  TCP ECN debugging is on
!
Router# telnet 10.1.25.234
!
Trying 10.1.25.234 ...
!
00:02:48: 10.1.25.31:11001 <---> 10.1.25.234:23 out ECN-setup SYN
00:02:48: tcp0: O CLOSED 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 4128
00:02:50: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
00:02:50: cwnd from 1460 to 1460, ssthresh from 65535 to 2920
00:02:50: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 4128
00:02:54: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
```

```
00:02:54: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:02:54: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 4128
00:03:02: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
00:03:02: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:02: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 ECE CWR SYN  WIN 4128
00:03:18: 10.1.25.31:11001 <---> 10.1.25.234:23 SYN with ECN disabled
00:03:18: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
00:03:18: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:18: tcp0: O SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 4128
00:03:20: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
00:03:20: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:20: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 4128
00:03:24: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
00:03:24: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:24: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 4128
00:03:32: 10.1.25.31:11001 <---> 10.1.25.234:23 congestion window changes
00:03:32: cwnd from 1460 to 1460, ssthresh from 2920 to 2920
00:03:32: tcp0: R SYNSENT 10.1.25.234:11001 10.1.25.31:23 seq 1922220018
        OPTS 4 SYN  WIN 4128
!Connection timed out; remote host not responding
```

The following is sample output from the **show debugging** command when user-group debugging is configured:

```
Router# show debugging
!
usergroup:
 Usergroup Deletions debugging is on
 Usergroup Additions debugging is on
 Usergroup Database debugging is on
 Usergroup API debugging is on
!
```

The following is sample output from the **show debugging** command when SNAP debugging is configured:

```
Router# show debugging
Persistent variable debugging is currently All
SNAP Server Debugging ON
SNAP Client Debugging ON
Router#
```

The table below describes the significant fields in the output.

Table 23: show debugging Field Descriptions

Field	Description
OPTS 4	Bytes of TCP expressed as a number. In this case, the bytes are 4.
ECE	Echo congestion experience.
CWR	Congestion window reduced.
SYN	Synchronize connections--Request to synchronize sequence numbers, used when a TCP connection is being opened.
WIN 4128	Advertised window size, in bytes. In this case, the bytes are 4128.

Field	Description
cwnd	Congestion window (cwnd)--Indicates that the window size has changed.
ssthresh	Slow-start threshold (ssthresh)--Variable used by TCP to determine whether or not to use slow-start or congestion avoidance.
usergroup	Statically defined usergroup to which source IP addresses are associated.

show debugging condition

To display the current state of debugging conditions, use the **show debugging condition**command in privileged EXEC mode.

show debugging condition [*condition-id*| **all**| **next-call** {**gprs**| **pdp**| **summary**}]

Syntax Description

condition-id	(Optional) Number of the condition for which you want to display its current state. The range is from 1 to 1000.
all	(Optional) Displays the current state for all conditions.
next-call	(Optional) Displays existing debug next-call conditions or Packet Data Protocol (PDP) with next-call debug conditions.
gprs	(Optional) Displays the information of all the (General Packet Radio System) GPRS under the next call debug condition.
pdp	(Optional) Displays the information of all the PDPs under the next call debug condition.
summary	(Optional) Displays existing debug next call conditions.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(22)YE	This command was introduced.
12.4(24)T	This command was integrated into a release earlier than Cisco IOS Release 12.4(24)T. The **gprs**, **pdp**, and **summary** keywords are not supported in T releases.

Usage Guidelin

Note The syntax of the command depends on your platform and release. The **next-call**, **gprs**, **pdp** and **summary** keywords are not supported in Cisco IOS Release 12.4(24)T and earlier releases.

Configure the **debug condition** command to enable conditional debgging.

Examples

The following is sample output from the **show debugging condition** command. The field descriptions are self-explanatory:

```
Router# show debugging condition
Condition 1: interface Fa0/0 (1 flags triggered)
        Flags: Fa0/0
Condition 2: interface Fa0/1 (1 flags triggered)
        Flags: Fa0/1
Condition 3: interface Et3/0 (1 flags triggered)
        Flags: Et3/0
Condition 4: username user1 (0 flags triggered)
```

Related Commands

Command	Description
debug condition	Limits output for some debug commands based on specified conditions.

voice call debug

To debug a voice call, use the **voice call debug** command in global configuration mode. To disable the **short-header** setting and return tothe **full-guid** setting, use the **no** form of this command.

{**voice call debug full-guid| short-header**}

{**no voice call debug full-guid| short-header**}

Syntax Description

full-guid	Displays the GUID in a 16-byte header. **Note** When the no version of this command is input with the full-guid keyword, the short 6-byte version displays. This is the default.
short-header	Displays the CallEntry ID in the header without displaying the GUID or module-specific parameters.

Command Default

The short 6-byte header displays.

Command Modes

Global configuration

Command History

Release	Modification
12.2(11)T	The new debug header was added to the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, Cisco 3660 series, Cisco AS5300, Cisco AS5350, Cisco AS5400, Cisco AS5800, Cisco AS5850, and Cisco MC3810.
12.2(15)T	The header-only keyword was replaced by the short-header keyword.

Usage Guidelines

Despite its nontraditional syntax (trailing rather than preceding "debug"), this is a normal **debug** command.

You can control the contents of the standardized header. Display options for the header are as follows:

- Short 6-byte GUID
- Full 16-byte GUID
- Short header which contains only the CallEntry ID

The format of the GUID headers is as follows: //CallEntryID/GUID/Module-Dependent-List/Function-name:.

The format of the short header is as follows: //CallEntryID/Function-name:.

When the voice call debug short-header command is entered, the header displays with no GUID or module-specific parameters. When the no voice call debug short-header command is entered, the header, the

6-byte GUID, and module-dependent parameter output displays. The default option is displaying the 6-byte GUID trace.

Note Using the no form of this command does not turn off debugging.

Examples

The following is sample output when the full-guid keyword is specified:

```
Router# voice call debug full-guid
!
00:05:12: //1/0E2C8A90-BC00-11D5-8002-DACCFDCEF87D/VTSP:(0:D):0:0:4385/vtsp_insert_cdb:
00:05:12: //-1/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: 00:05:12:
 //1/0E2C8A90-BC00-11D5-8002-DACCFDCEF87D/VTSP:(0:D):0:0:4385/vtsp_open_voice_and_set_params:
00:05:12:
//1/0E2C8A90-BC00-11D5-8002-DACCFDCEF87D/VTSP:(0:D):0:0:4385/vtsp_modem_proto_from_cdb:
00:05:12: //1/0E2C8A90-BC00-11D5-8002-DACCFDCEF87D/VTSP:(0:D):0:0:4385/set_playout_cdb:
00:05:12:
//1/0E2C8A90-BC00-11D5-8002-DACCFDCEF87D/VTSP:(0:D):0:0:4385/vtsp_dsp_echo_canceller_control:
```

Note The "//-1/" output indicates that CallEntryID for the CCAPI module is not available.

The table below describes significant fields shown in the display.

Table 24: voice call debug full-guid Field Descriptions

Field	Description
VTSP:(0:D):0:0:4385	VTSP module, port name, channel number, DSP slot, and DSP channel number.
vtsp_insert_cdb	Function name.
CCAPI	CCAPI module.

The following is sample output when the short-header keyword is specified:

```
Router(config)# voice call debug short-header
!
00:05:12: //1/vtsp_insert_cdb:
00:05:12: //-1/cc_incr_if_call_volume:
00:05:12: //1/vtsp_open_voice_and_set_params:
00:05:12: //1/vtsp_modem_proto_from_cdb:
00:05:12: //1/set_playout_cdb:
00:05:12: //1/vtsp_dsp_echo_canceller_control:
```

Note The "//-1/" output indicates that CallEntryID for CCAPI is not available.

Related Commands

Command	Description
debug rtsp api	Displays debug output for the RTSP client API.
debug rtsp client session	Displays debug output for the RTSP client data.
debug rtsp error	Displays error message for RTSP data.
debug rtsp pmh	Displays debug messages for the PMH.
debug rtsp socket	Displays debug output for the RTSP client socket data.
debug voip ccapi error	Traces error logs in the CCAPI.
debug voip ccapi inout	Traces the execution path through the CCAPI.
debug voip ivr all	Displays all IVR messages.
debug voip ivr applib	Displays IVR API libraries being processed.
debug voip ivr callsetup	Displays IVR call setup being processed.
debug voip ivr digitcollect	Displays IVR digits collected during the call.
debug voip ivr dynamic	Displays IVR dynamic prompt play debug.
debug voip ivr error	Displays IVR errors.
debug voip ivr script	Displays IVR script debug.
debug voip ivr settlement	Displays IVR settlement activities.
debug voip ivr states	Displays IVR states.
debug voip ivr tclcommands	Displays the TCL commands used in the script.
debug voip rawmsg	Displays the raw VoIP message.
debug vtsp all	Enables **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**.
debug vtsp dsp	Displays messages from the DSP.
debug vtsp error	Displays processing errors in the VTSP.
debug vtsp event	Displays the state of the gateway and the call events.

Command	Description
debug vtsp port	Limits VTSP debug output to a specific voice port.
debug vtsp rtp	Displays the voice telephony RTP packet debugging.
debug vtsp send-nse	Triggers the VTSP software module to send a triple redundant NSE.
debug vtsp session	Traces how the router interacts with the DSP.
debug vtsp stats	Debugs periodic statistical information sent and received from the DSP
debug vtsp vofr subframe	Displays the first 10 bytes of selected VoFR subframes for the interface.
debug vtsp tone	Displays the types of tones generated by the VoIP gateway.