Appendix A: Cable Debug Commands

Revised: August 12, 2013, OL-15510-17

Additional **debug** commands are documented in the *Cisco IOS Debug Command Reference*, available on Cisco.com and the Documentation CD-ROM.

Caution The **debug** commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. You should use caution when enabling debug messages because sending these messages to the console consumes system resources. Turning on too many types of debug messages can adversely affect the router's network performance, depending on what messages are being displayed and the type of traffic that is occurring.

New Commands

Command	Cisco IOS Software Release
debug cable ipv6	12.2(33)SCA
debug cable service-ds-selection	12.3(23)BC
debug cr10k-rp dbs-queue	12.3(23)BC1
debug ehsa	12.2(33)SCA
debug cable wbcmts resiliency	12.2(33)SCB
debug cable cm-ctrl	12.2(33)SCC
debug cable cm-status	12.2(33)SCC
debug cable mdd	12.2(33)SCC
debug cable md-sg	12.2(33)SCC
debug cable ubg	12.2(33)SCC
debug cable wbcmts admission-control	12.2(33)SCC
debug pxf atom	12.2(33)SCC
debug cable dbs	12.2(33)SCD
debug cable multicast counter clear	12.2(33)SCE
debug cable multicast counter start	12.2(33)SCE
debug cable multicast counter stop	12.2(33)SCE
debug cable multicast forwarding	12.2(33)SCE

Command	Cisco IOS Software Release
debug cable multicast latency	12.2(33)SCE
debg cable acfe filter	12.2(33)SCF
debug cable acfe	12.2(33)SCF
debug cable dynamic-qos subscriber	12.2(33)SCF
debug cable dynamic-qos trace	12.2(33)SCF
debug cmts ipc-cable base	12.2(33)SCF
debug cmts ipc-cable client	12.2(33)SCF
debug hccp rfswitch	12.2(33)SCG

Modified Commands

Command	Cisco IOS Software Release
debug cable wbcmts	12.2(33)SCB
debug cr10k-rp dbs-queue	12.2(33)SCB
debug hw-module all upgrade	12.2(33)SCB
debug hw-module bay	12.2(33)SCB
debug hw-module subslot	12.2(33)SCB
debug cable interface	12.2(33)SCC
debug cable mac-scheduler	12.2(33)SCC
debug cable tlvs	12.2(33)SCC
debug cable ipv6	12.2(33)SCF1
debug cable dsg	12.2(33)SCG

debug c10k-jacket

To enable debugging information for the Wideband SIP, use the **debug c10k-jacket** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug c10k-jacket [**events** | **plugin** | **spa-audits** | **spa-events** | **spa-vft**]

no debug c10k-jacket

Syntax Description

events	Displays event information for the Wideband SIP.
plugin	Displays plugging processing information for the Wideband SIP.
spa-audits	Displays audit information for the Wideband SIP.
spa-events	Displays event information for the Wideband SIP.
spa-vft	Displays vft information for the Wideband SIP.

Command Default

No Wideband SIP debug messages are enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(21)BC	This command was introduced for the Cisco uBR10012 router.

Usage Guidelines

The **debug c10k-jacket** command is intended for use by Cisco technical support personnel.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use **debug** commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco Systems technical support personnel. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased debug command processing overhead will affect system use.

Examples

The following example shows how to enable the **debug c10k-jacket spa-vft** debug messages for the Wideband SIP.

```
Router# debug c10k-jacket spa-vft
c10k jacket SPA VFT calls debugging is on
Router#
Sep  6 17:10:47.410: cr10k_wbcmts_rcv_spa_event: wbcmts spa event recv of type 3
Sep  6 17:10:47.410: NB chan stats: 1 entries.
Sep  6 17:10:47.414: cr10k_wbcmts_rcv_spa_event: wbcmts spa event recv of type 2
Sep  6 17:10:47.414: WB chan stats: 1 entries.
```

Related Commands

Command	Description
debug cable fn	Enables debugging information for cable fiber nodes.
debug cable wbcmts	Enables debugging information for the wideband CMTS.
debug hw-module bay	Enables debugging information for a Wideband SPA.

debug cable

To enable debugging of the cable interface, use the **debug cable** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable

no debug cable

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3 XA	This command was introduced.

Usage Guidelines

This command enables debugging of the cable interfaces. To avoid excessive output that could interfere with router performance, you should limit debugging to a particular cable interface, using the **debug cable interface** command, before giving the **debug cable** command.

Examples

The following shows sample output from the **debug cable bpiatp** command:

```
Router# debug cable
Router# debug cable interface c3/0
CMTS interface debugging is on

Router#
Jun 25 08:36:37.339: cmts_helper_forward 00e0.a3b6.f0af no match

Jun 25 08:36:37.339: cmts_dhcp_glean: type=2 sid 0 IP=221.222.151.182 dhcp mac=0
Jun 25 08:36:40.339: cmts_helper_forward 00e0.a3b6.f0af no match

Jun 25 08:39:13.419: Failed to find CM with mac address 0006.28dc.37fd
Jun 25 08:39:13.419: Failed to find CM with mac address 00d0.bad3.c0cd
Jun 25 08:39:13.419: Failed to find CM with mac address 0003.e350.9cdb

Jun 25 08:39:40.527: Lookup failed - unable to find CM with SID 0

Jun 25 08:38:53.583: Lookup failed - unable to find CM with SID 0
```

Note The last message displayed above, "unable to find CM with SID 0," is typically generated by the **show cable** commands and can be ignored.

Related Commands

Command	Description
debug cable interface	Enables debugging on a specific cable interface.

debug cable acfe

To show the debug information related to the algorithm or interaction with the system, use the **debug cable acfe** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable acfe [**algorithm** | **all** | **filter** | **hccp** | **process** | **read** | **topology** | **verbose** | **write**]

no debug cable acfe

Syntax Description

algorithm	Displays internal operations of Fairness Across DOCSIS Interfaces feature algorithms.
all	Displays debugging messages for all cable Fairness Across DOCSIS Interfaces feature events.
filter	Applies the filter to limit the debug output. See the **debug cable acfe filter** command.
hccp	Displays Fairness Across DOCSIS Interfaces feature high availability and Hot Standby Connection-to-Connection Protocol (HCCP) activities.
process	Displays Fairness Across DOCSIS Interfaces feature process activities.
read	Displays input from system.
topology	Displays cluster building information.
verbose	Displays all Fairness Across DOCSIS Interfaces feature internal data.
write	Displays output to system.

Command Default

Debug is disabled and Fairness Across DOCSIS Interfaces feature messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCF	This command was introduced.

Usage Guidelines

You should run the **debug cable acfe** command first to enable other debug options available in this command. Avoid enabling **all** and **verbose** options because they put a strain on the system resources.

Examples

The following examples shows how to enable debug messages for Fairness Across DOCSIS Interfaces feature.

The following is sample output from the **debug cable acfe** command with the **algorithm** keyword:

```
Router# debug cable acfe algorithm

ACFE Algorithms debugging is on
!
```

```
!
!
```

The following is sample output from the **debug cable acfe** command with the **all** keyword:

```
Router# debug cable acfe all

Jan 20 10:18:52.266: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:18:52.266: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:18:52.266: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:18:52.266: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:18:52.266: ACFE: next interval 5000 ms
Jan 20 10:18:57.266: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:18:57.266: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:18:57.266: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:18:57.266: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:18:57.266: ACFE: next interval 5000 ms
Jan 20 10:19:02.271: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:19:02.271: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:19:02.271: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:19:02.271: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:19:02.271: ACFE: next interval 5000 ms
Jan 20 10:19:07.271: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:19:07.271: ACFE: 1 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:19:07.271: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:19:07.271: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:19:07.271: ACFE: next interval 5000 ms
Jan 20 10:19:12.271: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:19:12.271: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:19:12.271: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:19:12.271: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:19:12.271: ACFE: next interval 5000 ms
!
!
!
```

The following is sample output from the **debug cable acfe** command with the **hccp** keyword:

```
Router# debug cable acfe hccp

Jan 20 10:22:02.309: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:22:07.310: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:22:12.310: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:22:17.310: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
Jan 20 10:22:22.310: ACFE: sch_rp_sync_acfe_guar_grp LBLT quanta sync sent for
Modular-Cable 1/0/0
!
!
!
```

The following is sample output from the **debug cable acfe** command with the **process** keyword:

```
Router# debug cable acfe process

Jan 20 10:23:57.323: ACFE: Modular-Cable 1/0/0 is to be processed
```

```
Jan 20 10:23:57.323: ACFE: next interval 5000 ms
Jan 20 10:24:02.335: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:24:02.335: ACFE: next interval 5000 ms
Jan 20 10:24:07.335: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:24:07.335: ACFE: next interval 5000 ms
Jan 20 10:24:12.336: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:24:12.336: ACFE: next interval 5000 ms
Jan 20 10:24:17.336: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:24:17.336: ACFE: next interval 5000 ms
Jan 20 10:24:22.336: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:24:22.336: ACFE: next interval 5000 ms
Jan 20 10:24:27.336: ACFE: Modular-Cable 1/0/0 is to be processed
Jan 20 10:24:27.336: ACFE: next interval 5000 ms
!
!
!
```

The following is sample output from the **debug cable acfe** command with the **read** keyword:

```
Router# debug cable acfe read

Jan 20 10:25:27.349: ACFE: 1 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:25:27.349: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:25:32.349: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:25:32.349: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:25:37.349: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:25:37.349: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:25:42.349: ACFE: 0 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:25:42.349: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
Jan 20 10:25:47.349: ACFE: 1 flows on BG 1 on Modular-Cable1/0/0:0
Jan 20 10:25:47.349: ACFE: 9000 kbps CIR on BG 1 on Modular-Cable1/0/0:0 above reserved BW
!
!
!
```

The following is sample output from the **debug cable acfe** command with the **topology** keyword:

```
Router# debug cable acfe topology

ACFE Cluster Buidling debugging is on
!
!
!
```

The following is sample output from the **debug cable acfe** command with the **verbose** keyword:

```
Router# debug cable acfe verbose

Jan 20 10:30:02.413: ACFE: rf_chan_num 3 rf_idx 3 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 0 rf_idx 0 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 1 rf_idx 1 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 2 rf_idx 2 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 3 rf_idx 3 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: BG 1 with ratio 71612497920000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 0 with ratio 71607255040000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 2 with ratio 71602012160000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 2 with ratio 71602012160000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 0 with ratio 71607255040000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 1 with ratio 71612497920000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 1 with ratio 83844136960000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 0 with ratio 83838894080000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 2 with ratio 83838894080000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 0 with ratio 83838894080000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 2 with ratio 83838894080000  found as pivot
Jan 20 10:30:07.414: ACFE: BG 1 with ratio 83844136960000  found as pivot
```

```
Jan 20 10:30:07.414: ACFE: rf_chan_num 0 rf_idx 0 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 1 rf_idx 1 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 2 rf_idx 2 on Modular-Cable 1/0/0
Jan 20 10:30:07.414: ACFE: rf_chan_num 3 rf_idx 3 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 0 rf_idx 0 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 1 rf_idx 1 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 2 rf_idx 2 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 3 rf_idx 3 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: BG 1 with ratio 71612497920000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 0 with ratio 71607255040000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 2 with ratio 71602012160000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 2 with ratio 71602012160000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 0 with ratio 71607255040000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 1 with ratio 71612497920000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 1 with ratio 83844136960000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 0 with ratio 83838894080000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 2 with ratio 83838894080000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 0 with ratio 83838894080000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 2 with ratio 83838894080000  found as pivot
Jan 20 10:30:12.414: ACFE: BG 1 with ratio 83844136960000  found as pivot
Jan 20 10:30:12.414: ACFE: rf_chan_num 0 rf_idx 0 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 1 rf_idx 1 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 2 rf_idx 2 on Modular-Cable 1/0/0
Jan 20 10:30:12.414: ACFE: rf_chan_num 3 rf_idx 3 on Modular-Cable 1/0/0
!
!
!
```

The following is sample output from the **debug cable acfe** command with the **write** keyword:

```
Router# debug cable acfe write

ACFE Output debugging is on
!
!
!
```

Related Commands

Command	Description
debug cable acfe filter	Applies filters to the Fairness Across DOCSIS Interfaces feature debug information.

debug cable acfe filter

To apply filters to the Fairness Across DOCSIS Interfaces feature debug information and limit the output to a specific controller, cluster, or interface, use the **debug cable acfe** command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

debug cable acfe filter {**controller** {**modular-cable** *slot/subslot/controller-unit* **cluster** *cluster-index*} | **interface** {**integrated-cable | modular-cable | wideband-cable**} *slot/subslot/port:interface-num*}

no debug cable acfe

Syntax Description

filter	Applies a filter to limit the debug output.
controller	Displays the specific controller information. • **modular-cable**—Specifies the controller interface. • *slot*—Controller slot number. The valid range is from 0 to 8. • *subslot*—Controller subslot number. The valid range is from 0 to 3. • *controller-unit*—Controller unit number. The valid value is 0. • **cluster**—Specifies the specific cluster. • *cluster-index*—Cluster index number.
interface	Identifies the specific interface. • **integrated-cable**—Specifies the integrated cable interface. This option is available only for the Cisco UBR-MC20X20V line card. • **modular-cable**—Specifies the modular cable interface. • **wideband-cable**—Specifies the wideband cable interface. • *slot*—Slot number of the cable interface. • *subslot*—Subslot number of the cable interface. • *port*—Port number. • *interface-num*—Interface number. The valid values for the above arguments depend on the CMTS router and cable interface line card. See the hardware documentation for your router chassis and cable interface line card for the supported values.

Command Default

Debug is disabled and Fairness Across DOCSIS Interfaces feature messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCF	This command was introduced.

Usage Guidelines

You should run the **debug cable acfe** command first to enable the debug option. Avoid enabling **all** and **verbose** options because they put a strain on the system resources.

Examples

The following examples show how to enable debug messages for Fairness Across DOCSIS Interfaces feature.

The following is sample output from the **debug cable acfe filter** command with the **controller** keyword:

```
Router# debug cable acfe filter controller modular-Cable 1/0/0 cluster 0
```

The following is sample output from the **debug cable acfe filter** command with the **interface** keyword:

```
Router# debug cable acfe filter interface modular-Cable 1/0/0:0
```

Related Commands

Command	Description
debug cable acfe	Enables debug operation for Fairness Across DOCSIS Interfaces feature.

debug cable admission-control

To enable automatic Admission Control troubleshooting processes on the Cisco CMTS, use the **debug cable admission-control** command in privileged EXEC mode. To disable debugging mode, use the **no** form of this command.

debug cable admission-control {cpu | memory | us-bandwidth | ds-bandwidth}

no debug cable admission-control

Syntax Description

cpu	Keyword displays CPU debugging information and processes.
memory	Keyword displays physical memory debugging information and processes.
us-bandwidth	Keyword displays upstream debugging information and processes.
ds-bandwidth	Keyword displays downstream debugging information and processes.

Command Default

Admission Control and Service Flow Admission Control debugging is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(13a)BC	This command was introduced on the Cisco uBR10012 and the Cisco uBR7246VXR router.
12.3(21)BC	This command continues on the Cisco uBR10012 router and the Cisco uBR7246VXR router to support the Service Flow Admission Control feature.

Usage Guidelines

Admission Control debugging processes have some impact to resources on the Cisco CMTS. Any one of the command options can be used for selective debugging and to minimize impact.

For additional Admission Control feature information, refer to the following document on Cisco.com:

- *Admission Control for the Cisco Cable Modem Termination System*

For additional information for Service Flow Admission Control, commencing in Cisco IOS Release 12.3(21)BC, refer to the following document on Cisco.com:

- *Service Flow Admission Control for the Cisco Cable Modem Termination System*

Examples

The following example illustrates CPU debugging with Admission Control:

```
Router# debug cable admission control cpu
*Sep 12 23:08:53.255: CPU admission control check succeeded
*Sep 12 23:08:53.255: System admission control check succeeded
*Sep 12 23:08:53.255: CPU admission control check succeeded
*Sep 12 23:08:53.255: System admission control check succeeded
```

The following example illustrates memory debugging with Admission Control:

```
Router# debug cable admission control memory
*Sep 12 23:08:53.255: CPU admission control check succeeded
*Sep 12 23:08:53.255: System admission control check succeeded
*Sep 12 23:08:53.255: CPU admission control check succeeded
*Sep 12 23:08:53.255: System admission control check succeeded
```

The following example illustrates event debugging with Admission Control:

```
Router# debug cable admission control event
*Sep 12 23:15:22.867: Entering admission control check on PRE and it's a cm-registration
*Sep 12 23:15:22.867: Admission control event check is TRUE
```

The following example illustrates upstream throughput debugging with Admission Control:

```
Router# debug cable admission control us-bandwidth
Oct  8 23:29:11: Failed to allocate US bandwidth for CM 0007.0e01.9b45 in adding a new
service entry
```

The following example illustrates downstream throughput debugging with Admission Control:

```
Router# debug cable admission control ds-bandwidth
Oct  8 23:29:11: Failed to allocate DS bandwidth for CM 0007.0e01.1db5 in adding a new
service entry
```

Related Commands

Command	Description
cable admission-control	Configures the CPU and memory thresholds for the Cisco CMTS and supporting broadband processing engines (BPEs)
cable admission-control event	Configures and enables Admission Control event types on the Cisco CMTS.
cable admission-control ds-bandwidth	Configures Admission Control downstream bandwidth thresholds on the Cisco CMTS.
cable admission-control us-bandwidth	Configures Admission Control upstream bandwidth thresholds on the Cisco CMTS.
clear cable admission control counters	Clears all Admission Control resource counters on the Cisco CMTS.
show cable admission-control	Displays status information for running configuration, traffic metrics, and Admission Control events on the Cisco CMTS.

debug cable admission-control flow-categorization

To display service flow categorization results, enabled when a service flow is classified, use the **debug cable admission-control flow categorization** command in Privileged EXEC mode. This command displays the application by which it was categorized, along with which rule is matched. Use the **no** form of this command to disable this debugging.

debug cable admission-control flow-categorization

no debug cable admission-control flow-categorization

Command Default

Debugging for Service Flow Admission Control is disabled by default on the Cisco CMTS, through Cisco IOS Release 12.3(21a) enables this feature by default, with associated default functions.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(21)BC	This command was introduced for the Cisco uBR10012 router and the Cisco uBR7246VXR router.

Usage Guidelines

For additional information for Service Flow Admission Control, commencing in Cisco IOS Release 12.3(21)BC, refer to the following document on Cisco.com:

- *Service Flow Admission Control for the Cisco Cable Modem Termination System*

Examples

Below is a shortened example of the information displayed when the **debug cable admission-control flow-categorization** command is enabled on the Cisco CMTS. This command displays interface-level information.

```
Router# debug cable admission-control flow-categorization

int ca 5/1/1 sfid 55 identified as video pcmm priority 6 matched.
```

Related Commands

Command	Description
cable admission-control ds-bandwidth	Sets minor, major and exclusive thresholds for downstream voice or data bandwidth for each or all interfaces on the Cisco CMTS
cable admission-control preempt priority-voice	Changes the default PacketCable Emergency 911 call preemption functions on the Cisco CMTS, supporting throughput and bandwidth requirements for Emergency 911 calls above all other buckets on the Cisco CMTS.
cable admission-control us-bandwidth	Configures global or interface-level upstream bandwidth thresholds and exclusive or non-exclusive resources on the Cisco CMTS.
cable application-type include	Associates an application type with a specific and prioritized bucket on the Cisco CMTS.
cable application-type name	Assigns an alpha-numeric name for the specified bucket.

Command	Description
cable admission-control us-bandwidth	Configures per-upstream bandwidth thresholds and exclusive or non-exclusive resources on the Cisco CMTS.
debug cable admission-control flow-categorization	Displays service flow categorization results, enabled when a service flow is classified.
show application-buckets	Displays rules for any or all buckets supporting Service Flow Admission Control on the Cisco CMTS.
show interface cable admission-control reservation	Displays service flows, categorizations, and bandwidth consumption on the Cisco CMTS, for the specified interface, and the specified service flow direction.

debug cable arp

To enable debugging of the Address Resolution Protocol (ARP) when it is used on the cable interface, use the **debug cable arp** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable arp

no debug cable arp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)BC3	This command was introduced.

Usage Guidelines

The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. In particular, avoid using the **debug cable arp** command on an interface with a significant number of CMs, because the resulting volume of debug output could impact system performance. Cisco recommends that when you use the **debug cable arp** command, you limit its output to a particular interface or CM, using the **debug cable interface** or **debug cable mac-address** commands.

Examples

The following shows typical output for a particular cable modem using the **debug cable arp** command:

```
Router# debug cable arp
CMTS arp debugging is on

Router# debug cable mac-address 0020.4072.7418

ARPGLEAN cmts glean idb Cable3/0 MAC 0020.4072.7418 SID 2 ipaddr 31.0.0.2
```

Related Commands

Command	Description
debug cable dhcp	Enables debugging of the Dynamic Host Configuration Protocol on the cable interface.
debug cable encap	Enables debugging of encapsulated packets that are transmitted over the cable interface.
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for a particular CM.

debug cable arp filter

To display debugging messages about the filtering of Address Resolution Protocol (ARP) broadcasts, use the **debug cable arp filter** command in privileged EXEC mode. To the debugging messages, use the **no** form of this command.

debug cable arp filter

no debug cable arp filter

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(15)BC2	This command was introduced for the Cisco uBR7246VXR and Cisco uBR10012 universal broadband routers.

Usage Guidelines

If you suspect a particular CM is generating a large volume of ARP traffic, you can enable debugging for that particular CM using the **debug cable arp** command. If the ARP traffic is excessive, you can enable ARP filtering on the associated cable interface using the **cable arp filter** command. To show the results of that ARP filtering, use the **debug cable arp filter** command.

Tip Because this command can produce a large volume of debug information, it does not produce any output until you first limit debugging output to a particular Service ID (SID) or one or more particular CM MAC addresses, using the **debug cable interface sid** or **debug cable mac-address** commands, respectively.

Examples

The following example shows how to enable debugging messages for ARP filtering for a particular MAC address, and samples of the typical messages that can be displayed:

```
Router# debug cable mac-address 000C.0102.0304
Router# debug cable arp filter

CMTS arp filter debugging is ON

Router#

ARP Req Filter = T shdw 000C.0102.0304 sip 10.11.13.1 dhdw 00C0.0809.0A0B dip
192.168.100.14 cnt 2
ARP Req Filter = T src_ip 10.11.13.1 dst_ip 192.168.100.14
```

The following example shows how to enable debugging messages for ARP filtering for a particular SID on a cable interface:

```
Router# debug cable interface cable c5/0 sid 31
Router# debug cable arp filter

CMTS arp filter debugging is ON

Router#
```

Related Commands

Command	Description
cable arp	Activates cable Address Resolution Protocol (ARP).
cable arp filter	Controls the number of ARP packets that are allowable for each Service ID (SID) on a cable interface.
cable proxy-arp	Activates cable proxy ARP on the cable interface.
cable arp	Clears the ARP table on the router.
clear counters	Clears the packet counters on all interfaces or on a specific interface.
debug cable arp	Enables debugging of ARP traffic on a cable interface.
debug cable interface	Enables debugging for a particular Service ID (SID) on a specific cable interface.
debug cable mac-address	Enables debugging for a particular CM.
show cable arp-filter	Displays the total number of ARP replies and requests that have been sent and received, including the number of requests that have been filtered.

debug cable bpiatp

To enable debugging of the Baseline Privacy Interface (BPI) handler, use the **debug cable bpiatp** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable bpiatp

no debug cable bpiatp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3 XA	This command was introduced.

Usage Guidelines

This command activates debugging of the BPI feature. When this command is activated, the Cisco CMTS displays debugging information for any BPI-related messages that the CMTS sends or receives.

Note This command is supported only on images that support BPI or BPI+ encryption.

Tip Debugging must be enabled for one or more cable interfaces, using the **debug cable interface** command, before the **debug cable bpiatp** command displays any output.

Examples

The following shows sample output from the **debug cable bpiatp** command:

```
Router# debug cable interface c3/0
Router# debug cable bpiatp

CMTS bpi_atp debugging is on
Router#
 SID : 1         Latest : 2      Current : 1
  Status[0] : 1  DES Key[0] : DBF08460BF8073B    DES IV[0]  : D8F26C81A2F01BC
  Key Life[0]: 392 sec
  Status[1] : 1  DES Key[1] : BEC1CAE02022349    DES IV[1]  : 198C1AB9255113DC
  Key Life[1]: 87 sec
  Req : 1        Rply : 1        Rej : 0 Inv : 0
  Upstream Pri SID : 1
   SID : 1  Even Key : D7C2230BF019D02   Even IV : D8F26C81A2F01BC
   SID : 1  Odd  Key : BD875702048A401   Odd IV : 198C1AB9255113DC

 SID : 2002      Latest : 2      Current : 1
  Status[0] : 1  DES Key[0] : 237C029E0879190B   DES IV[0]  : 166B0B6207580457
  Key Life[0]: 219 sec
  Status[1] : 1  DES Key[1] : B8020021FDF1AB5    DES IV[1]  : 7A10B6E235E196E
```

```
 Key Life[1]: 69 sec
CM List:
  0050.f112.3414
IP List:
  224.0.1.3
 Req : 0        Rply : 0        Rej : 0 Inv : 0
 Upstream QoS SID : 0
  SID : 0  Even Key : FFFF0000FFFF0000  Even IV : FFFF0000FFFF0000
  SID : 0  Odd  Key : FFFF0000FFFF0000  Odd IV : FFFF0000FFFF0000
```

Related Commands

Command	Description
debug cable interface	Enables debugging on a specific cable interface.
debug cable keyman	Displays debugging information about BPI key management.
debug cable privacy	Displays debugging information whenever the BPI state changes or a BPI event occurs.

debug cable bundle

To enable debugging of the bundling of cable interfaces, use the **debug cable bundle** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable bundle [**pkt**]

no debug cable bundle [**pkt**]

Syntax Description

pkt	(Optional) Displays detailed information about packets that are transmitted over the cable bundle.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR, 12.1(0)6SC, 12.1(2)EC1	This command was introduced.
12.1(3a)EC	Subinterface support was added.

Usage Guidelines

This command activates debugging for cable interfaces that are bundled together into a master/slave relationship, using the **cable bundle** command.

Note The **debug cable bundle pkt** command does not display information about multicast packets on the Cisco uBR10012 router. Use the **show ip mroute count** and **show hardware pxf cpu mroute** commands to display this information on the Cisco uBR10012 router.

Tip This command can generate a significant amount of output. To limit the debug output to a specific cable interface, use the **debug cable interface** command, before using the **debug cable bundle** command.

Examples

The following shows typical output from the **debug cable bundle** command for a particular cable interface:

```
Router# debug cable interface c3/0
Router# debug cable bundle

00:53:23: Sending multicast packet to bundle int Cable3/0
00:53:23: Unicast packet sent out bundle int Cable3/0 to 0003.e3fa.5e21
00:53:24: Sending multicast packet to bundle int Cable3/0
00:53:25: Sending multicast packet to bundle int Cable3/0
00:53:26: Sending multicast packet to bundle int Cable3/0

Router#
```

Related Commands

Command	Description
debug cable interface	Enables debugging on a specific cable interface.

debug cable cm-status

To enable debugging information for cable modem (CM) status messages on the Cisco CMTS routers, use the **debug cable cm-status** command in privileged EXEC mode. To stop the display of debug messages, use the **no** form of this command.

debug cable cm-status

no debug cable cm-status

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced in Cisco IOS Release 12.2(33)SCC.

Examples

The following example shows how to enable debugging output using the **debug cable cm-status** command:

```
Router# debug cable cm-status

CMTS CM-STATUS message debugging is on

Apr 27 07:01:21.659: cmts_send_cm_status_test_msg: SEND IPC to Cable5/0/1 type 139 if_num
1 len 272
Apr 27 07:01:21.631: Cable5/0/1: CM 001e.6bfa.f5bc CM-STATUS msg
Apr 27 07:01:21.631: CM-STATUS:
Apr 27 07:01:21.631:     0x0000: 01 03 05 01 01
Apr 27 07:01:21.631: Send T4 Timeout to USR event handling
```

Related Commands

Command	Description
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-address	Enables debugging output for the cable modems that match the specified hardware MAC address or range of addresses.
debug cable cm-ctrl	Enables debugging output for the CM-CTRL messages on the Cisco CMTS routers.

debug cable classifiers

To display debugging messages for DOCSIS packet classifiers, use the **debug cable classifiers** command in privileged EXEC mode. To stop the display of debugging messages, use the **no** form of this command.

debug cable classifiers

no debug cable classifiers

Syntax Description

No additional keywords or syntax components are required.

Command Modes

Privileged EXEC mode

Command Default

DOCSIS packet classifier debugging is disabled by default.

Usage Guidelines

The **debug cable classifiers** command provides detailed information about the allocation, removal, activation and deactivation of packet classifiers. Generally, classifiers are used to identify IP packets by source port, destination port, or type of service. Classifiers are associated with service flows. For example, packet classifiers are dynamically created in most VOIP deployments and this debug command can be used to troubleshoot issues related to these classifiers as VOIP calls are created and torn down.

Because this command can produce a large volume of debug information, use this command only when you have also enabled debugging for a particular MAC address, set of MAC addresses, or a MAC address mask, using the **debug cable mac-address** command.

Examples

The following example enables classifier debugging for a single MAC address:

```
Router# debug cable mac-address 000a.73fa.dbaa
Router# debug cable classifiers
CMTS Packet Classifiers debugging is on
```

The following enables classifier debugging for all MAC addresses with Organizational Unique Identifier (OUI) OUI 0013.11:

```
Router# debug cable mac-addr 0013.1100.0000 ffff.ff00.0000
Routerv# debug cable classifiers
CMTS Packet Classifiers debugging is on
```

The following example illustrates sample output of the **debug cable classifiers** command for the given MAC addresses:

```
Feb  7 18:43:50.181: CFR cmts_deactivate_us_srv_flow_act_cfrs 000a.73fa.dbaa sid 1 sfid 3 st 2 dir 0 prov 1 adm 1 act
1
Feb  7 18:43:50.181: CFR cmts_remove_cm_srv_flow_cfrs 000a.73fa.dbaa sid 1 sfid 3 st 2 dir 0 prov 1 adm 0 act 0
Feb  7 18:43:50.181: CFR cmts_deactivate_ds_srv_flow_act_cfrs 000a.73fa.dbaa sid 0 sfid 4 st 2 dir 1 prov 2 adm 2 act
2
Feb  7 18:43:50.181: CFR cmts_remove_cm_srv_flow_cfrs 000a.73fa.dbaa sid 0 sfid 4 st 2 dir 1 prov 2 adm 0 act 0
Feb  7 18:43:50.181: CFR cmts_deactivate_us_srv_flow_act_cfrs 000a.73fa.dbaa sid 1 sfid 3 st 2 dir 0 prov 3 adm 0 act
0
Feb  7 18:43:50.181: CFR cmts_deactivate_us_srv_flow_act_cfrs 000a.73fa.dbaa sid 1 sfid 3 st 1 dir 0 prov 3 adm 3 act
0
Feb  7 18:43:50.181: CFR cmts_activate_us_srv_flow_act_cfrs 000a.73fa.dbaa sid 1 sfid 3 st 2 dir 0 prov 3 adm 3 act 3
```

```
Feb  7 18:43:50.181: CFR cmts_deactivate_ds_srv_flow_act_cfrs 000a.73fa.dbaa sid 0 sfid 4 st 2 dir 1 prov 4 adm 0 act
0
Feb  7 18:43:50.181: CFR cmts_deactivate_ds_srv_flow_act_cfrs 000a.73fa.dbaa sid 0 sfid 4 st 1 dir 1 prov 4 adm 4 act
0
Feb  7 18:43:50.181: CFR cmts_activate_ds_srv_flow_act_cfrs 000a.73fa.dbaa sid 0 sfid 4 st 2 dir 1 prov 4 adm 4 act 4
Feb  7 18:43:50.181: CFR cmts_set_cfr_params 000a.73fa.dbaa cfrid 1 pri 0 ord 0 dir 0 st 2 phsi 0
Feb  7 18:43:50.181: CFR cmts_activate_cfr 000a.73fa.dbaa cfrid 1 pri 1 ord 0 dir 0 st 2
Feb  7 18:43:50.181: CFR cmts_add_pkt_cfr 000a.73fa.dbaa cfrid 1 pri 1 ord 0 dir 0 st 1 phsi 0
Feb  7 18:43:50.181: CFR cmts_handle_cfr_parsed_data CFR_ADD 000a.73fa.dbaa sfid 0 action 0 dir 0 type 0 cfrid 0 pri 1
ord 0 dir 0 st 1 phsi 0
Feb  7 18:43:50.181: CFR cmts_set_cfr_params 000a.73fa.dbaa cfrid 2 pri 0 ord 0 dir 1 st 2 phsi 0
Feb  7 18:43:50.181: CFR cmts_activate_cfr 000a.73fa.dbaa cfrid 2 pri 1 ord 0 dir 1 st 2
Feb  7 18:43:50.181: CFR cmts_add_pkt_cfr 000a.73fa.dbaa cfrid 2 pri 1 ord 1 dir 1 st 1 phsi 0
Feb  7 18:43:50.181: CFR cmts_handle_cfr_parsed_data CFR_ADD 000a.73fa.dbaa sfid 0 action 0 dir 1 typ
```

Related Commands

Command	Description
debug cable dynsrv	Displays information about DOCSIS 1.1 dynamic service flow messages.
debug cable qos	Activates quality-of-service (QoS) debugging.

debug cable cm-ctrl

To enable debugging information for CM-CTRL messages on the Cisco CMTS routers, use the **debug cable cm-ctrl** command in privileged EXEC mode. To stop the display of debug messages, use the **no** form of this command.

debug cable cm-ctrl

no debug cable cm-ctrl

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced in Cisco IOS Release 12.2(33)SCC.

Examples

The following example shows how to enable debugging information for CM-CTRL messages using the **debug cable cm-ctrl** command:

```
Router# debug cable cm-ctrl

CMTS CM-CTRL message debugging is on

Apr 27 06:53:54.695: CM-CTRL-REQ IPC Msg:
*Apr 27 06:53:54.695:     0x0000: 01 06 00 1E 6B FA F5 BC 02 02 00 00 03 04 00 00
*Apr 27 06:53:54.695:     0x0010: 13 88 04 02 00 00 05 01 00 06 01 01 07 01 03 08
*Apr 27 06:53:54.695:     0x0020: 01 02 0B 01 01
Apr 27 06:53:54.647: cmts_clc_proc_cm_ctrl_req called
Apr 27 06:53:54.647: cmts_clc_proc_cm_ctrl_req: calling cmts_cm_ctrl_handle_ipc bdy_len 37
Apr 27 06:53:54.647: CM-CTRL-IPC Recv:
Apr 27 06:53:54.647:     0x0000: 01 06 00 1E 6B FA
F5 BC 02 02 00 00 03 04 00 00
Apr 27 06:53:54.647:     0x0010: 13 88 04 02 00 00 05 01 00 06 01 01 07 01 03 08
Apr 27 06:53:54.647:     0x0020: 01 02 0B 01 01
Apr 27 06:53:54.647: CM-CTRL IPC encode MAC: 001e.6bfa.f5bc, timeout: 5000, retry: 0,
Apr 27 06:53:54.647: CM-CTRL IPC encode tid: 0, pending: 0, num_tlvs: 1,
Apr 27 06:53:54.647: CM-CTRL-REQ CM node: Cable5/0/1, CM 001e.6bfa.f5bc is enqueued
Apr 27 06:53:54.651: CM-CTRL encode MAC: 001e.6bfa.f5bc, timeout: 5000, retry: 0,
Apr 27 06:53:54.651: CM-CTRL encode tid: 0, pending: 0, num_tlvs: 1,
Apr 27 06:53:54.651: TLV[0]: type 3, result: 2, value: 0x01000000
Apr 27 06:53:54.651: CM-CTRL: Cable5/0/1 CM 001e.6bfa.f5bc control bitmask: 0x07FE
Apr 27 06:53:54.651: CM-CTRL-REQ TLV:
Apr 27 06:53:54.651:     0x0000: 03 01 01
Apr 27 06:53:54.651: CM-CTRL-REQ Msg:
Apr 27 06:53:54.651:     0x0000: C2 00 00 1D 00 00 00 1E 6B FA F5 BC 00 19 2F E6
Apr 27 06:53:54.651:     0x0010: 06 79 00 0B 00 00 03 04 2A 00 00 00 03 01 01
Apr 27 06:53:54.651: CM-CTRL-REQ: Cable5/0/1, CM 001e.6bfa.f5bc enqueued locally
Apr 27 06:53:54.651: CM-CTRL-RSP Timer started.
Apr 27 06:53:54.651: CM-CTRL-REQ is sent to CM.
Apr 27 06:53:54.659: cmts_cm_ctrl_rsp called
Apr 27 06:53:54.659: Received CM-CTRL-RSP msg from 001e.6bfa.f5bc transaction id (0)
```

```
Apr 27 06:53:54.659: CM-CTRL-RSP:
Apr 27 06:53:54.659:     0x0000: C2 00 00 1D 9C 24 00 19 2F E6 06 79 00 1E 6B FA
Apr 27 06:53:54.659:     0x0010: F5 BC 00 0B 00 00 03 04 2B 00 00 00 03 01 00
Apr 27 06:53:54.659: CM-CTRL-RSP: Cable5/0/1 CM 001e.6bfa.f5bc timeout 5000 retry 0.
Apr 27 06:53:54.659: CM-CTRL-RSP:
Apr 27 06:53:54.659:     0x0000: 03 01 00
Apr 27 06:53:54.659: CM-CTRL-REQ MAC: 001e.6bfa.f5bc, timeout: 5000, retry: 0,
Apr 27 06:53:54.659: CM-CTRL-REQ tid: 0, pending: 1, num_tlvs: 1.
Apr 27 06:53:54.659: CM-CTRL decode req_msg->tid: 0, rsp_msg->tid: 0.
Apr 27 06:53:54.659:    TLV CM reinitialize Success
```

Related Commands

Command	Description
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-address	Enables debugging output for the cable modems that match the specified hardware MAC address or range of addresses.
debug cable cm-status	Enables debugging output for CM status messages on the Cisco CMTS routers.

debug cable config-file

To display information about the DOCSIS configuration files that CMTS generates internally, use the **debug cable config-file** command in the Privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

Note This command applies to configuration files created using the internal DOCSIS configuration file editor and the "cable dynamic-secret" functionality.

debug cable config-file

no debug cable config-file

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.1(2)EC	This command was introduced for Cisco uBR7200 series routers.
12.1(5)EC	Support for this command was added for the Cisco uBR7100 series routers.
12.2(4)BC1	Support was added to the Release 12.2 BC.
12.2(11)BC2	Support for this command was added to the Release 12.2 BC.

Usage Guidelines

This command shows the DOCSIS configuration files debug messages that the Cisco CMTS generated internally. These configuration files may be generated with the internal configuration file editor using the **cable config-file** command and its subcommands, or through the "dynamic shared secret" functionality using the **cable dynamic-secret** command.

Caution The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic.

Examples

The following example enables debugging for the internally generated configuration files using the conditional and config-file debug command.

Enable conditional debugs using one of the following commands:

```
Router# debug cable mac-address 0000.aaaa.bbbb
```

or

```
Router# debug cable interface cable 6/1/0 verbose
```

Enable the config-file debug using the following commands:

```
Router# debug cable config-file

CMTS config file debugging is on

Router# show debug
CMTS:
  CMTS config file debugging is on
CMTS specific:
  Debugging is on for Address 0000.aaaa.bbbb, Mask ffff.ffff.ffff
Router#
```

When debugging is turned on and the Cisco CMTS internally generates a DOCSIS configuration file for transmission to a CM that matches the conditional debug, the CMTS displays the following message:

```
Server instance created for modem 001a.c3ff.e3f0. Local 3.0.0.1 Read request from CM
001a.c3ff.e3f0 (3.0.0.119) File successfully downloaded to CM 001a.c3ff.e3f0
```

When debugging is turned on and a **verbose** keyword is used in the conditional debugs, the Cisco CMTS displays the following message:

```
Server instance created for modem 001a.c3ff.e3f0. Local 3.0.0.1
config: Searching download image for 001a.c3ff.e3f0: not found Generated tftp config file:
0x0000: 03 01 01 04 1F 01 01 01 02 04 05 F5 E1 00 03 04
0x0010: 05 F5 E1 00 04 01 00 05 04 00 00 00 00 06 02 07
0x0020: D0 07 01 01 11 2A 01 04 00 00 00 0A 02 04 00 00
0x0030: 00 0A 03 04 00 00 02 58 04 04 00 00 00 01 05 04
0x0040: 00 00 00 01 06 04 00 00 02 58 07 04 00 00 00 3C
0x0050: 12 01 0A 06 10 D8 CB DD 33 6A 2D 49 AE E1 EE DE
0x0060: 1D E5 90 4A 35 07 10 69 7B F1 67 8A 82 F0 74 F0
0x0070: 0D A1 89 B3 8D 9C F7 FF
Config file for 001a.c3ff.e3f0, Size 120 Read request from CM 001a.c3ff.e3f0 (3.0.0.119)
TFTP Server: Sent OACK to CM 001a.c3ff.e3f0 TFTP Server: ACK from CM 001a.c3ff.e3f0 for
block 0 Sending block 1, Size 120 to CM 001a.c3ff.e3f0 TFTP Server: ACK from CM
001a.c3ff.e3f0 for block 1 File successfully downloaded to CM 001a.c3ff.e3f0
```

Note See the DOCSIS 1.1 specification (revision SP-RFIv1.1-I05-000714 and above) for a description of the fields that can appear in a DOCSIS configuration file.

Related Commands

cable config-file	Creates a DOCSIS configuration file and enters configuration file mode.
access-denied	Disables access to the network.
channel-id	Specifies upstream channel ID.
cpe max	Specifies the maximum number of CPE devices allowed access.

debug cable interface	Displays debug messages for a specific cable interface, or for traffic related to a specific MAC address or Service ID (SID) on that interface.
debug cable mac-address	Displays debug information for a specific CM.
download	Specifies the filename and server IP address for downloading a new software image.
frequency	Specifies the downstream frequency.
option	Specifies options for the configuration file that are not provided for by the other commands.
privacy	Specifies privacy options for baseline privacy images.
service-class	Specifies service class definitions for the configuration file.
snmp manager	Specifies Simple Network Management Protocol (SNMP) options.
time-stamp	Enables time-stamp generation.
show running-config	Displays the current run-time configuration, which includes defined configuration files.
show startup-config	Displays the current saved configuration, which includes defined and saved configuration files.

debug cable dbs

To display debugging messages for Dynamic Bandwidth Sharing (DBS), use the **debug cable dbs** command in privileged EXEC mode. To disable DBS debugging, use the **no** form of this command.

debug cable dbs

no debug cable dbs

Syntax Description

This command has no arguments or keywords.

Command Default

DBS debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCD	This command was introduced on the Cisco uBR7225VXR and Cisco uBR7246VXR routers.

Usage Guidelines

This command is used only on the Cisco uBR7225VXR and Cisco uBR7246VXR routers.

Because this command can produce a large volume of information, use this command only when you have also enabled debugging for a particular interface, using the **debug cable interface** command.

Examples

The following example shows how to enable debugging output using the **debug cable dbs** command:

```
Router# debug cable dbs

DBS for cable 8x8 LC debugging is on
```

The following sample shows the DBS debug output:

```
Router# conf t
Router(config)# interface integrated-Cable 5/0:0
Router(config-if)# shutdown
Router(config-if)# cable dynamic-bw-sharing
Router(config-if)# no shutdown
Router(config-if)# cable rf-bandwidth-percent 30
Router(config-if)# exit
Router(config)# exit
Router# debug cable dbs
SLOT 5: Jan  8 05:01:53.074: DBS CLC: bw_sum_kbps:37500, mc_guar:4500
30%,bw_link_kbps:37500, wb_guar: wb0:5500 40%;
SLOT 5: Jan  8 05:01:53.074: RF channel 0: quantum IC: 4500
SLOT 5: Jan  8 05:01:53.074:               quantum WB[0]: 5500
SLOT 5: Jan  8 05:01:53.074:               quantum WB[1]: 0
SLOT 5: Jan  8 05:01:53.074:               quantum WB[2]: 0
SLOT 5: Jan  8 05:01:53.074:               quantum WB[3]: 0
```

```
SLOT 5: Jan  8 05:01:53.074:                quantum WB[4]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[5]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[6]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[7]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[8]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[9]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[10]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[11]: 0
SLOT 5: Jan  8 05:01:53.074: Slot 5: add RF 0, cr7200_clc_dbs_rfchannel_bitmap 255,
policy_rate 4687, tokens 46870, dbs pct 100
SLOT 5: Jan  8 05:01:53.074: DBS CLC: bw_sum_kbps:37500, mc_guar:4500
30%,bw_link_kbps:37500, wb_guar: wb0:5500 40%;
SLOT 5: Jan  8 05:01:53.074: RF channel 0: quantum IC: 4500
SLOT 5: Jan  8 05:01:53.074:                quantum WB[0]: 5500
SLOT 5: Jan  8 05:01:53.074:                quantum WB[1]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[2]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[3]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[4]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[5]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[6]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[7]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[8]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[9]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[10]: 0
SLOT 5: Jan  8 05:01:53.074:                quantum WB[11]: 0
SLOT 5: Jan  8 05:01:53.074: Slot 5: add RF 0, cr7200_clc_dbs_rfchannel_bitmap 255,
policy_rate 4687, tokens 46870, dbs pct 100
```

Related Commands

Command	Description
cable dynamic-bw-sharing	Enables dynamic bandwidth sharing on a specific integrated cable or wideband cable interface.
show interface wideband-cable	Displays the current configuration and status for a wideband channel.
show interface integrated-cable	Displays the current configuration and status for an integrated channel.

debug cable dcc

To display information about DOCSIS 1.1 Dynamic Channel Change (DCC) messages, use the **debug cable dcc** command in Privileged EXEC mode. To disable debugging output for DCC messages, use the **no** form of this command.

debug cable dcc

no debug cable dcc

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(7)CX	This command was introduced for DOCSIS 1.1 operation.
12.2(4)BC1	Support was added to the Release 12.2 BC train.

Usage Guidelines

This command shows debugging messages about the DCC Request (DCC-REQ) message that the CMTS sends to the CM, and about the DCC Response (DCC-RSP) message that the CM sends in reply.

Caution The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. In particular, avoid using the **debug cable dcc** command on an interface with a significant number of CMs, because the resulting volume of debug output could impact system performance. Cisco recommends that when you use the **debug cable dcc** command, you limit its output to a particular interface or CM, using the **debug cable interface** command.

Tip To display the contents of the DCC messages, enable TLV debugging with the **debug cable tlvs** command.

Note See the DOCSIS 1.1 specification (revision SP-RFIv1.1-I05-000714 and above) for information on the content and format of the DCC-REQ and DCC-RSP messages.

Examples

The following examples show typical output for the **debug cable dcc** command. The following example shows the debug output from a DCC-REQ message that instructs the CM to change its upstream channel to channel 1, using the default initialization technique (option 0, the CM is to reinitialize its MAC layer, requiring a complete reregistration):

```
Router# debug cable int c3/0
Router# debug cable dcc
CMTS DCC encodings debugging is on
Router#
00:02:57: Found DCC TLV
00:02:57:       US Channel ID 1
00:02:57: received a DCC_RSP.  upstream = 1.
```

The following example shows the debug output from a DCC-REQ message that instructs the CM to change its upstream channel to channel 1, using initialization technique 4 (the CM is to continue its normal operations after synchronizing to the new channel).

```
0:04:29: Found DCC TLV
00:04:29:       US Channel ID 1
00:04:29: Found DCC TLV
00:04:29:       Init Tech 4
00:04:29: received a DCC_RSP.  upstream = 1.
00:04:29: CM new state = 15
00:04:30: DCC-RSP-NEW-TIMEOUT: CmMac->00ac.0000.0070 OrgId->32770
00:04:30: DCC-RSP-ARRIVE is lost.
00:04:30: received a DCC_RSP.  upstream = 0.
00:04:30: ERROR: Received DCC-RSP-ARRIVE w/o DCC-REQ.
00:04:30: DCC-ACK Message Contents:
00:04:30: 0x0000: C2 00 00 1A 00 00 00 30 EB 15 2E 97 00 AC 00 00
00:04:30: 0x0010: 00 70 00 08 00 00 03 02 19 00 80 02
00:04:30: DCC-ACK-SENT: CM->0030.eb15.2e97 TranscId->32770
```

The following example shows the debug output from a DCC-REQ message that instructs the CM to change its upstream channel to channel 2, using initialization technique 4, and substituting a new service flow ID (SFID) of 160 for the current SFID of 14.

```
00:05:19: Found DCC TLV
 00:05:19:       US Channel ID 2
 00:05:19: Found DCC TLV
 00:05:19:       Init Tech 4
 00:05:19: Found DCC TLV
 00:05:19:       SFID Substitute old 14 new 160
 00:05:19: received a DCC_RSP.  upstream = 0.
 00:05:19: CM new state = 15
 00:05:20: DCC-RSP-NEW-TIMEOUT: CmMac->00ac.0000.0070 OrgId->32771
 00:05:20: DCC-RSP-ARRIVE is lost.
 00:05:20: received a DCC_RSP.  upstream = 1.
 00:05:20: ERROR: Received DCC-RSP-ARRIVE w/o DCC-REQ.
 00:05:20: DCC-ACK Message Contents:
 00:05:20: 0x0000: C2 00 00 1A 00 00 00 30 EB 15 2E 97 00 AC 00 00
 00:05:20: 0x0010: 00 70 00 08 00 00 03 02 19 00 80 03
 00:05:20: DCC-ACK-SENT: CM->0030.eb15.2e97 TranscId->32771
```

The following example shows the debug output from a DCC-REQ message that specifies an upstream channel with a frequency that is not a multiple of 62,500 Hz, which is required by the DOCSIS specification:

```
00:07:28: Found DCC TLV
 00:07:28:       US Channel ID 2
 00:07:28: Found DCC TLV
 00:07:28:       Init Tech 4
 00:07:28: Found DCC TLV
 00:07:28:       ERROR: DS frequency not a multiple of 62500Hz.
```

```
00:07:28: Not able to parse the TLVs.
```

Related Commands

debug cable tlvs	Displays debugging messages about the TLV values used for service flow encodings, classifier encodings, and PHS rules.

debug cable dci

To display information about DOCSIS 1.1 Device Class Identification (DCI) messages, use the **debug cable dci** command in privileged EXEC mode. To disable debugging output for DCI messages, use the **no** form of this command.

debug cable dci

no debug cable dci

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(4)CX	This command was introduced for DOCSIS 1.1 operation.
12.2(4)BC1	Support was added to the Release 12.2 BC train.

Usage Guidelines

This command shows debugging messages about the DCI-REQ messages that the Cisco CMTS receives from CMs.

Note See the DOCSIS 1.1 specification (revision SP-RFIv1.1-I05-000714 and above) for additional information on the DCI-REQ, DCI-RSP, and Upstream Transmitter Disable (UP-DIS) messages.

Examples

The following example shows typical output displayed by the **debug cable dci** command:

```
Router# debug cable interface c3/0
Router# debug cable dci
CMTS dci debugging is on
Router#
DCI-REQ: CM->1234.5678.abcd SID->1
Device Class 1st half->0000000000000000 Device Class 2nd half->0000000000000001
```

Related Commands

cable dci-response	Configures how a cable interface responds to DCI-REQ messages coming from CMs on that interface.
cable dci-upstream-disable	Configures the cable interface so that it sends an Upstream Transmitter Disable (UP-DIS) message in response to a DCI-REQ message from a particular CM.

debug cable dhcp

To enable debugging of the Dynamic Host Configuration Protocol (DHCP) when it is used on the cable interface, use the **debug cable dhcp** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable dhcp

no debug cable dhcp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)BC3	This command was introduced.

Usage Guidelines

The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. In particular, avoid using the **debug cable dhcp** command on an interface with a significant number of CMs, because the resulting volume of debug output could impact system performance. Cisco recommends that when you use the **debug cable dhcp** command, you limit its output to a particular interface or CM, using the **debug cable interface** or **debug cable mac-address** commands.

Examples

The following shows typical debugging messages for the **debug cable dhcp** command for a particular cable modem that is acquiring a DHCP address:

```
Router# debug cable dhcp
CMTS dhcp debugging is on

Router# debug cable mac-address 0020.4072.7418
Router# clear cable modem 0020.4072.7418 reset
Router#
DHCPINFO hwidb Cable3/1 MAC 0020.4072.7418 SID 2 dhcp_op 1
DHCPINFO wan info circuit 80010006 remote_system_id 0020.4072.7418 remote_device_class 0
DHCP cmts_helper_forward idb Cable3/1.1 client 0020.4072.7418 31.1.1.1
DHCPGLEAN input idb Cable3/1.1 MAC 0020.4072.7418 type 1
DHCPGLEAN hwidb Cable3/1 found for MAC 0020.4072.7418
DHCPGLEAN cmts glean hwidb Cable3/0 mac 0020.4072.7418 sid 2 ipaddr 0.0.0.0
DHCPGLEAN input idb Cable3/1.1 MAC 0020.4072.7418 type 2
DHCPGLEAN hwidb Cable3/1 found for MAC 0020.4072.7418
DHCPGLEAN SID 2 MAC 0020.4072.7418 on Cable3/0 mapped to swidb Cable3/1.1 ipaddr 31.0.0.2
DHCPINFO hwidb Cable3/1 MAC 0020.4072.7418 SID 2 dhcp_op 1
DHCPINFO wan info circuit 80010006 remote_system_id 0020.4072.7418 remote_device_class 0
DHCP cmts_helper_forward idb Cable3/1.1 client 0020.4072.7418 31.1.1.1 Host
DHCPGLEAN input idb Cable3/1.1 MAC 0020.4072.7418 type 3
DHCPGLEAN hwidb Cable3/1 found for MAC 0020.4072.7418
DHCPGLEAN cmts glean hwidb Cable3/0 mac 0020.4072.7418 sid 2 ipaddr 0.0.0.0
DHCPGLEAN input idb Cable3/1.1 MAC 0020.4072.7418 type 5
```

```
DHCPGLEAN hwidb Cable3/1 found for MAC 0020.4072.7418
DHCPGLEAN cmts glean hwidb Cable3/0 mac 0020.4072.7418 sid 0 ipaddr 31.0.0.2
DHCPGLEAN SID 2 MAC 0020.4072.7418 on Cable3/0 mapped to swidb Cable3/1.1 ipaddr 31.0.0.2
```

Related Commands

Command	Description
debug cable arp	Enables debugging of the Address Resolution Protocol on the cable interface.
debug cable encap	Enables debugging of encapsulated PPPoE packets that are transmitted over the cable interface.
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for a particular CM.

debug cable dsg

To enable general, downstream channel descriptor (DCD), or packet-related debugging for Advanced-Mode DOCSIS Set-Top Gateway (A-DSG) on a Cisco CMTS router, use the **debug cable dsg** command in privileged EXEC mode. To disable A-DSG debugging, us the **no** form of this command.

debug cable dsg [**dcd** | **pkt** | **name**]

no debug cable dsg

Syntax Description

dcd	(Optional) Enables DCD-related debugging. Can be combined with **pkt**.
pkt	(Optional) Enables packet-related debugging. Can be combined with **dcd**.
name	(Optional) Enables DSG host name-related debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(15)BC2	This command was introduced for the Cisco uBR7100 series and Cisco uBR7246VXR routers.
12.3(9a)BC	This command was introduced for the Cisco uBR10012 router.
12.3(13a)BC	This command was modified to begin support of A-DSG on the Cisco uBR10012 router and Cisco uBR7200 series routers. The **dcd** and **pkt** keyword options were added.
12.2(33)SCA	This command was integrated into Cisco IOS Release 12.2(33)SCA. Support for the Cisco uBR7225VXR router was added.
12.2(33)SCG	This command was modified. A new keyword, **name**, was added to enable DSG name-related debugging.

Usage Guidelines

Debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic because the resulting volume of debug output could impact system performance. Because this command can produce a large volume of information, use this command only when you have also enabled debugging for a particular interface or MAC address, using the **debug cable interface** and **debug cable mac-address** commands, respectively.

When using the **debug cable dsg** command with the **dcd** keyword, it shows DCD counters. If the configuration is changed, the whole DCD message content is displayed, including the MAC header.

Related Commands

Command	Description
show cable dsg tunnel	Displays information about A-DSG tunnel configuration on a Cisco CMTS router.
show interface	Displays general interface information for the specified or all interfaces.
show interface cable dsg downstream	Displays interface configuration and status information for A-DSG downstreams on a Cisco CMTS router.

debug cable dynamic-qos subscriber

To enable debugging of the call trace functionality on the Cisco CMTS router for a particular subscriber, use the **debug cable dynamic-qos subscriber** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug cable dynamic-qos subscriber [ipv4-address [mask-address] | ipv6-address] [verbose]

no debug cable dynamic-qos subscriber [ipv4-address [mask-address] | ipv6-address] [verbose]

Syntax Description

ipv4-address	(Optional) IPv4 address of the subscriber.
mask-address	(Optional) IPv4 mask for the specified IPv4 address. If you do not specify the mask address, the default mask (255.255.255.255) is used.
ipv6-address	(Optional) IPv6 address of the subscriber.
verbose	(Optional) Provides detailed debugging information with the verbose output.

Command Default

Debugging is not enabled for any of the subscribers.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS Release 12.2(33)SCF	This command was introduced.

Usage Guidelines

The **debug cable dynamic-qos subscriber** command enables debugging for a particular subscriber. For the debugs to get printed, you must enable call trace debugging using the **debug cable dynamic-qos trace** command.

The following two debug commands enable most of the subscriber-based debug logs relevant to PacketCable, PacketCable Multimedia (PCMM), and dynamic quality of service (DQoS) lite subscriber debugging:

- **debug cable dynamic-qos trace**
- **debug cable dynamic-qos subscriber**

Examples

The following example shows debugging information for all the configured subscribers on the Cisco uBR10012 router in Cisco IOS Release 12.2(33)SCF:

```
Router# debug cable dynamic-qos subscriber

CMTS dynqos subscriber debugging is on ubr10k#
*Mar 17 08:32:27.135: Pktcbl(gdb): Created gate IE on Cable7/1/0, gateid = 10804
2
*Mar 17 08:32:27.135: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:27.135: Pktcbl(mm): Change profile 0 qos 0
```

```
*Mar 17 08:32:27.135: Pktcbl(gdb): IPC timer [id 108042] [10000 msec]
*Mar 17 08:32:27.135: Pktcbl(gdb): Started gate [id 108042] timer [type 8] [1000
0 msec]
*Mar 17 08:32:27.135: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:27.135: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 08:32:27.135: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 08:32:27.135: Pktcbl(gdb): MM Authorized Profile
*Mar 17 08:32:27.135: Pktcbl(gdb): MM Reserved Profile
*Mar 17 08:32:27.135: Pktcbl(gdb): MM Committed Profile
*Mar 17 08:32:27.135: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 08:32:27.179: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 108
042 rsp 1 state new(4) old(2)
*Mar 17 08:32:27.179: Pktcbl(gdb): Cancelled gate [id 108042] timer [type 8]
*Mar 17 08:32:27.179: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:27.179: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:27.179: Pktcbl(gdb): Started gate [id 108042] timer [type 3] [0 ms
ec]
*Mar 17 08:32:27.179: PktCbl(d2r): extract id: gate=108042, resource=74
*Mar 17 08:32:27.179: PktCbl(d2r): extract id: gate=108042, resource=74
*Mar 17 08:32:27.179: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=108042 d
ir=1 gie=26DD5C98
*Mar 17 08:32:27.179: Pktcbl(gdb): TOS Overwrite Gate=108042 DSCP=0xD0 mask=0xF
*Mar 17 08:32:27.179: PktCbl(d2r): extract id: gate=108042, resource=74
*Mar 17 08:32:27.179: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 108042 sfid 74
*Mar 17 08:32:27.179: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:27.183: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:27.183: Pktcbl(mm): Building GCP message, added obj TRANSACTION
ubr10k# ID      ; len:8 padding:0
*Mar 17 08:32:27.183: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 08:32:27.183: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 08:32:27.183: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 08:32:27.183: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 08:32:27.183: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 08:32:27.183:  --- Pktcbl: Sending GCP message ------
*Mar 17 08:32:27.183: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 08:32:27.183:     transaction id   : 0x1
*Mar 17 08:32:27.183:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 08:32:27.183: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 08:32:27.183:     AM ID            : 0x1 (0/1)
*Ma
ubr10k#r 17 08:32:27.183: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 08:32:27.183:     Addr             : 21.21.2.10
*Mar 17 08:32:27.183: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 08:32:27.183:     GateID           : 108042 (0x1A60A)
*Mar 17 08:32:27.183: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 08:32:27.183:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 08:32:27.183:  ---------------------------------------
SLOT 7/1: Mar 17 08:32:27.152: Pktcbl(gdb): Gate ID 108042 not found in gdb, pkt
cbl_find_gate_ie.
SLOT 5/0: Mar 17 08:32:27.151: Pktcbl(gdb): Gate ID 108042 not found in gdb, pkt
cbl_find_gate_ie.
ubr10k#
*Mar 17 08:32:56.656: Pktcbl(mm): Received GATE SET message, tid=0x2
*Mar 17 08:32:56.656: --- Pktcbl(mm): Received GCP message ------
*Mar 17 08:32:56.656: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 08:32:56.656:     transaction id   : 0x2
*Mar 17 08:32:56.656:     gcp cmd          : 4 (GATE SET)
```

```
*Mar 17 08:32:56.656: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 08:32:56.656:     AM ID            : 0x1 (0/1)
*Mar 17 08:32:56.656: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 08:32:56.656:     Addr             : 21.21.2.10
*Mar 17 08:32:56.656: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 08:32:56.656:     GateID           : 108042 (0x1A60A)
*Mar 17 08:32:56.656: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 08:32:56.656:     flag             : 0x3
*Mar 17 08:32:56.656:     dscp             : 0xD8
*Mar 17 08:32:56.656:     dscp tos mask    : 0xF0
*Mar 17 08:32:56.656:     Timers  t1       : 0,  t2 : 0
*Mar 17 08:32:56.656:             t3       : 0,  t4 : 0
*Mar 17 08:32:56.656:     session class    : 0x0
*Mar 17 08:32:56.656: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 08:32:56.656:     envelope         : 0x7
*Mar 17 08:32:56.656:     service number   : 0x0
*Mar 17 08:32:56.656:     Authorized :
*Mar 17 08:32:56.656: Request Xmit Policy: 0x17F
*Mar 17 08:32:56.656: Grant size         : 232
*Mar 17 08:32:56.656: Grant Per Interval : 2
*Mar 17 08:32:56.656: Grant Interval     : 20000
*Mar 17 08:32:56.656: Tolerated Jitter   : 800
*Mar 17 08:32:56.656: Required Mask      : 0
*Mar 17 08:32:56.656: Forbidden Mask     : 0
*Mar 17 08:32:56.656: Aggr Rule Mask     : 0
*Mar 17 08:32:56.656:     Reserved   :
*Mar 17 08:32:56.656: Request Xmit Policy: 0x17F
*Mar 17 08:32:56.656: Grant size         : 232
*Mar 17 08:32:56.656: Grant Per Interval : 2
*Mar 17 08:32:56.656: Grant Interval     : 20000
*Mar 17 08:32:56.656: Tolerated Jitter   : 800
*Mar 17 08:32:56.656: Required Mask      : 0
*
ubr10k#Mar 17 08:32:56.656: Forbidden Mask     : 0
*Mar 17 08:32:56.656: Aggr Rule Mask     : 0
*Mar 17 08:32:56.656: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 08:32:56.656:     protocol         : 1
*Mar 17 08:32:56.656:     dscp             : 0x0
*Mar 17 08:32:56.656:     dscp tos mask    : 0x0
*Mar 17 08:32:56.656:     src/port         : 9.9.1.95 0
*Mar 17 08:32:56.656:     dest/port        : 2.39.26.11 0
*Mar 17 08:32:56.656:     priority         : 64
*Mar 17 08:32:56.656: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 08:32:56.656:     protocol         : 1
*Mar 17 08:32:56.656:     dscp             : 0x0
*Mar 17 08:32:56.656:     dscp tos mask    : 0x0
*Mar 17 08:32:56.656:     src/port         : 9.9.1.43 0
*Mar 17 08:32:56.656:     dest/port        : 2.39.26.19 0
*Mar 17 08:32:56.656:     priority         : 64
*Mar 17 08:32:56.656: OPAQUE               : Object.[snum/stype/
ubr10k#len 11/1/12]
*Mar 17 08:32:56.656:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 08:32:56.656: ------------------------------------
*Mar 17 08:32:56.656:  Backup gate IE [108042]
*Mar 17 08:32:56.656: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:56.656: Pktcbl(mm): Change profile 1 qos 1
*Mar 17 08:32:56.656: Pktcbl(gdb): IPC timer [id 108042] [10000 msec]
*Mar 17 08:32:56.656: Pktcbl(gdb): Started gate [id 108042] timer [type 8] [1000
0 msec]
*Mar 17 08:32:56.656: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:56.656: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 08:32:56.656: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 08:32:56.656: Pktcbl(gdb): MM Authorized Profile
*Mar 17 08:32:56.660: Pktcbl(gdb): MM Reserved Profile
```

```
*Mar 17 08:32:56.660: Pktcbl(gdb): MM Committed Profile
*Mar 17 08:32:56.660: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 08:32:56.660: Classifier prototype: 1, src: 9.9.1.43, dest: 2.39.26.19,
src port: 0, dest port: 0
*Mar 17 08:32:56.696: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 108
042 rsp 1 state new(4) old(4)
*Mar 17 08:32:56.696: Pktcbl(gdb): Cancelled gate [id 108042] timer [type 8]
*Mar 17 08:32:56.696: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:56.696: Pktcbl(gdb): Started gate [id 108042] timer [type 3] [0 ms
ec]
*Mar 17 08:32:56.696: Pktcbl(gdb): Cleanup saved gate IE info, gate(108042)
*Mar 17 08:32:56.696: PktCbl(d2r): extract id: gate=108042, resource=74
*Mar 17 08:32:56.696: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=108042 d
ir=1 gie=26DD5C98
*Mar 17 08:32:56.696: Pktcbl(gdb): TOS Overwrite Gate=108042 DSCP=0xD0 mask=0xF
*Mar 17 08:32:56.696: PktCbl(d2r): extract id: gate=108042, resource=74
*Mar 17 08:32:56.696: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 108042 sfid 74
*Mar 17 08:32:56.696: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:56.696: Pktcbl(gdb): Found Cable7/1/0 for Gate=108042 21.21.2.10
*Mar 17 08:32:56.696: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 08:32:56.696: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 08:32:56.696: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 08:32:56.696: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 08:32:56.696: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 08:32:56.696: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 08:32:56.696:  --- Pktcbl: Sending GCP message ------
*Mar 17 08:32:56.696: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 08:32:56.696:     transaction id   : 0x2
*Mar 17 08:32:56.696:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 08:32:56.696: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 08:32:56.696:     AM ID            : 0x1 (0/1)
*Mar 17 08:32:56.696: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 08:32:56.696:     Addr             : 21.21.2.10
*Mar 17 08:32:56.696: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 08:32:56.696:     GateID           : 108042 (0x1A60A)
*Mar 17 08:32:56.696: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 08:32:56.696:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 08:32:56.696:  ---------------------------------------
```

The following example shows debugging information based on the subscriber IP address on the Cisco uBR10012 router in Cisco IOS Release 12.2(33)SCF:

```
Router# debug cable dynamic-qos subscriber 21.21.2.10

CMTS dynqos subscriber debugging is on

*Mar 17 02:29:13.293: Pktcbl(gdb): Created new Subscriber IE for 21.21.2.10
*Mar 17 02:29:13.293: Pktcbl(gdb): Updated subscriber IE [subs addr: 21.21.2.10,
 gate: 26118]
*Mar 17 02:29:13.297: Pktcbl(gdb): Created gate IE on Cable7/1/0, gateid = 26118
*Mar 17 02:29:13.297: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:13.297: Pktcbl(mm): Change profile 0 qos 0
*Mar 17 02:29:13.297: Pktcbl(gdb): IPC timer [id 26118] [10000 msec]
*Mar 17 02:29:13.297: Pktcbl(gdb): Started gate [id 26118] timer [type 8] [10000
 msec]
```

```
*Mar 17 02:29:13.297: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:13.297: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:29:13.297: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:29:13.297: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:29:13.297: Pktcbl(gdb): MM Reserved Profile
*Mar 17 02:29:13.297: Pktcbl(gdb): MM Committed Profile
*Mar 17 02:29:13.297: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:29:13.333: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 261
18 rsp 1 state new(4) old(2)
*Mar 17 02:29:13.333: Pktcbl(gdb): Cancelled gate [id 26118] timer [type 8]
*Mar 17 02:29:13.333: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:13.333: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:13.333: Pktcbl(gdb): Started gate [id 26118] timer [type 3] [0 mse
c]
*Mar 17 02:29:13.333: PktCbl(d2r): extract id: gate=26118, resource=62
*Mar 17 02:29:13.333: PktCbl(d2r): extract id: gate=26118, resource=62
*Mar 17 02:29:13.333: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=26118 di
r=1 gie=2A8EA2D0
*Mar 17 02:29:13.333: Pktcbl(gdb): TOS Overwrite Gate=26118 DSCP=0xD0 mask=0xF
*Mar 17 02:29:13.337: PktCbl(d2r): extract id: gate=26118, resource=62
*Mar 17 02:29:13.337: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 26118 sfid 62
*Mar 17 02:29:13.337: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*M
ubr10k#ar 17 02:29:13.337: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.
10
*Mar 17 02:29:13.337: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:29:13.337: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:29:13.337: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:29:13.337: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:29:13.337: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:29:13.337: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:29:13.337:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:29:13.337: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:29:13.337:     transaction id   : 0x3
*Mar 17 02:29:13.337:     gcp cmd          :
ubr10k# 5 (GATE SET ACK)
*Mar 17 02:29:13.337: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:29:13.337:     AM ID            : 0x1 (0/1)
*Mar 17 02:29:13.337: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:29:13.337:     Addr             : 21.21.2.10
*Mar 17 02:29:13.337: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:29:13.337:     GateID           : 26118 (0x6606)
*Mar 17 02:29:13.337: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:29:13.337:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:29:13.337:  ------------------------------------
SLOT 7/1: Mar 17 02:29:13.307: Pktcbl(gdb): Gate ID 26118 not found in gdb, pktc
bl_find_gate_ie.
SLOT 5/0: Mar 17 02:29:13.307: Pktcbl(gdb): Gate ID 26118 not found in gdb, pktc
bl_find_gate_ie.
ubr10k#
*Mar 17 02:29:50.547: Pktcbl(mm): Received GATE SET message, tid=0x4
*Mar 17 02:29:50.547: --- Pktcbl(mm): Received GCP message ------
*Mar 17 02:29:50.547: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:29:50.547:     transaction id   : 0x4
*Mar 17 02:29:50.547:     gcp cmd          : 4 (GATE SET)
*Mar 17 02:29:50.547: AM ID                : Object.[snum/stype/len 2/1/8]
```

```
*Mar 17 02:29:50.547:     AM ID            : 0x1 (0/1)
*Mar 17 02:29:50.547: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:29:50.547:     Addr             : 21.21.2.10
*Mar 17 02:29:50.547: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:29:50.547:     GateID           : 26118 (0x6606)
*Mar 17 02:29:50.547: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 02:29:50.547:     flag             : 0x3
*Mar 17 02:29:50.547:     dscp             : 0xD8
*Mar 17 02:29:50.547:     dscp tos mask    : 0xF0
*Mar 17 02:29:50.547:     Timers  t1       : 0,  t2 : 0
*Mar 17 02:29:50.547:             t3       : 0,  t4 : 0
*Mar 17 02:29:50.547:     session class    : 0x0
*Mar 17 02:29:50.547: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 02:29:50.547:     envelope         : 0x7
*Mar 17 02:29:50.547:     service number   : 0x0
*Mar 17 02:29:50.547:     Authorized :
*Mar 17 02:29:50.547: Request Xmit Policy: 0x17F
*Mar 17 02:29:50.547: Grant size         : 232
*Mar 17 02:29:50.547: Grant Per Interval : 2
*Mar 17 02:29:50.547: Grant Interval     : 20000
*Mar 17 02:29:50.547: Tolerated Jitter   : 800
*Mar 17 02:29:50.547: Required Mask      : 0
*Mar 17 02:29:50.547: Forbidden Mask     : 0
*Mar 17 02:29:50.547: Aggr Rule Mask     : 0
*Mar 17 02:29:50.547:     Reserved   :
*Mar 17 02:29:50.547: Request Xmit Policy: 0x17F
*Mar 17 02:29:50.547: Grant size         : 232
*Mar 17 02:29:50.547: Grant Per Interval : 2
*Mar 17 02:29:50.547: Grant Interval     : 20000
*Mar 17 02:29:50.547: Tolerated Jitter   : 800
*Mar 17 02:29:50.547: Required Mask      : 0
*Ma
ubr10k#r 17 02:29:50.547: Forbidden Mask     : 0
*Mar 17 02:29:50.547: Aggr Rule Mask     : 0
*Mar 17 02:29:50.547: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:29:50.547:     protocol         : 1
*Mar 17 02:29:50.547:     dscp             : 0x0
*Mar 17 02:29:50.547:     dscp tos mask    : 0x0
*Mar 17 02:29:50.551:     src/port         : 9.9.1.95 0
*Mar 17 02:29:50.551:     dest/port        : 2.39.26.11 0
*Mar 17 02:29:50.551:     priority         : 64
*Mar 17 02:29:50.551: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:29:50.551:     protocol         : 1
*Mar 17 02:29:50.551:     dscp             : 0x0
*Mar 17 02:29:50.551:     dscp tos mask    : 0x0
*Mar 17 02:29:50.551:     src/port         : 9.9.1.43 0
*Mar 17 02:29:50.551:     dest/port        : 2.39.26.19 0
*Mar 17 02:29:50.551:     priority         : 64
*Mar 17 02:29:50.551: OPAQUE               : Object.[snum/stype/le
ubr10k#n 11/1/12]
*Mar 17 02:29:50.551:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:29:50.551: ---------------------------------------
*Mar 17 02:29:50.551:  Backup gate IE [26118]
*Mar 17 02:29:50.551: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:50.551: Pktcbl(mm): Change profile 1 qos 1
*Mar 17 02:29:50.551: Pktcbl(gdb): IPC timer [id 26118] [10000 msec]
*Mar 17 02:29:50.551: Pktcbl(gdb): Started gate [id 26118] timer [type 8] [10000
 msec]
*Mar 17 02:29:50.551: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:50.551: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:29:50.551: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:29:50.551: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:29:50.551: Pktcbl(gdb): MM Reserved Profile
*Mar 17 02:29:50.551: Pktcbl(gdb): MM Committed Profile
```

```
*Mar 17 02:29:50.551: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:29:50.551: Classifier prototype: 1, src: 9.9.1.43, dest: 2.39.26.19,
src port: 0, dest port: 0
*Mar 17 02:29:50.583: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 261
18 rsp 1 state new(4) old(4)
*Mar 17 02:29:50.583: Pktcbl(gdb): Cancelled gate [id 26118] timer [type 8]
*Mar 17 02:29:50.583: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:50.583: Pktcbl(gdb): Started gate [id 26118] timer [type 3] [0 mse
c]
*Mar 17 02:29:50.583: Pktcbl(gdb): Cleanup saved gate IE info, gate(26118)
*Mar 17 02:29:50.583: PktCbl(d2r): extract id: gate=26118, resource=62
*Mar 17 02:29:50.583: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=26118 di
r=1 gie=2A8EA2D0
*Mar 17 02:29:50.583: Pktcbl(gdb): TOS Overwrite Gate=26118 DSCP=0xD0 mask=0xF
*Mar 17 02:29:50.583: PktCbl(d2r): extract id: gate=26118, resource=62
*Mar 17 02:29:50.583: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 26118 sfid 62
*Mar 17 02:29:50.583: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:50.587: Pktcbl(gdb): Found Cable7/1/0 for Gate=26118 21.21.2.10
*Mar 17 02:29:50.587: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:29:50.587: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:29:50.587: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:29:50.587: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:29:50.587: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:29:50.587: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:29:50.587:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:29:50.587: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:29:50.587:     transaction id   : 0x4
*Mar 17 02:29:50.587:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 02:29:50.587: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:29:50.587:     AM ID            : 0x1 (0/1)
*Mar 17 02:29:50.587: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:29:50.587:     Addr             : 21.21.2.10
*Mar 17 02:29:50.587: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:29:50.587:     GateID           : 26118 (0x6606)
*Mar 17 02:29:50.587: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:29:50.587:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:29:50.587:  ---------------------------------------
```

The following example shows debugging information based on the subscriber IP address when using the **debug cable dynamic-qos subscriber** command with the **verbose** keyword on the Cisco uBR10012 router in Cisco IOS Release 12.2(33)SCF:

```
Router# debug cable dynamic-qos subscriber 21.21.2.10 verbose

CMTS dynqos subscriber verbose debugging is on ubr10k#

*Mar 17 02:31:52.633: Pktcbl(mm): Received GATE SET message, tid=0x5
*Mar 17 02:31:52.633: --- Pktcbl(mm): Received GCP message ------
*Mar 17 02:31:52.633: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:31:52.633:     transaction id   : 0x5
*Mar 17 02:31:52.633:     gcp cmd          : 4 (GATE SET)
*Mar 17 02:31:52.633: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:31:52.633:     AM ID            : 0x1 (0/1)
*Mar 17 02:31:52.633: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
```

```
*Mar 17 02:31:52.633:     Addr             : 21.21.2.10
*Mar 17 02:31:52.633: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 02:31:52.633:     flag             : 0x3
*Mar 17 02:31:52.633:     dscp             : 0xD8
*Mar 17 02:31:52.633:     dscp tos mask    : 0xF0
*Mar 17 02:31:52.633:     Timers  t1       : 0,  t2 : 0
*Mar 17 02:31:52.633:             t3       : 0,  t4 : 0
*Mar 17 02:31:52.633:     session class    : 0x0
*Mar 17 02:31:52.633: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 02:31:52.633:     envelope         : 0x7
*Mar 17 02:31:52.633:     service number   : 0x0
*Mar 17 02:31:52.633:     Authorized :
*Mar 17 02:31:52.633: Request Xmit Policy: 0x17F
*Mar 17 02:31:52.633: Grant size         : 232
*Mar 17 02:31:52.633: Grant Per Interval : 1
*Mar 17 02:31:52.633: Grant Interval     : 20000
*Mar 17 02:31:52.633: Tolerated Jitter   : 800
*Mar 17 02:31:52.633: Required Mask      : 0
*Mar 17 02:31:52.633: Forbidden Mask     : 0
*Mar 17 02:31:52.633: Aggr Rule Mask     : 0
*Mar 17 02:31:52.633:     Reserved   :
*Mar 17 02:31:52.633: Request Xmit Policy: 0x17F
*Mar 17 02:31:52.633: Grant size         : 232
*Mar 17 02:31:52.633: Grant Per Interval : 1
*Mar 17 02:31:52.633: Grant Interval     : 20000
*Mar 17 02:31:52.633: Tolerated Jitter   : 800
*Mar 17 02:31:52.633: Required Mask      : 0
*Mar 17 02:31:52.633: Forbidden Mask     : 0
*Mar 17 02:31:52.633: Aggr Rule Mask     : 0
*Mar 17 02:31:52.633: CLASSIFIER           : Obj
ubr10k#ect.[snum/stype/len 6/1/24]
*Mar 17 02:31:52.633:     protocol         : 1
*Mar 17 02:31:52.633:     dscp             : 0x0
*Mar 17 02:31:52.633:     dscp tos mask    : 0x0
*Mar 17 02:31:52.633:     src/port         : 9.9.1.95 0
*Mar 17 02:31:52.637:     dest/port        : 2.39.26.11 0
*Mar 17 02:31:52.637:     priority         : 64
*Mar 17 02:31:52.637: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:31:52.637:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:31:52.637: ---------------------------------------
*Mar 17 02:31:52.637: Pktcbl(gdb): Allocating gate entry 0/2. Instance cnt: 1, g
ate_count is now 3
*Mar 17 02:31:52.637: Pktcbl(gdb): Updated subscriber IE [subs addr: 21.21.2.10,
 gate: 42502]
*Mar 17 02:31:52.637: Pktcbl(gdb): Created gate IE on Cable7/1/0, gateid = 42502
*Mar 17 02:31:52.637: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:31:52.637: Pktcbl(mm)
ubr10k#: Change profile 0 qos 0
*Mar 17 02:31:52.637: Pktcbl(gdb): IPC timer [id 42502] [10000 msec]
*Mar 17 02:31:52.637: Pktcbl(gdb): Started gate [id 42502] timer [type 8] [10000
 msec]
*Mar 17 02:31:52.637: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:31:52.637: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:31:52.637: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:31:52.637: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:31:52.637: Pktcbl(gdb): MM Reserved Profile
*Mar 17 02:31:52.637: Pktcbl(gdb): MM Committed Profile
*Mar 17 02:31:52.637: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:31:52.669: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 425
02 rsp 1 state new(4) old(2)
*Mar 17 02:31:52.669: Pktcbl(gdb): Cancelled gate [id 42502] timer [type 8]
*Mar 17 02:31:52.669: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:31:52.669: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
```

```
*Mar 17 02:31:52.669: Pktcbl(gdb): Started gate [id 42502] timer [type 3] [0 mse
c]
*Mar 17 02:31:52.669: PktCbl(d2r): extract id: gate=42502, resource=63
*Mar 17 02:31:52.669: PktCbl(d2r): extract id: gate=42502, resource=63
*Mar 17 02:31:52.669: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=42502 di
r=1 gie=265BD720
*Mar 17 02:31:52.669: Pktcbl(gdb): TOS Overwrite Gate=42502 DSCP=0xD0 mask=0xF
*Mar 17 02:31:52.673: PktCbl(d2r): extract id: gate=42502, resource=63
*Mar 17 02:31:52.673: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 42502 sfid 63
*Mar 17 02:31:52.673: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:31:52.673: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:31:52.673: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:31:52.673: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:31:52.673: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:31:52.673: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:31:52.673: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:31:52.673: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:31:52.673:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:31:52.673: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:31:52.673:     transaction id   : 0x5
*Mar 17 02:31:52.673:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 02:31:52.673: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:31:52.673:     AM ID            : 0x1 (0/1)
*Mar 17 02:31:52.673: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:31:52.673:     Addr             : 21.21.2.10
*Mar 17 02:31:52.673: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:31:52.673:     GateID           : 42502 (0xA606)
*Mar 17 02:31:52.673: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:31:52.673:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:31:52.673:  ---------------------------------------
SLOT 7/1: Mar 17 02:31:52.651: Pktcbl(gdb): Gate ID 42502 not found in gdb, pktc
bl_find_gate_ie.
SLOT 5/0: Mar 17 02:31:52.655: Pktcbl(gdb): Gate ID 42502 not found in gdb, pktc
bl_find_gate_ie.
ubr10k#
ubr10k#
*Mar 17 02:32:25.667: Pktcbl(mm): Received GATE SET message, tid=0x6
*Mar 17 02:32:25.667: --- Pktcbl(mm): Received GCP message ------
*Mar 17 02:32:25.667: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:32:25.667:     transaction id   : 0x6
*Mar 17 02:32:25.667:     gcp cmd          : 4 (GATE SET)
*Mar 17 02:32:25.667: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:32:25.667:     AM ID            : 0x1 (0/1)
*Mar 17 02:32:25.667: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:32:25.667:     Addr             : 21.21.2.10
*Mar 17 02:32:25.667: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:32:25.667:     GateID           : 42502 (0xA606)
*Mar 17 02:32:25.667: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 02:32:25.667:     flag             : 0x3
*Mar 17 02:32:25.667:     dscp             : 0xD8
*Mar 17 02:32:25.667:     dscp tos mask    : 0xF0
*Mar 17 02:32:25.667:     Timers  t1       : 0,  t2 : 0
*Mar 17 02:32:25.667:             t3       : 0,  t4 : 0
*Mar 17 02:32:25.667:     session class    : 0x0
*Mar 17 02:32:25.667: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 02:32:25.667:     envelope         : 0x7
*Mar 17 02:32:25.667:     service number   : 0x0
```

```
*Mar 17 02:32:25.667:     Authorized :
*Mar 17 02:32:25.667: Request Xmit Policy: 0x17F
*Mar 17 02:32:25.667: Grant size         : 232
*Mar 17 02:32:25.667: Grant Per Interval : 2
*Mar 17 02:32:25.667: Grant Interval     : 20000
*Mar 17 02:32:25.667: Tolerated Jitter   : 800
*Mar 17 02:32:25.667: Required Mask      : 0
*Mar 17 02:32:25.671: Forbidden Mask     : 0
*Mar 17 02:32:25.671: Aggr Rule Mask     : 0
*Mar 17 02:32:25.671:     Reserved   :
*Mar 17 02:32:25.671: Request Xmit Policy: 0x17F
*Mar 17 02:32:25.671: Grant size         : 232
*Mar 17 02:32:25.671: Grant Per Interval : 2
*Mar 17 02:32:25.671: Grant Interval     : 20000
*Mar 17 02:32:25.671: Tolerated Jitter   : 800
*Mar 17 02:32:25.671: Required Mask      : 0
*Ma
ubr10k#r 17 02:32:25.671: Forbidden Mask     : 0
*Mar 17 02:32:25.671: Aggr Rule Mask     : 0
*Mar 17 02:32:25.671: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:32:25.671:     protocol         : 1
*Mar 17 02:32:25.671:     dscp             : 0x0
*Mar 17 02:32:25.671:     dscp tos mask    : 0x0
*Mar 17 02:32:25.671:     src/port         : 9.9.1.95 0
*Mar 17 02:32:25.671:     dest/port        : 2.39.26.11 0
*Mar 17 02:32:25.671:     priority         : 64
*Mar 17 02:32:25.671: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:32:25.671:     protocol         : 1
*Mar 17 02:32:25.671:     dscp             : 0x0
*Mar 17 02:32:25.671:     dscp tos mask    : 0x0
*Mar 17 02:32:25.671:     src/port         : 9.9.1.43 0
*Mar 17 02:32:25.671:     dest/port        : 2.39.26.19 0
*Mar 17 02:32:25.671:     priority         : 64
*Mar 17 02:32:25.671: OPAQUE               : Object.[snum/stype/le
ubr10k#n 11/1/12]
*Mar 17 02:32:25.671:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:32:25.671: ------------------------------------
*Mar 17 02:32:25.671:  Backup gate IE [42502]
*Mar 17 02:32:25.671: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:32:25.671: Pktcbl(mm): Change profile 1 qos 1
*Mar 17 02:32:25.671: Pktcbl(gdb): IPC timer [id 42502] [10000 msec]
*Mar 17 02:32:25.671: Pktcbl(gdb): Started gate [id 42502] timer [type 8] [10000
 msec]
*Mar 17 02:32:25.671: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:32:25.671: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:32:25.671: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:32:25.671: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:32:25.671: Pktcbl(gdb): MM Reserved Profile
*Mar 17 02:32:25.671: Pktcbl(gdb): MM Committed Profile
*Mar 17 02:32:25.671: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:32:25.671: Classifier prototype: 1, src: 9.9.1.43, dest: 2.39.26.19,
src port: 0, dest port: 0
*Mar 17 02:32:25.707: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 425
02 rsp 1 state new(4) old(4)
*Mar 17 02:32:25.707: Pktcbl(gdb): Cancelled gate [id 42502] timer [type 8]
*Mar 17 02:32:25.707: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:32:25.707: Pktcbl(gdb): Started gate [id 42502] timer [type 3] [0 mse
c]
*Mar 17 02:32:25.707: Pktcbl(gdb): Cleanup saved gate IE info, gate(42502)
*Mar 17 02:32:25.707: PktCbl(d2r): extract id: gate=42502, resource=63
*Mar 17 02:32:25.707: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=42502 di
r=1 gie=265BD720
*Mar 17 02:32:25.707: Pktcbl(gdb): TOS Overwrite Gate=42502 DSCP=0xD0 mask=0xF
```

```
*Mar 17 02:32:25.707: PktCbl(d2r): extract id: gate=42502, resource=63
*Mar 17 02:32:25.707: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 42502 sfid 63
*Mar 17 02:32:25.707: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:32:25.707: Pktcbl(gdb): Found Cable7/1/0 for Gate=42502 21.21.2.10
*Mar 17 02:32:25.707: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:32:25.707: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:32:25.707: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:32:25.707: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:32:25.707: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:32:25.707: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:32:25.707:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:32:25.707: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:32:25.707:     transaction id   : 0x6
*Mar 17 02:32:25.707:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 02:32:25.707: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:32:25.707:     AM ID            : 0x1 (0/1)
*Mar 17 02:32:25.707: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:32:25.707:     Addr             : 21.21.2.10
*Mar 17 02:32:25.707: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:32:25.707:     GateID           : 42502 (0xA606)
*Mar 17 02:32:25.707: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:32:25.707:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:32:25.707:  ---------------------------------------
```

Related Commands

Command	Description
cable dynamic-qos trace	Enables the call trace functionality on the Cisco CMTS router for PacketCable or PacketCable Multimedia service subscribers.
debug cable dynamic-qos trace	Enables call trace debugging on the Cisco CMTS router for all the subscribers for whom call trace is configured.
show cable dynamic-qos trace	Displays the number of subscribers for whom call trace is enabled on the Cisco CMTS router.

debug cable dynamic-qos trace

To enable debugging of the call trace functionality on the Cisco CMTS router for all the subscribers for whom call trace is configured, use the **debug cable dynamic-qos trace** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug cable dynamic-qos trace [verbose]

no debug cable dynamic-qos trace

Syntax Description

verbose	(Optional) Provides detailed debugging information with the verbose output.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS Release 12.2(33)SCF	This command was introduced.

Usage Guidelines

The **debug cable dynamic-qos trace** command helps in debugging intermittent issues, which could be reproduced on any of the subscribers. Because the cable operator does not know which subscriber is likely to get intermittent issues, the operator cannot enable debugging for a particular subscriber. If debugging is enabled for all subscribers, this might impact the system performane due to huge volume of debugging output. With the PacketCable and PacketCable Multimedia (PCMM) call trace functionality, you can enable debugging only for the configured number of subscribers based on first-come first-served basis when a PacketCable call or PCMM session is attempted. The operator can periodically check if any of the subscribers is getting intermittent issues from the debug, or clear the logs and wait for the issue to surface on one of the subscribers having active trace.

We recommend that you use the **debug cable dynamic-qos trace** command during off-peak hours or under lower CPU utilization.

The following two debug commands enable most of the subscriber-based debug logs relevant to PacketCable, PCMM, and dynamic quality of service (DQoS) lite subscriber debugging:

- **debug cable dynamic-qos trace**
- **debug cable dynamic-qos subscriber**

Examples

The following example shows call trace debugging information for all subscribers on the Cisco uBR10012 router in Cisco IOS Release 12.2(33)SCF:

```
Router# debug cable dynamic-qos trace

CMTS dynqos trace debugging is on ubr10k#

*Mar 17 02:43:05.448: Pktcbl(mm): Received GATE SET message, tid=0x2
*Mar 17 02:43:05.448: --- Pktcbl(mm): Received GCP message ------
*Mar 17 02:43:05.448: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:43:05.448:     transaction id   : 0x2
*Mar 17 02:43:05.448:     gcp cmd          : 4 (GATE SET)
*Mar 17 02:43:05.448: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:43:05.448:     AM ID            : 0x1 (0/1)
*Mar 17 02:43:05.448: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:43:05.448:     Addr             : 21.21.2.10
*Mar 17 02:43:05.448: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:43:05.452:     GateID           : 58886 (0xE606)
*Mar 17 02:43:05.452: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 02:43:05.452:     flag             : 0x3
*Mar 17 02:43:05.452:     dscp             : 0xD8
*Mar 17 02:43:05.452:     dscp tos mask    : 0xF0
*Mar 17 02:43:05.452:     Timers  t1       : 0,  t2 : 0
*Mar 17 02:43:05.452:             t3       : 0,  t4 : 0
*Mar 17 02:43:05.452:     session class    : 0x0
*Mar 17 02:43:05.452: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 02:43:05.452:     envelope         : 0x7
*Mar 17 02:43:05.452:     service number   : 0x0
*Mar 17 02:43:05.452:     Authorized :
*Mar 17 02:43:05.452: Request Xmit Policy: 0x17F
*Mar 17 02:43:05.452: Grant size         : 232
*Mar 17 02:43:05.452: Grant Per Interval : 2
*Mar 17 02:43:05.452: Grant Interval     : 20000
*Mar 17 02:43:05.452: Tolerated Jitter   : 800
*Mar 17 02:43:05.452: Required Mask      : 0
*Mar 17 02:43:05.452: Forbidden Mask     : 0
*Mar 17 02:43:05.452: Aggr Rule Mask     : 0
*Mar 17 02:43:05.452:     Reserved   :
*Mar 17 02:43:05.452: Request Xmit Policy: 0x17F
*Mar 17 02:43:05.452: Grant size         : 232
*Mar 17 02:43:05.452: Grant Per Interval : 2
*Mar 17 02:43:05.452: Grant Interval     : 20000
*Mar 17 02:43:05.452: Tolerated Jitter   : 800
*Mar 17 02:43:05.452: Required Mask      : 0
*Ma
ubr10k#r 17 02:43:05.452: Forbidden Mask     : 0
*Mar 17 02:43:05.452: Aggr Rule Mask     : 0
*Mar 17 02:43:05.452: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:43:05.452:     protocol         : 1
*Mar 17 02:43:05.452:     dscp             : 0x0
*Mar 17 02:43:05.452:     dscp tos mask    : 0x0
*Mar 17 02:43:05.452:     src/port         : 9.9.1.95 0
*Mar 17 02:43:05.452:     dest/port        : 2.39.26.11 0
*Mar 17 02:43:05.452:     priority         : 64
*Mar 17 02:43:05.452: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:43:05.452:     protocol         : 1
*Mar 17 02:43:05.452:     dscp             : 0x0
*Mar 17 02:43:05.452:     dscp tos mask    : 0x0
*Mar 17 02:43:05.452:     src/port         : 9.9.1.43 0
*Mar 17 02:43:05.452:     dest/port        : 2.39.26.19 0
*Mar 17 02:43:05.452:     priority         : 64
*Mar 17 02:43:05.452: OPAQUE               : Object.[snum/stype/le
ubr10k#n 11/1/12]
```

```
*Mar 17 02:43:05.452:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:43:05.452: ---------------------------------------
*Mar 17 02:43:05.452:  Backup gate IE [58886]
*Mar 17 02:43:05.452: Pktcbl(gdb): Found Cable7/1/0 for Gate=58886 21.21.2.10
*Mar 17 02:43:05.452: Pktcbl(mm): Change profile 1 qos 1
*Mar 17 02:43:05.452: Pktcbl(gdb): IPC timer [id 58886] [10000 msec]
*Mar 17 02:43:05.452: Pktcbl(gdb): Started gate [id 58886] timer [type 8] [10000
 msec]
*Mar 17 02:43:05.452: Pktcbl(gdb): Found Cable7/1/0 for Gate=58886 21.21.2.10
*Mar 17 02:43:05.452: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:43:05.452: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:43:05.452: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:43:05.452: Pktcbl(gdb): MM Reserved Profile
*Mar 17 02:43:05.452: Pktcbl(gdb): MM Committed Profile
*Mar 17 02:43:05.452: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:43:05.452: Classifier prototype: 1, src: 9.9.1.43, dest: 2.39.26.19,
src port: 0, dest port: 0
*Mar 17 02:43:05.480: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 588
86 rsp 1 state new(4) old(4)
*Mar 17 02:43:05.480: Pktcbl(gdb): Cancelled gate [id 58886] timer [type 8]
*Mar 17 02:43:05.480: Pktcbl(gdb): Found Cable7/1/0 for Gate=58886 21.21.2.10
*Mar 17 02:43:05.480: Pktcbl(gdb): Started gate [id 58886] timer [type 3] [0 mse
c]
*Mar 17 02:43:05.480: Pktcbl(gdb): Cleanup saved gate IE info, gate(58886)
*Mar 17 02:43:05.484: PktCbl(d2r): extract id: gate=58886, resource=64
*Mar 17 02:43:05.484: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=58886 di
r=1 gie=265BDB84
*Mar 17 02:43:05.484: Pktcbl(gdb): TOS Overwrite Gate=58886 DSCP=0xD0 mask=0xF
*Mar 17 02:43:05.484: PktCbl(d2r): extract id: gate=58886, resource=64
*Mar 17 02:43:05.484: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 58886 sfid 64
*Mar 17 02:43:05.484: Pktcbl(gdb): Found Cable7/1/0 for Gate=58886 21.21.2.10
*Mar 17 02:43:05.484: Pktcbl(gdb): Found Cable7/1/0 for Gate=58886 21.21.2.10
*Mar 17 02:43:05.484: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:43:05.484: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:43:05.484: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:43:05.484: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:43:05.484: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:43:05.484: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:43:05.484:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:43:05.484: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:43:05.484:     transaction id   : 0x2
*Mar 17 02:43:05.484:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 02:43:05.484: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:43:05.484:     AM ID            : 0x1 (0/1)
*Mar 17 02:43:05.484: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:43:05.484:     Addr             : 21.21.2.10
*Mar 17 02:43:05.484: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:43:05.484:     GateID           : 58886 (0xE606)
*Mar 17 02:43:05.484: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:43:05.484:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:43:05.484:  ---------------------------------------
```

The following example shows call trace debugging information for all subscribers when using the **debug cable dynamic-qos trace** command with the **verbose** keyword on the Cisco uBR10012 router in Cisco IOS Release 12.2(33)SCF:

```
Router# debug cable dynamic-qos trace verbose

CMTS dynqos trace verbose debugging is on ubr10k#

*Mar 17 02:45:47.180: Pktcbl(mm): Received GATE SET message, tid=0x3
*Mar 17 02:45:47.180: --- Pktcbl(mm): Received GCP message ------
*Mar 17 02:45:47.180: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:45:47.180:     transaction id   : 0x3
*Mar 17 02:45:47.180:     gcp cmd          : 4 (GATE SET)
*Mar 17 02:45:47.180: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:45:47.180:     AM ID            : 0x1 (0/1)
*Mar 17 02:45:47.180: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:45:47.180:     Addr             : 21.21.2.10
*Mar 17 02:45:47.180: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 02:45:47.180:     flag             : 0x3
*Mar 17 02:45:47.180:     dscp             : 0xD8
*Mar 17 02:45:47.180:     dscp tos mask    : 0xF0
*Mar 17 02:45:47.180:     Timers  t1       : 0,  t2 : 0
*Mar 17 02:45:47.180:             t3       : 0,  t4 : 0
*Mar 17 02:45:47.180:     session class    : 0x0
*Mar 17 02:45:47.180: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 02:45:47.180:     envelope         : 0x7
*Mar 17 02:45:47.180:     service number   : 0x0
*Mar 17 02:45:47.180:     Authorized :
*Mar 17 02:45:47.180: Request Xmit Policy: 0x17F
*Mar 17 02:45:47.180: Grant size         : 232
*Mar 17 02:45:47.180: Grant Per Interval : 1
*Mar 17 02:45:47.180: Grant Interval     : 20000
*Mar 17 02:45:47.180: Tolerated Jitter   : 800
*Mar 17 02:45:47.180: Required Mask      : 0
*Mar 17 02:45:47.180: Forbidden Mask     : 0
*Mar 17 02:45:47.180: Aggr Rule Mask     : 0
*Mar 17 02:45:47.180:     Reserved   :
*Mar 17 02:45:47.180: Request Xmit Policy: 0x17F
*Mar 17 02:45:47.180: Grant size         : 232
*Mar 17 02:45:47.180: Grant Per Interval : 1
*Mar 17 02:45:47.180: Grant Interval     : 20000
*Mar 17 02:45:47.180: Tolerated Jitter   : 800
*Mar 17 02:45:47.180: Required Mask      : 0
*Mar 17 02:45:47.180: Forbidden Mask     : 0
*Mar 17 02:45:47.180: Aggr Rule Mask     : 0
*Mar 17 02:45:47.180: CLASSIFIER           : Obj
ubr10k#ect.[snum/stype/len 6/1/24]
*Mar 17 02:45:47.180:     protocol         : 1
*Mar 17 02:45:47.180:     dscp             : 0x0
*Mar 17 02:45:47.180:     dscp tos mask    : 0x0
*Mar 17 02:45:47.180:     src/port         : 9.9.1.95 0
*Mar 17 02:45:47.180:     dest/port        : 2.39.26.11 0
*Mar 17 02:45:47.180:     priority         : 64
*Mar 17 02:45:47.180: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:45:47.180:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:45:47.180: ---------------------------------------
*Mar 17 02:45:47.180: Pktcbl(gdb): Allocating gate entry 0/4. Instance cnt: 1, g
ate_count is now 5
*Mar 17 02:45:47.180: Pktcbl(gdb): Updated subscriber IE [subs addr: 21.21.2.10,
 gate: 75270]
*Mar 17 02:45:47.180: Pktcbl(gdb): Created gate IE on Cable7/1/0, gateid = 75270
*Mar 17 02:45:47.184: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:45:47.184: Pk
ubr10k#tcbl(mm): Change profile 0 qos 0
```

```
*Mar 17 02:45:47.184: Pktcbl(gdb): IPC timer [id 75270] [10000 msec]
*Mar 17 02:45:47.184: Pktcbl(gdb): Started gate [id 75270] timer [type 8] [10000
 msec]
*Mar 17 02:45:47.184: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:45:47.184: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:45:47.184: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:45:47.184: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:45:47.184: Pktcbl(gdb): MM Reserved Profile
*Mar 17 02:45:47.184: Pktcbl(gdb): MM Committed Profile
*Mar 17 02:45:47.184: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:45:47.216: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 752
70 rsp 1 state new(4) old(2)
*Mar 17 02:45:47.220: Pktcbl(gdb): Cancelled gate [id 75270] timer [type 8]
*Mar 17 02:45:47.220: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:45:47.220: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:45:47.220: Pktcbl(gdb): Started gate [id 75270] timer [type 3] [0 mse
c]
*Mar 17 02:45:47.220: PktCbl(d2r): extract id: gate=75270, resource=65
*Mar 17 02:45:47.220: PktCbl(d2r): extract id: gate=75270, resource=65
*Mar 17 02:45:47.220: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=75270 di
r=1 gie=26DD56A4
*Mar 17 02:45:47.220: Pktcbl(gdb): TOS Overwrite Gate=75270 DSCP=0xD0 mask=0xF
*Mar 17 02:45:47.220: PktCbl(d2r): extract id: gate=75270, resource=65
*Mar 17 02:45:47.220: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 75270 sfid 65
*Mar 17 02:45:47.220: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:45:47.220: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:45:47.220: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:45:47.220: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:45:47.220: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:45:47.220: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:45:47.220: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:45:47.220: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:45:47.220:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:45:47.220: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:45:47.220:     transaction id   : 0x3
*Mar 17 02:45:47.220:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 02:45:47.220: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:45:47.220:     AM ID            : 0x1 (0/1)
*Mar 17 02:45:47.220: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:45:47.220:     Addr             : 21.21.2.10
*Mar 17 02:45:47.220: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:45:47.220:     GateID           : 75270 (0x12606)
*Mar 17 02:45:47.220: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:45:47.220:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:45:47.224:  ------------------------------------
SLOT 7/1: Mar 17 02:45:47.231: Pktcbl(gdb): Gate ID 75270 not found in gdb, pktc
bl_find_gate_ie.
SLOT 5/0: Mar 17 02:45:47.227: Pktcbl(gdb): Gate ID 75270 not found in gdb, pktc
bl_find_gate_ie.
ubr10k#
ubr10k#
*Mar 17 02:46:14.446: Pktcbl(mm): Received GATE SET message, tid=0x4
*Mar 17 02:46:14.446: --- Pktcbl(mm): Received GCP message ------
*Mar 17 02:46:14.446: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:46:14.446:     transaction id   : 0x4
*Mar 17 02:46:14.446:     gcp cmd          : 4 (GATE SET)
```

```
*Mar 17 02:46:14.446: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:46:14.446:     AM ID            : 0x1 (0/1)
*Mar 17 02:46:14.446: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:46:14.446:     Addr             : 21.21.2.10
*Mar 17 02:46:14.446: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:46:14.446:     GateID           : 75270 (0x12606)
*Mar 17 02:46:14.446: GATE SPEC            : Object.[snum/stype/len 5/1/16]
*Mar 17 02:46:14.446:     flag             : 0x3
*Mar 17 02:46:14.446:     dscp             : 0xD8
*Mar 17 02:46:14.446:     dscp tos mask    : 0xF0
*Mar 17 02:46:14.446:     Timers  t1       : 0,  t2 : 0
*Mar 17 02:46:14.446:             t3       : 0,  t4 : 0
*Mar 17 02:46:14.446:     session class    : 0x0
*Mar 17 02:46:14.446: TRAFFIC PROFILE      : Object.[snum/stype/len 7/6/56]
*Mar 17 02:46:14.446:     envelope         : 0x7
*Mar 17 02:46:14.446:     service number   : 0x0
*Mar 17 02:46:14.446:     Authorized :
*Mar 17 02:46:14.446: Request Xmit Policy: 0x17F
*Mar 17 02:46:14.446: Grant size         : 232
*Mar 17 02:46:14.446: Grant Per Interval : 2
*Mar 17 02:46:14.446: Grant Interval     : 20000
*Mar 17 02:46:14.446: Tolerated Jitter   : 800
*Mar 17 02:46:14.446: Required Mask      : 0
*Mar 17 02:46:14.446: Forbidden Mask     : 0
*Mar 17 02:46:14.446: Aggr Rule Mask     : 0
*Mar 17 02:46:14.446:     Reserved   :
*Mar 17 02:46:14.446: Request Xmit Policy: 0x17F
*Mar 17 02:46:14.446: Grant size         : 232
*Mar 17 02:46:14.446: Grant Per Interval : 2
*Mar 17 02:46:14.446: Grant Interval     : 20000
*Mar 17 02:46:14.446: Tolerated Jitter   : 800
*Mar 17 02:46:14.446: Required Mask      : 0
*M
ubr10k#ar 17 02:46:14.446: Forbidden Mask     : 0
*Mar 17 02:46:14.446: Aggr Rule Mask     : 0
*Mar 17 02:46:14.446: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:46:14.446:     protocol         : 1
*Mar 17 02:46:14.446:     dscp             : 0x0
*Mar 17 02:46:14.446:     dscp tos mask    : 0x0
*Mar 17 02:46:14.446:     src/port         : 9.9.1.95 0
*Mar 17 02:46:14.446:     dest/port        : 2.39.26.11 0
*Mar 17 02:46:14.446:     priority         : 64
*Mar 17 02:46:14.446: CLASSIFIER           : Object.[snum/stype/len 6/1/24]
*Mar 17 02:46:14.446:     protocol         : 1
*Mar 17 02:46:14.446:     dscp             : 0x0
*Mar 17 02:46:14.446:     dscp tos mask    : 0x0
*Mar 17 02:46:14.446:     src/port         : 9.9.1.43 0
*Mar 17 02:46:14.446:     dest/port        : 2.39.26.19 0
*Mar 17 02:46:14.446:     priority         : 64
*Mar 17 02:46:14.446: OPAQUE               : Object.[snum/stype/l
ubr10k#en 11/1/12]
*Mar 17 02:46:14.446:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:46:14.446: ------------------------------------
*Mar 17 02:46:14.450:  Backup gate IE [75270]
*Mar 17 02:46:14.450: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:46:14.450: Pktcbl(mm): Change profile 1 qos 1
*Mar 17 02:46:14.450: Pktcbl(gdb): IPC timer [id 75270] [10000 msec]
*Mar 17 02:46:14.450: Pktcbl(gdb): Started gate [id 75270] timer [type 8] [10000
 msec]
*Mar 17 02:46:14.450: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:46:14.450: Pktcbl(gdb): MM gate spec: t1:200, t2:0, t3:0, t4:0
*Mar 17 02:46:14.450: Pktcbl(gdb): MM traffic profile type: 6
*Mar 17 02:46:14.450: Pktcbl(gdb): MM Authorized Profile
*Mar 17 02:46:14.450: Pktcbl(gdb): MM Reserved Profile
```

```
*Mar 17 02:46:14.450: Pktcbl(gdb): MM Committed Profile
*Mar 17 02:46:14.450: Classifier prototype: 1, src: 9.9.1.95, dest: 2.39.26.11,
src port: 0, dest port: 0
*Mar 17 02:46:14.450: Classifier prototype: 1, src: 9.9.1.43, dest: 2.39.26.19,
src port: 0, dest port: 0
*Mar 17 02:46:14.482: Pktcbl(mm): Received gate-set IPC RSP from LC for gate 752
70 rsp 1 state new(4) old(4)
*Mar 17 02:46:14.482: Pktcbl(gdb): Cancelled gate [id 75270] timer [type 8]
*Mar 17 02:46:14.482: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:46:14.482: Pktcbl(gdb): Started gate [id 75270] timer [type 3] [0 mse
c]
*Mar 17 02:46:14.482: Pktcbl(gdb): Cleanup saved gate IE info, gate(75270)
*Mar 17 02:46:14.482: PktCbl(d2r): extract id: gate=75270, resource=65
*Mar 17 02:46:14.482: Pktcbl(gdb): TOS Overwrite gate spec info,gate_id=75270 di
r=1 gie=26DD56A4
*Mar 17 02:46:14.482: Pktcbl(gdb): TOS Overwrite Gate=75270 DSCP=0xD0 mask=0xF
*Mar 17 02:46:14.482: PktCbl(d2r): extract id: gate=75270, resource=65
*Mar 17 02:46:14.482: PktCbl(mm-r2d): DSA-ACK notification received on RP, gatei
d 75270 sfid 65
*Mar 17 02:46:14.482: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:46:14.482: Pktcbl(gdb): Found Cable7/1/0 for Gate=75270 21.21.2.10
*Mar 17 02:46:14.482: Pktcbl(mm): Building GCP message, added obj TRANSACTION ID
      ; len:8 padding:0
*Mar 17 02:46:14.482: Pktcbl(mm): Building GCP message, added obj AM ID
      ; len:8 padding:0
*Mar 17 02:46:14.482: Pktcbl(mm): Building GCP message, added obj SUBSCRIBER ID
      ; len:8 padding:0
*Mar 17 02:46:14.482: Pktcbl(mm): Building GCP message, added obj GATE ID
      ; len:8 padding:0
*Mar 17 02:46:14.482: Pktcbl(mm): Building GCP message, added obj OPAQUE
      ; len:12 padding:0
*Mar 17 02:46:14.482: Pktcbl(mm): Built GCP message, GATE SET ACK        , lengt
h: 44, copsLen 72
*Mar 17 02:46:14.482:  --- Pktcbl: Sending GCP message ------
*Mar 17 02:46:14.482: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
*Mar 17 02:46:14.482:     transaction id   : 0x4
*Mar 17 02:46:14.482:     gcp cmd          : 5 (GATE SET ACK)
*Mar 17 02:46:14.482: AM ID                : Object.[snum/stype/len 2/1/8]
*Mar 17 02:46:14.482:     AM ID            : 0x1 (0/1)
*Mar 17 02:46:14.482: SUBSCRIBER ID        : Object.[snum/stype/len 3/1/8]
*Mar 17 02:46:14.482:     Addr             : 21.21.2.10
*Mar 17 02:46:14.482: GATE ID              : Object.[snum/stype/len 4/1/8]
*Mar 17 02:46:14.482:     GateID           : 75270 (0x12606)
*Mar 17 02:46:14.482: OPAQUE               : Object.[snum/stype/len 11/1/12]
*Mar 17 02:46:14.482:     data             : [31 32 33 34 00 00 00 00 ]
*Mar 17 02:46:14.482:  ---------------------------------------
```

Related Commands

Command	Description
cable dynamic-qos trace	Enables the call trace functionality on the Cisco CMTS router for PacketCable or PCMM service subscribers.
debug cable dynamic-qos subscriber	Enables debugging of the call trace functionality on the Cisco CMTS router for a particular subscriber.
show cable dynamic-qos trace	Displays the number of subscribers for whom call trace is enabled on the Cisco CMTS router.

debug cable dynamic-secret

To display debugging messages for the Dynamic Shared Secret feature, use the **debug cable dynamic-secret** command in privileged EXEC mode. To stop the display of debugging messages, use the **no** form of this command.

debug cable dynamic-secret

no debug cable dynamic-secret

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(15)BC1	This command was introduced for Cisco uBR7100 series, Cisco uBR7200 series, and Cisco uBR10012 routers.

Usage Guidelines

Tip Because this command can produce a large volume of debug information, use this command only when you have also enabled debugging for a particular interface or MAC address, using the **debug cable interface** and **debug cable mac-address** commands, respectively.

Examples

The following example shows how to enable debugging output using the **debug cable dynamic-secret** command:

```
Router# debug cable dynamic-secret

CMTS dynamic-secret debugging is on

Router#
```

The following example shows typical output for a cable modem that is reset and reregisters while the Dynamic Shared Secret feature is enabled:

```
02:15:59: Closing file for modem 00c0.2345.6789: New modem state from resetting to offline
02:16:10: Found file name in BOOT_FILE option
02:16:10: Parsing IP address from 10.10.35.200, len 10..  Got 10.10.35.200
Setting 10 bytes to PAD

02:16:10: Config file for 00c0.2345.6789 set to 10.8.35.200:/cm/UBsgvca698. Was
10.10.35.200:/cm/config.cm
```

```
02:16:10: Discarding contents of /cm/config.cm @ 10.10.35.200
02:16:10: FIle /cm/config.cm from 10.10.35.200 read successfully
02:16:10: Found file name in BOOT_FILE option
02:16:10: Parsing IP address from 10.10.35.200, len 10..  Got 10.10.35.200
Setting 10 bytes to PAD

02:16:10: Config file for 00c0.2345.6789 set to 10.10.35.200:/cm/wrkldJKDHS. Was
10.10.35.200:/cm/config.cm
02:16:15: Registration request from 00c0.2345.6789, SID 1 on Cable3/0/U0
02:16:15: TLV-Block Bytes:
02:16:15: 0x0000: 03 01 01 04 1F 01 01 01 02 04 00 3D 09 00 03 04
02:16:15: 0x0010: 00 01 F4 00 04 01 07 05 04 00 01 86 A0 06 02 00
02:16:15: 0x0020: 00 07 01 00 12 01 0A 06 10 97 C2 BA 04 6A 76 BD
02:16:15: 0x0030: AC 40 FA E6 EB CD 49 E9 54 07 10 89 86 20 E2 73
02:16:15: 0x0040: 16 EE 01 0B 24 4A 65 F8 55 93 90 0C 04 03 12 01
02:16:15: 0x0050: 09 08 03 00 06 28 05 1E 02 01 01 03 01 01 04 01
02:16:15: 0x0060: 01 06 01 01 07 01 00 08 01 04 0A 01 01 0B 01 08
02:16:15: 0x0070: 0C 01 01 01 01 01
02:16:15: Found Network Access TLV
02:16:15: Ntw Access Control : 1
02:16:15: Found Class Of Service TLV Block
02:16:15:       Class Id : 1
02:16:15:       Maximum Downstream Rate : 4000000
02:16:15:       Maximum Upstream Rate : 128000
02:16:15:       Upstream Traffic Priority : 7
02:16:15:       Guaranteed Minimum Upstream Rate : 100000
02:16:15:       Maximum Upstream Transmit Burst : 0
02:16:15:       Privacy Enable : 0
02:16:15: Found Max CPEs TLV
02:16:15: Maximum Number Of CPEs : 10
02:16:15: Found CM-MIC TLV
02:16:15: CM MIC:
02:16:15: 0x0000: 97 C2 BA 04 6A 76 BD AC 40 FA E6 EB CD 49 E9 54
02:16:15: Found CMTS-MIC TLV
02:16:15: CMTS MIC:
02:16:15: 0x0000: 89 86 20 E2 73 16 EE 01 0B 24 4A 65 F8 55 93 90
02:16:15: Found CM IP Address TLV
02:16:15: Modem IP Address : 10.18.1.9
02:16:15: Vendor Id:
02:16:15: 0x0000: 00 06 28
02:16:15: Found Modem Capabilities TLV
02:16:15:       DOCSIS Version : 1
02:16:15:       Fragmentation Support : 1
02:16:15:       Payload Header Suppresion Support : 1
02:16:15:       Privacy Support : 1
02:16:15:       Downstream SAID Support : 0
02:16:15:       Upstream SID Support : 4
02:16:15:       Tx Equalizer Taps Per Symbol : 1
02:16:15:       Tx Equalizer Taps Support : 8
02:16:15:       DCC Support : 1
02:16:15:       Concatenation Support : 1
02:16:15: Computing CMTS-MIC using Dynamic Secret to validate REG-REQ data.
02:16:15: CMTS_MIC(rfc2104) failed text + key
02:16:15: CMTS_MIC(rfc2104) passed
02:16:15: Performing admission control check
02:16:15: Mapping Primary DOCSIS 1.0 CoS block into service flows
02:16:15: Added Modem Capabilities TLV:
02:16:15: 0x0000: 05 1E 02 01 01 03 01 01 04 01 01 06 01 01 07 01
02:16:15: 0x0010: 00 08 01 04 0A 01 01 0B 01 08 0C 01 01 01 01 01
02:16:15: ClassId:1 assigned SID:1
02:16:15: Added Service Class Data TLV:
02:16:15: 0x0000: 01 07 01 01 01 02 02 00 01
02:16:15: REG-RSP Status : ok (0)
02:16:15: Closing file for modem 00c0.2345.6789: New modem state from init(o) to online
```

```
02:16:15: Registration Response Transmitted

Router#
```

The following example shows the typical messages that are shown when you use the **clear cable modem lock** command to clear the lock on one or more cable modems that have failed the Dynamic Shared Secret checks and were locked into restrictive quality of service (QoS) configurations:

```
Router# clear cable modem all lock

01:46:37: Closing file for modem 00c0.2854.73f5: New modem state from online to offline
01:46:37: Closing file for modem 00c0.734e.b4aa: New modem state from online(pt) to
offline
01:46:37: Closing file for modem 00c0.7366.17cb: New modem state from init(d) to offline
01:46:37: Closing file for modem 00c0.80bc.22b5: New modem state from online to offline

Router#
```

The following debug message is displayed when the CMTS has verified the CMTS MIC in a Data-over-Cable Service Interface Specifications (DOCSIS) configuration file against the shared secret that is configured on a cable interface:

```
Validated against Shared secret
```

The CMTS displays the following debug message when the CMTS has verified the CMTS MIC in a DOCSIS configuration file against one of the 16 possible secondary shared secrets that are configured on a cable interface:

```
Validated against secondary secret 11
```

The CMTS displays the following debug message when the CMTS obtains a new DOCSIS configuration file from the TFTP server. The CMTS keeps this file in its internal cache for 30 seconds, so that it can be used for other cable modems that might request it.

```
Creating new cache for config-file.cm @ 10.10.10.21
```

The CMTS displays the following debug messages when it receives a registration request or TFTP request for a DOCSIS configuration file from a device that is not a cable modem. This can indicate that a user is trying to manually download a DOCSIS configuration file so that it can be decoded and modified.

```
MAC address 0102.0304.0506 not of a CM
MAC Address 0102.0304.0506, IP address 10.10.10.13 is not a modem
```

The CMTS displays the following debug message when it receives a TFTP request for an unknown TFTP server. This can indicate that a user is trying to download a DOCSIS configuration file from a local TFTP server.

```
TFTP server unknown for CM 0102.0304.0506
```

The CMTS displays the following debug message when it receives a TFTP request for a non-cable interface. This can indicate that a user is trying to spoof an IP or Ethernet MAC address, and is trying to access a TFTP server through the Internet or a local network.

```
TFTP request from a non-cable interface GE1/0/0 ignored
```

The CMTS displays the following debug messages when it receives a TFTP request for a DOCSIS configuration file that is not available on the TFTP server:

```
16:11:33: Could not locate bootfile name for 0102.0304.0506
16:11:33: Cannot locate config-file.cm requested by CM 0102.0304.0506
16:11:33: Closing file for modem 0102.0304.0506: New modem state from init(rc) to offline
```

Related Commands

Command	Description
cable dynamic-secret	Enables the Dynamic Shared Secret feature, so that DOCSIS configuration files are verified with a Message Integrity Check (MIC) that has been created with a dynamically generated shared secret.
cable shared-secondary-secret	Configures one or more secondary shared secret keys that CMs can use to successfully process the DOCSIS configuration file and register with the CMTS.
cable shared-secret	Configures an authentication shared secret key that CMs must use to successfully process the DOCSIS configuration file and register with the CMTS.
cable tftp-enforce	Requires that all CMs on a cable interface attempt to download a DOCSIS configuration file using the Trivial File Transfer Protocol (TFTP) through the cable interface before being allowed to register and come online.
show cable modem rogue	Displays a list of cable modems that have been marked, locked, or rejected because they failed the dynamic shared-secret authentication checks.
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-address	Enables debugging output for the cable modems that match the specified hardware (MAC) address or range of addresses.

debug cable dynsrv

To display information about DOCSIS 1.1 dynamic service flow messages, use the **debug cable dynsrv** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable dynsrv

no debug cable dynsrv

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR and 12.1(1a)T1	This command was introduced for DOCSIS 1.0+ operation.
12.1(4)CX	This command was enhanced for DOCSIS 1.1 operation.
12.2(4)BC1	Support was added to the Release 12.2 BC train.

Usage Guidelines

This command activates dynamic service flow debugging, which displays a debugging message whenever one of the following DOCSIS 1.1 dynamic service messages is processed:

- Dynamic Service Add (DSA)—This message is used to create a new service flow.
- Dynamic Service Change (DSC)—This message is used to change the attributes of an existing service flow.
- Dynamic Service Deletion (DSD)—This message is used to delete an existing service flow.

Each type of dynamic service message includes a request (DSA-REQ, DSC-REQ, DSD-REQ) and a response (DSA-RSP, DSC-RSP, DSD-RSP).

Do not use this command when you have a large number of active CMs on your network, because it could generate a huge amount of output to the console port.

Note Debugging must be enabled for one or more cable interfaces, using the **debug cable interface** command, before the **debug cable dynsrv** command will display any output.

Tip If using the **debug cable dynsrv** command, also use the **debug cable mac-address** and the **debug cable tlvs** command to enable full DOCSIS MAC-layer debugging.

Examples

The following example shows typical output displayed by the **debug cable dynsrv** command:

```
Router# debug cable interface c3/0
Router# debug cable dynsrv
CMTS dynsrv debugging is on
```

```
Router#
     *May 5 05:15:36.531: DSA-REQ-RECD: OrgMac->0050.734e.b5b1 OrgId->52
     *May 5 05:15:36.531: DSx-STATE-CREATED: OrgMac->0050.734e.b5b1 OrgId->52
     *May 5 05:15:36.531: DSA-REQ TLV Information:
     *May 5 05:15:36.531: Type Subtype Subtype Length Value
     *May 5 05:15:36.531: 24 10
     *May 5 05:15:36.531: 19 2 89
     *May 5 05:15:36.531: 20 4 20000
     *May 5 05:15:36.531: 80 69
     *May 5 05:15:36.531: DSA-REQ: Requested QoS Parameter Information:
     *May 5 05:15:36.531: Srv Flow Ref: 0 Grant Size: 89 Grant Intvl: 20000
     *May 5 05:15:36.531: Requested QoS parameters match QoS Profile:3 (G729)
     *May 5 05:15:36.531: DSA-REQ-SID-ASSIGNED: CM 0050.734e.b5b1 SID 11
     *May 5 05:15:36.531: DSA-RSP-SEND: OrgMac->0050.734e.b5b1 OrgId->52
     *May 5 05:15:36.531: DSA-RSP msg TLVs
     *May 5 05:15:36.531: Type:Length:Value
     *May 5 05:15:36.531: US QoS Encodings 24:8
     *May 5 05:15:36.531: SID 3:2:11
     *May 5 05:15:36.531: Service Flow Reference 1:2:0
     *May 5 05:15:36.531: DSA-RSP hex dump:
     *May 5 05:15:36.531: 0x0000: C2 00 00 26 00 00 00 50 73 4E B5 B1 00 10 0B AF
     *May 5 05:15:36.531: 0x0010: BC 54 00 14 00 00 03 01 10 00 00 34 00 18 08 03
     *May 5 05:15:36.531: 0x0020: 02 00 0B 01 02 00 00 00
```

Related Commands

Command	Description
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for MAC-layer information for a specific CM.
debug cable tlvs	Enables debugging for the Type/Length/Value encodings (TLVs) parsed by the DOCSIS 1.1 TLV parser/encoder.

debug cable encap

To enable debugging of encapsulated packets that are transmitted over the cable interface, use the **debug cable encap** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable encap

no debug cable encap

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)BC3	This command was introduced.

Usage Guidelines

The **debug cable encap** command displays debugging messages about the following types of encapsulated packets:

- Debug messages prefixed with ENCAPSTR display information about CEF adjacency operations.
- Debug messages prefixed with VENCAP display information about MAC-layer encapsulation for switched packets (such as PPPoE and IP packets).

The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. In particular, avoid using the **debug cable encap** command on an interface with a significant number of CMs, because the resulting volume of debug output could impact system performance. Cisco recommends that when you use the **debug cable encap** command, you limit its output to a particular interface or CM, using the **debug cable interface** or **debug cable mac-address** commands.

Examples

The following shows typical output for the **debug cable encap** command. Messages are shown for both CEF adjacency and MAC-layer encapsulation events:

```
Router# debug cable encap
CMTS encap debugging is on

Router# clear adjacency

ENCAPSTR hwidb Cable3/1 idb Cable3/1.1 mac 0020.4072.7418 ipaddr 31.0.0.2 linktype 7
ENCAPSTR 0020.4072.7418 SID 2 STR 00000000800200204072741800D0582770540800

Router# ping 31.0.0.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 31.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/8 ms
```

```
Router#

VENCAP hwidb Cable3/0 idb Cable3/1.1 mac 0020.4072.7418 ip 31.0.0.2 linktype 7
VENCAP  0020.4072.7418 SID 2 STR 00000076800200204072741800D0582770540800

VENCAP hwidb Cable3/0 idb Cable3/1.1 mac 0020.4072.7418 ip 31.0.0.2 linktype 7
VENCAP  0020.4072.7418 SID 2 STR 00000076800200204072741800D0582770540800

VENCAP hwidb Cable3/0 idb Cable3/1.1 mac 0020.4072.7418 ip 31.0.0.2 linktype 7
VENCAP  0020.4072.7418 SID 2 STR 00000076800200204072741800D0582770540800

VENCAP hwidb Cable3/0 idb Cable3/1.1 mac 0020.4072.7418 ip 31.0.0.2 linktype 7
VENCAP  0020.4072.7418 SID 2 STR 00000076800200204072741800D0582770540800

VENCAP hwidb Cable3/0 idb Cable3/1.1 mac 0020.4072.7418 ip 31.0.0.2 linktype 7
VENCAP  0020.4072.7418 SID 2 STR 00000076800200204072741800D0582770540800
```

Related Commands

Command	Description
debug cable arp	Enables debugging of the Address Resolution Protocol on the cable interface.
debug cable dhcp	Enables debugging of the Dynamic Host Configuration Protocol on the cable interface.
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for a particular CM.

debug cable envm

To display information about the Cisco CMTS physical environment, including internal temperature, midplane voltages, fan performance, and power supply voltages, use the **debug cable env** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable envm

no debug cable envm

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command is used to debug the sensor circuitry used to measure internal temperature, midplane voltages, fan performance, and power supply voltages on the Cisco CMTS console.

Examples

The following example shows sample output displayed by the **debug cable envm** command

```
Router# debug cable envm
ENVM: ps id=0xFF0, v=0x2050, r=0xC0AB, pstype=1
ENVM: ps id=0x2FD0, v=0x2050, r=0x24201, pstype=27
NVM: Sensor 0: a2dref=131, a2dact=31, vref=12219, vact=1552 Alpha=8990, temp=27
```

Table 255 provides description for the output.

Table 255 Sample Output for the debug cable envm Command

Field	Description
ps id	Power supply raw voltage reading.
pstype	Power supply type determined from ps ID, v, and r. If the Cisco CMTS contains dual power supplies, the ID information for two types is usually printed.
Sensor	Sensor number.
a2dref	Analog-to-digital converter reference reading.
a2dact	Analog-to-digital converter actual (measured reading).
vref	Reference voltage.
vact	Actual voltage.
Alpha	Raw temperature reading.
temp	Temperature corresponding to Alpha.

Related Commands

Command	Description
show environment	Displays the temperature and voltage information for the chassis.

debug cable error

To display errors that occur in the cable MAC protocols, use the **debug cable err** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable error

no debug cable error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command displays unexpected DOCSIS MAC protocol messages. When the Cisco CMTS does not to expect to receive a specific MAC message, or if a packet contains errors in size or content, an error message and a hex dump of the packet are printed. Other miscellaneous error conditions may also show up in output.

Examples

The following shows typical output displayed by the **debug cable error** command.

```
Router# debug cable error
Router#
 00:19:22: Receive data error, status=0x0D04
 00:19:22: Receive data error:
 00:19:22: Prepended data:
        5C 1E 00 02 00 5F 58 5F 01 12 36 FF F2 00 00 00
 00:19:22: Packet data (size=36):
 00:19:22: 0x000: C2 00 00 1E 8E 07 00 E0 1E D7 CC 94 00 50 73 18
 00:19:22: 0x010: E9 F1 00 0C
 00:19:22:
```

debug cable flap

To display information about the operation of the CM flap list that is maintained for the cable interfaces, use the **debug cable flap** command in the Privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable flap

no debug cable flap

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
11.3(8)NA, 12.0(4)T	This command was introduced for Cisco uBR7200 series routers.
12.0(6)SC	Support for this command was added to the Release 12.0 SC.
12.1(5)EC	Support for this command was added for the Cisco uBR7100 series routers.
12.2(4)BC1	Support was added to the Release 12.2 BC.
12.2(11)BC2	Support for this command was added to the Release 12.2 BC.

Usage Guidelines

This command shows debugging messages about the operation of the CM flap list that is maintained for the cable interfaces.

Caution The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic.

Examples

The following example enables debugging for the CM flap list:

```
Router# debug cable flap

CMTS flap debugging is on

Router# show debug
CMTS:
  CMTS flap debugging is on
Router#
```

When debugging is turned on and a CM is added to the flap list the Cisco CMTS displays the following message:

```
Cable modem <<mac address>> added to flap-list
```

When debugging is turned on and a CM is removed from the flap list, the Cisco CMTS displays the following message:

```
Cable modem <<mac address>> removed from flap-list (<<number>> seconds after last flap)
```

When debugging is turned on and the Cisco CMTS checks the flap list for aged out CMs that need to be removed from the list, the CMTS displays the following message:

```
CMTS flap list aging check (interval = 120 min)
```

Related Commands

cable flap-list aging	Specifies the number of days to keep a CM in the flap-list table before aging it out of the table.
cable flap-list insertion-time	Sets the insertion time interval that determines whether a CM is placed in the flap list.
cable flap-list miss-threshold	Specifies miss threshold for recording a flap-list event.
cable flap-list power-adjust threshold	Specifies the power-adjust threshold for recording a CM flap-list event.
cable flap-list size	Specifies the maximum number of CMs that can be listed in the flap-list table.
clear cable flap-list	Clears all the entries in the flap-list table.
ping docsis	Sends a DOCSIS ping to a CM and increments the flap-list counters as appropriate.
show cable flap-list	Displays the current contents of the flap list.

debug cable fn

To enable debugging information for cable fiber nodes, use the **debug cable fn** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable fn

no debug cable fn

Syntax Description

This command has no keywords or arguments.

Command Default

No cable fiber node debug messages are enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(21)BC	This command was introduced for the Cisco uBR10012 router.

Usage Guidelines

The **debug cable fn** command is intended for use by Cisco technical support personnel.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use **debug** commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support personnel. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased debug command processing overhead will affect system use.

For general (non-wideband) information on CMTS debugging commands, see the *Cisco Broadband Cable Command Reference Guide*.

Examples

The following example shows how to enable debug messages for cable fiber nodes:

```
Router# debug cable fn

CMTS fiber node debugging is on

...
```

The following sample messages are displayed when RF channels 0 and 1 are removed from fiber node 1:

```
Router(config)# cable fiber 1
Router(config-fiber-node)# no downstream modular-cable 3/0/0 rf-channel 0 - 1
...
*Nov 7 17:42:27.903: fn_get_cdb: cdb slot 3(3) subslot 0(0)appl_no 0(0)
*Nov 7 17:42:27.903: cfg_cable_fn_cmd: fiber-node 1 status 0x01 cmd 3 sense 0
```

```
*Nov 7 17:42:27.903: no downstream FN 1, 3/0/0 port range 0 - 1
*Nov 7 17:42:27.903: no downstream FN 1, 3/0/0 bg_rfch 0000000000000004
*Nov 7 17:42:27.903: set FN 1 status old 0x01 to new 0x01
*Nov 7 17:42:27.903: fn_freq_unique: FN 1
*Nov 7 17:42:27.903: prim_rfch 35 slot 8 subslot 1 unit 0
*Nov 7 17:42:27.903: prim_rfch 35 freq 555000000 channel_id 203
*Nov 7 17:42:27.903: bg_rfch 2 slot 3 subslot 0 unit 2
*Nov 7 17:42:27.903: fn_get_cdb: cdb slot 3(3) subslot 0(0)appl_no 0(2)
*Nov 7 17:42:27.903: bg_rfch 2 freq 711000000 channel_id 26
*Nov 7 17:42:27.903: clr FN 1 status old 0x01 to new 0x01
*Nov 7 17:42:27.903: fn_channel_id_unique: FN 1
*Nov 7 17:42:27.903: prim_rfch 35 slot 8 subslot 1 unit 0
*Nov 7 17:42:27.903: prim_rfch 35 freq 203 channel_id 203
*Nov 7 17:42:27.903: bg_rfch 2 slot 3 subslot 0 unit 2
*Nov 7 17:42:27.903: fn_get_cdb: cdb slot 3(3) subslot 0(0)appl_no 0(2)
*Nov 7 17:42:27.903: bg_rfch 2 freq 711000000 channel_id 26
*Nov 7 17:42:27.903: clr FN 1 status old 0x01 to new 0x01
*Nov 7 17:42:27.903: fn_bundle_same: FN 1
*Nov 7 17:42:27.903: prim_rfch 35 slot 8 subslot 1 unit 0
*Nov 7 17:42:27.903: prim_rfch 35 nb_hwidb Cable8/1/0 bundle 1
*Nov 7 17:42:27.903: bg_rfch 2 slot 3 subslot 0 unit 2
*Nov 7 17:42:27.903: fn_get_cdb: cdb slot 3(3) subslot 0(0)appl_no 0(2)
*Nov 7 17:42:27.903: clr FN 1 status old 0x01 to new 0x01
*Nov 7 17:42:27.903: cfg_cable_fn_cmd: fiber-node 1 status 0x01
*Nov 7 17:42:35.267: FN 1 save context status 0x01(0x01)
*Nov 7 17:42:35.267: FN 1 status 0x01 saved status 0x01)
*Nov 7 17:42:35.267: fiber-node 1 update SG status = 0x01
*Nov 7 17:42:35.267: DS-SG: Updating Association Table from DS-SG. Current: 0x1 New: 0x0
*Nov 7 17:42:35.267: DS-SG: Updating the Wideband Channel Mask: 0x0
```

Related Commands

Command	Description
debug c10k-jacket	Enables debugging information for the Wideband SIP.
debug cable wbcmts	Enables debugging information for the wideband CMTS.
debug hw-module bay	Enables debugging information for a Wideband SPA.

debug cable freqhop

To display debug messages for frequency hopping, use the **debug cable freqhop** command in privileged EXEC mode. Use the **no** form of this command to disable debugging output.

debug cable freqhop

no debug cable freqhop

Syntax Description

This command has no arguments or keywords.

Command History

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

Examples

The following example shows how to display debug messages for frequency hopping:

```
router# debug cable interface c6/0
router# debug cable freqhop
CMTS freqhop debugging is on
router#
Sep 27 10:36:41.202: Cable5/0 U0: last hop delta: 1
Sep 27 10:36:42.202: Cable5/0 U0: last hop delta: 2
```

debug cable hw-spectrum

To display debug messages for spectrum management (frequency agility), use the **debug cable hw-spectrum** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable hw-spectrum

no debug cable hw-spectrum

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3 NA	This command was introduced as **debug cable specmgmt**.
12.0(4)XI	This command was enhanced and renamed to **debug cable hw-spectrum**.

Examples

The following shows how to enable **debug cable hw-spectrum** debugging.

```
Router# debug cable hw-spectrum
CMTS spectrum analyzer debugging is on
Router#
```

Related Commands

Command	Description
debug cable specmgmt	Displays debugging output for the CMTS spectrum analyzer process.

debug cable interface

To display debug messages for a specific cable interface, or for traffic related to a specific MAC address or Service ID (SID) on that interface, use the **debug cable interface** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable interface cable {*slot/port* | *slot/subslot/port*} [**mac-address** *address* [*mask*] | **sid** *number* [**track**] | [**verbose**]]

no debug cable interface cable {*slot/port* | *slot/subslot/port*} [**mac-address** *address* [*mask*] | **sid** *number* [**track**] | [**verbose**]]

Syntax Description

slot/port	Identifies the cable interface and downstream port on the Cisco uBR7100 series and Cisco uBR7200 series routers. On the Cisco uBR7100 series router, the only valid value is **1/0**. On the Cisco uBR7200 series router, *slot* can range from 3 to 6, and *port* can be 0 or 1, depending on the cable interface.
slot/subslot/port	Identifies the cable interface on the Cisco uBR10012 router. The following are the valid values: • *slot* = 5 to 8 • *subslot* = 0 or 1 • *port* = 0 to 4 (depending on the cable interface)
mac-address *address mask*	(Optional) Specifies that debugging is to be done only on traffic related to the specified MAC *address*. An optional *mask* can be specified to indicate a range of MAC addresses. The *mask* is ANDed with the *address* to determine which bits of the address must match to be included in the debugging display.
sid *number*	(Optional) Specifies that debugging is to be done only on traffic related to the specified SID. The valid range is from 1 to 8191.
track	(Optional) Enables SID tracking on a cable interface line card.
verbose	(Optional) Displays detailed debug information.

Defaults

Debugging for the cable interfaces is not enabled, which means most of the other **debug cable** commands will not display any output, even when debugging is enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(6)T	This command was introduced.
12.2(33)SCC	This command was integrated into Cisco IOS Release 12.2(33)SCC. A new keyword, **track**, was added to enable SID tracking on a cable interface line card.

Usage Guidelines

The **debug cable interface** command must be used to enable debugging on a cable interface before other **debug** commands can be used on that interface. The **mac-address** and **sid** options can be used to restrict the debug output to only those messages that are related to a specific MAC address or SID, so that the volume of debug messages does not affect system performance.

Examples

The following example shows how to enable debugging on the cable interface in slot 6:

```
Router# debug cable interface c6/0
Router# show debug
CMTS specific:
Debugging is on for Cable6/0
Router#
```

The following shows how to enable verbose debugging on the cable interface in slot 6:

```
Router# debug cable interface c6/0 verbose
Router# show debug
CMTS specific:
Debugging is on for Cable6/0 (verbose)
Router#
```

The following example shows how to enable debugging on the cable interface in slot 6 for all traffic coming from CMs and other devices with MAC addresses that match the address range 0010.0000.0000 through 0010.00FF.FFFF (0010.00xx.xxxx):

```
Router# debug cable interface c6/0 mac-address 0010.0000.0000 FFFF.FF00.0000
Router# show debug
CMTS specific:
Debugging is on for Cable6/0, Address 0010.0000.0000, Mask ffff.ff00.0000
Router#
```

Related Commands

Command	Description
debug cable dynsrv	Displays debugging information about DOCSIS 1.1 dynamic service flow messages.
debug cable mac-address	Enables debugging on traffic from CMs with the specific MAC address or within the specific MAC address range.

debug cable ipv6

To enable debugging of IPv6 transactions on a cable interface on a Cisco CMTS router, use the **debug cable ipv6** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable ipv6 [**db** | **dhcp** | **ha** | **lq** | **nd** | **source-verify**]

no debug cable ipv6

Syntax Description

db	(Optional) Displays messages associated with host database transactions.
dhcp	(Optional) Displays messages associated with Dynamic Host Control Protocol for IPv6 (DHCPv6) transactions.
ha	(Optional) Displays messages associated with high availability (HA) IPv6 transactions.
lq	(Optional) Displays messages associated with leasequery (LQ) transactions.
nd	(Optional) Displays messages associated with Neighbor Discovery (ND) transactions.
source-verify	(Optional) Displays messages associated with source verification transactions.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCA	This command was introduced.
12.2(33)SCF1	This command was modified. The **lq** keyword was added to display leasequery transactions.

Usage Guidelines

The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. In particular, avoid using the **debug cable ipv6** command on an interface with a significant number of CMs, because the resulting volume of debug output could impact system performance. We recommend that when you use the **debug cable ipv6** command, you limit its output to a particular interface or cable modem (CM), using the **debug cable interface** or **debug cable mac-address** commands.

Related Commands

Command	Description
debug cable arp	Enables debugging of the Address Resolution Protocol on the cable interface.
debug cable encap	Enables debugging of encapsulated PPPoE packets that are transmitted over the cable interface.
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging on a particular CM.

debug cable keyman

To activate debugging of TEK and KEK BPI key management, use the **debug cable keyman** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable keyman

no debug cable keyman

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of the TEK and KEK baseline privacy key activity. When this command is activated, all activity related to KEK and TEK keys is displayed on the Cisco CMTS console. This command is used to display encryption key management debugging output.

Note This command is supported only on images that support BPI or BPI+ encryption.

Examples

The following shows typical output from the **debug cable keyman** command.

```
Router# debug cable keyman
Router#

 Dec 11 23:16:19.139: CMTS Received AUTH REQ.
 Dec 11 23:16:19.139: Created a new CM key for 0010.7b43.aa2f.
 Dec 11 23:16:19.139: CMTS generated AUTH_KEY.
 Dec 11 23:16:19.139: Input : 23FD16A701FD2091
 Public Key :
 Dec 11 23:16:19.139: 30 68 02 61 00 C1 26 6F 53 3E CE 17 AB 18 84 C5
 Dec 11 23:16:19.139: 63 B3 A2 DA 66 29 96 13 23 B8 A3 C8 AF B7 CF C8
 Dec 11 23:16:19.139: 54 6B 16 14 E2 9B 12 0B 34 79 51 DA 18 AB DE 8C
 Dec 11 23:16:19.139: 65 8F 0B 8A AB 25 3B 88 F1 6D 53 5F 64 C3 3E 50
 Dec 11 23:16:19.139: 81 57 AA C5 8F CE 4F 3C A8 96 2F 60 0F F6 30 E6
 Dec 11 23:16:19.139: 91 61 29 42 E1 C2 96 0F CB 10 EF F9 0D 6F 45 76
 Dec 11 23:16:19.139: 1D 17 FD 26 6D 02 03 01 00 01
 Dec 11 23:16:19.139:
 Dec 11 23:16:19.139: RSA publlic Key subject:
 Dec 11 23:16:19.143: 30 7C 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05
 Dec 11 23:16:19.143: 00 03 6B 00 30 68 02 61 00 C1 26 6F 53 3E CE 17
 Dec 11 23:16:19.143: AB 18 84 C5 63 B3 A2 DA 66 29 96 13 23 B8 A3 C8
 Dec 11 23:16:19.143: AF B7 CF C8 54 6B 16 14 E2 9B 12 0B 34 79 51 DA
 Dec 11 23:16:19.143: 18 AB DE 8C 65 8F 0B 8A AB 25 3B 88 F1 6D 53 5F
 Dec 11 23:16:19.143: 64 C3 3E 50 81 57 AA C5 8F CE 4F 3C A8 96 2F 60
 Dec 11 23:16:19.143: 0F F6 30 E6 91 61 29 42 E1 C2 96 0F CB 10 EF F9
```

```
Dec 11 23:16:19.143: 0D 6F 45 76 1D 17 FD 26 6D 02 03 01 00 01
Dec 11 23:16:19.155: RSA encryption result = 0
Dec 11 23:16:19.155: Output :
Dec 11 23:16:19.155: 88 5F 67 22 86 68 2B 1D A6 F4 E9 62 43 58 1A C8
Dec 11 23:16:19.155: 49 97 7E 81 EE EF B0 DD C4 42 30 FD 24 B0 54 2E
Dec 11 23:16:19.155: 01 CC 84 53 BD 71 50 9D B3 82 4D 7B 49 42 E1 F0
Dec 11 23:16:19.155: 2D 67 3D 46 CB 27 4D 60 16 00 4D EE E1 F3 FD 1D
Dec 11 23:16:19.155: 9C E6 03 3C 77 C8 3A 44 B9 FA 34 2E 44 1B 69 F4
Dec 11 23:16:19.155: AA 68 BF BB A5 43 9B F7 85 82 ED 39 45 02 92 56
Dec 11 23:16:19.155: CMTS sent AUTH response.
Dec 11 23:16:24.267: CMTS Received TEK REQ.
Dec 11 23:16:24.267: Message Digest Verification Failed.
Dec 11 23:16:24.267: Sending KEK INVALID.
Dec 11 23:16:24.323: CMTS Received AUTH REQ.
Dec 11 23:16:24.323: Find a match CM key for abcd.0123.4455
```

Related Commands

Command	Description
debug cable bpiatp	Displays debugging information for BPI-related messages that the CMTS sends or receives.
debug cable privacy	Displays debugging information whenever the BPI state changes or a BPI event occurs.

debug cable l2-vpn

To display debugging messages for the Layer 2 mapping of cable modems to particular permanent virtual connections (PVC) or to a virtual local area network (VLAN), use the **debug cable l2-vpn** command in privileged EXEC mode. To stop the display of debugging messages, use the **no** form of this command.

debug cable l2-vpn [**conditional**]

no debug cable l2-vpn [**conditional**]

Note This command is not supported for the Cisco uBR10012 router, through release 12.3(13a)BC.

Syntax Description

conditional	(Optional) Displays the packets that are sent or received for a particular cable modem or cable interface. **Note** The **conditional** option does not display any output until you have also enabled debugging for a particular interface, using the **debug cable interface** command, or for a particular MAC address, using the **debug cable mac-address** command.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)BC3	This command was introduced for Cisco uBR7100 series and Cisco uBR7246VXR universal broadband routers to debug the Layer 2 mapping of cable modems to a PVC on an Asynchronous Transfer Mode (ATM) interface.
12.2(15)BC2	Support was added for the debugging of the Layer 2 mapping of cable modems to a virtual local area network (VLAN) on an outbound Ethernet interface.

Usage Guidelines

The **debug cable l2-vpn** command displays status information for the mapping of cable modems to PVCs on an ATM interface (see the **cable vc-map** command) or to a VLAN on an Ethernet, Fast Ethernet, or Gigabit Ethernet interface (see the **cable dot1q-vc-map** command). The debug messages show when a cable modem is mapped to a PVC or VLAN, when the mapping is changed or removed, and when packets are sent and received over the mapping.

The **conditional** option displays information for each packet that is sent and received over an ATM PVC or VLAN mapping. Because this can produce a large volume of debug information, the **conditional** option can be used only when you have also enabled debugging for a particular interface or MAC address, using the **debug cable interface** and **debug cable mac-address** commands, respectively.

Examples

The following example shows typical output for the **debug cable l2-vpn** command when a cable modem is mapped to an ATM PVC:

```
Router# debug cable l2-vpn

CMTS L2 VPN debugging is on

Router# configure terminal
Router(config)# cable l2-vpn-service atm-vc
Router(config)# cable vc-map 0007.0e03.69f9 ATM2/0 1/1

6d00h: Associating vc ATM2/0.1 1/1 to CM 0007.0e03.69f9 sid 0x1
6d00h: Writing vc-map info to sid 0x1

Router(config)#
```

The following example shows typical output for the **debug cable l2-vpn** command when a cable modem is mapped to an IEEE 802.1Q VLAN:

```
Router# debug cable l2-vpn

CMTS L2 VPN debugging is on

Router# configure terminal
Router(config)# cable l2-vpn-service dot1q
Router(config)# cable dot1q-vc-map 0007.0e03.69f9 FastEthernet0/0 5
Router(config)#

Set promiscuous mode for FastEthernet0/0
Mapped DS srv flow 13 on Cable5/0 to FastEthernet0/0 VLAN 5
Mapped US srv flow 11 sid 31 on Cable5/0 to FastEthernet0/0 VLAN 5
```

The following example shows typical output for the **debug cable l2-vpn** command when a mapping is deleted:

```
Router# debug cable l2-vpn

CMTS L2 VPN debugging is on

Router# configure terminal
Router(config)# no cable vc-map 0007.0e03.69f9 ATM2/0 1/1

6d00h: Disassociating vc ATM2/0.1 1/1 from CM 0007.0e03.69f9 sid 0x1
6d00h: Erasing vc-map info to sid 0x1

Router(config)#
```

The following example shows typical output for the **conditional** option. This example shows output for traffic to and from one particular cable modem. Each debug message shows the size of the packet, the source and destination MAC addresses, the cable interface and SID being used, and the ATM interface and PVC/PVI being used.

```
Router# debug cable mac-address 000C.0807.06.05
Router# debug cable l2-vpn conditional

CMTS L2 VPN conditional debugging is on

6d00h: Fwd pkt size 74 from 000C.0807.0605 on Cable4/0:0x1 to 0900.2b00.000f on ATM2/0:1/1
6d00h: Fwd pkt size 74 from 000C.0807.0605 on Cable4/0:0x1 to 0900.07ff.ffff on ATM2/0:1/1
6d00h: Fwd pkt size 1028 from 000C.0807.0605 on Cable4/0:0x1 to 0002.4a1d.dc1d on
ATM2/0:1/1
```

```
6d00h: Send pkt size 1020 encsize 6 from 0002.4a1d.dc1d on ATM2/0:1/1 to 000C.0807.0605 on
Cable4/0:0x1
6d00h: Fwd pkt size 74 from 000C.0807.0605 on Cable4/0:0x1 to 0900.07ff.ffff on ATM2/0:1/1

Router#
```

Related Commands

Command	Description
cable dot1q-vc-map	Maps a cable modem to a particular Virtual Local Area Network (VLAN) on a local outbound Ethernet interface.
cable l2-vpn-service atm-vc	Enables the use of Layer 2 tunnels for the Customer Premises Equipment (CPE) traffic that is behind cable modems, so that individual CPE traffic can be routed to a particular PVC on an ATM interface.
cable l2-vpn-service dot1q	Enables the use of Layer 2 tunnels so that traffic for individual cable modems can be routed over a particular Virtual Local Area Network (VLAN).
cable vc-map	Maps a cable modem to a particular PVC on an ATM interface.
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-address	Enables debugging output for the cable modems that match the specified hardware (MAC) address or range of addresses.
show cable l2-vpn dot1q-vc-map	Displays the mapping of one or all cable modems to IEEE 802.1Q Virtual Local Area Networks (VLANs) on the router's Ethernet interfaces.
show cable l2-vpn vc-map	Displays the mapping of one or all cable modems to PVCs on the ATM interfaces.

debug cable load-balance

To display debugging messages for load-balancing operations on the router, use the **debug cable load-balance** command in privileged EXEC mode. To stop the display of debugging messages, use the **no** form of this command.

debug cable load-balance [**error**]

no debug cable load-balance [**error**]

Syntax Description

error	(Optional) Displays debugging messages about errors that might occur during load-balancing operations about the actual number of CMs that are active on a channel.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)BC1	This command was introduced for the Cisco uBR7246VXR and Cisco uBR10012 routers.

Usage Guidelines

The **debug cable load-balance** command displays debugging output for when cable modems come online those cable interfaces that are part of a load balance group. It also displays debugging messages when cable modems are moved to achieve balanced loads on those cable interfaces.

Tip Because this command can produce a large volume of debug information, Cisco recommends that you limit debugging output to a particular interface or MAC address, using the **debug cable interface** and **debug cable mac-address** commands, respectively.

Examples

The following example shows typical output for the **debug cable load-balance** command for a particular cable interface:

```
Router# debug cable interface c3/0
Router# debug cable load-balance

CMTS load balancing debugging is on

*Feb 13 13:39:21.594 PDT: lb: Removing Modem entry 0050.7318.e615: at target (upstream)

*Feb 13 13:39:50.754 PDT: lb: US request from Cable3/0/U0[0050.7366.1f7b]: target U3
[enforce], 0 retries

*Feb 13 12:53.010 PDT: lb: Moving Modem 0050.7366.218b from Cable3/0/U0 to U2
```

```
*Feb 13 13:40:03.034 PDT: lb: Moving Modem 0050.7366.1c7f from Cable3/0/U1 to U2
*Feb 13 13:40:04.790 PDT: lb: US request from Cable3/0/U1[0050.7318.e615]: target U2
[enforce], 0 retries
*Feb 13 13:40:11.150 PDT: lb: Removing Modem entry 0050.7318.e615: at target (upstream)
*Feb 13 13:40:23.042 PDT: lb: Moving Modem 0050.7366.21c7 from Cable3/0/U0 to U3
*Feb 13 13:40:23.042 PDT: lb: Moving Modem 0050.7366.2197 from Cable3/0/U0 to U3
*Feb 13 13:40:23.042 PDT: lb: Moving Modem 0050.7366.1f75 from Cable3/0/U0 to U3

Router#
```

The following example shows typical output for the **debug cable load-balance** command when the **error** option is used to display messages about possible errors in the number of CMs that are active on the interface.

```
Router# debug cable interface c5/1/0
Router# debug cable load-balance error

Router#
CMTS load balancing error debugging is on

lb: c5/1/0/U4: Total modems: 126 active modems: 123
c5/1/0: delete_sid_state on modem 0001.0203.0405 in state offline
```

Related Commands

Command	Description
cable load-balance exclude	Excludes a particular cable modem, or all cable modems from a particular vendor, from one or more types of load-balancing operations.
cable load-balance group (global configuration)	Creates and configures a load-balance group.
cable load-balance group (interface configuration)	Assigns a downstream to a load-balance group.
cable load-balance group interval	Configures the frequency of the load-balancing policy updates.
cable load-balance group policy ugs	Configures how the Cisco CMTS should load balance cable modems with active unsolicited grant service (USG) service flows.
cable load-balance group threshold	Configures the threshold values that a load-balance group should use for load-balancing operations.
cable upstream load-balance group	Assigns an upstream to a load-balance group.
clear cable load-balance	Clears the counters or state machine used to track load-balancing operations.
debug cable interface	Enables debugging output for a specific cable interface.
debug cable l2-vpn	Displays debugging messages for load-balancing operations on the router.
debug cable mac-address	Enables debugging output for the cable modems that match the specified hardware (MAC) address or range of addresses.
show cable load-balance	Displays real-time statistical and operational information for load-balancing operations.

debug cable mac-address

To display debug information for a specific CM, use the **debug cable mac** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cable mac-address *address* [*address-mask*] [**verbose**]

no debug cable mac-address *address* [*address-mask*] [**verbose**]

Syntax Description

address	Specifies the particular MAC address to debug.
address-mask	Specifies an address mask to indicate a range of addresses to debug. This mask is bit-ANDed with the given address, and debug messages are displayed for any CM that matches the resulting non-zero bits.
verbose	Specifies detailed output for the particular MAC address.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1 T	This command was introduced.

Usage Guidelines

You can repeat this command for other MAC addresses. Each time you specify a different MAC address, debugging is turned on for that particular MAC address.

If you enter two commands with the same MAC address, but with different values for the *mask* or **verbose** keywords, the router treats both commands as the same. In this case, the latest debugging information supersedes the previous debugging information.

Note Do not use this command if you have a large number of CMs on your network. The Cisco CMTS will become flooded with console printouts.

Examples

The following example demonstrates how to enable debugging for all traffic coming from cable interface 3/0 of CMs with the MAC address 00E0.1Exx.xxxx:

```
Router# debug cable interface c3/0
Router# debug cable mac-address 00E0.1E00.0000 ffff.ff00.0000
Router#
004042: Jul  2 08:47:13.656: Ranging Modem 00E0.1E23.4567, SID 73 on Interface Cable3/U0
004043: Jul  2 08:47:13.656: Ranging request from 00E0.1E23.4567, SID 73 [73] on Interface
Cable3/U0
004044: Jul  2 08:47:13.656: Timing error 0, power error 0.25dB, freq error 4 (adjust 0)
004045: Jul  2 08:47:13.656: Ranging successful
```

Tip To monitor a specific CM when it comes online, use the **debug cable mac-address, debug cable mac-protocol,** and **debug cable registration** commands.

Table 0-256 debug cable mac Command Field Descriptions

Field	Description
SID value is....	Reports the service ID of the modem. The range is from 1 through 891. The information on this line should agree with the first line of the return (that is, Ranging Modem with Sid...).
CM mac address....	The MAC address of the specified CM.
Timing offset is....	The time by which to offset the frame transmission upstream so that the frame arrives at the expected minislot time at the CMTS.
Power value is FE0, or 0 dB	The raw value derived from the 3137 Broadcom chip. Alternatively, the dB value specifies the relative change in the transmission power level that the CM needs to make sure that transmissions arrive at the CMTS at the desired power level.
Freq Error = ...	The raw value derived from the 3137 Broadcom chip.
Freq offset is	Specifies the relative change in the transmission frequency that the CM will make to match the CMTS.

Related Commands

Command	Description
debug cable dynsrv	Displays debugging information about DOCSIS 1.1 dynamic service flow messages.
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-protocol	Displays debugging output for the MAC layer protocol.
debug cable mac-scheduler	Displays debugging output for the MAC layer scheduler and admission control activities.
debug cable registration	Displays debugging output for the registration messages sent when a CM comes online with the CMTS.
debug cable tlvs	Enables debugging for the Type/Length/Value encodings (TLVs) parsed by the DOCSIS 1.1 TLV parser/encoder.
show controllers cable	Displays interface controller information for the specified slot.

debug cable mac-protocol

To display MAC-layer information for a specific CM, use the **debug cable mac** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cable mac-protocol

no debug cable mac-protocol

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1 T	This command was introduced.

Usage Guidelines

Do not use this command if you have a large number of CMs on your network. The Cisco CMTS will become flooded with console printouts.

Examples

The following example shows the return for the MAC layer:

```
router# debug cable mac-protocol

CMTS mac debugging is on
19:46:27: Ranging Modem with Sid 1 on i/f : Cable6/0/U0

19:46:27: Got a ranging request
19:46:27: SID value is 1 on Interface Cable6/0/U0
19:46:27: CM mac address 00:E0:1E:B2:BB:07
19:46:27: Timing offset is 0
19:46:27: Power value is FE0, or 0 dB
19:46:27: Freq Error = 0, Freq offset is 0
19:46:27: Ranging has been successful for SID 1 on Interface Cable6/0/U0

19:46:29: Ranging Modem with Sid 2 on i/f : Cable6/0/U0
19:46:29: Got a ranging request
19:46:29: SID value is 2 on Interface Cable6/0/U0
19:46:29: CM mac address 00:E0:1E:B2:BB:8F
19:46:29: Timing offset is 1
19:46:29: Power value is 1350, or 0 dB
19:46:29: Freq Error = 0, Freq offset is 0
19:46:29: Ranging has been successful for SID 2 on Interface Cable6/0/U0

19:46:32: Ranging Modem with Sid 3 on i/f : Cable6/0/U0

19:46:32: Got a ranging request
19:46:32: SID value is 3 on Interface Cable6/0/U0
19:46:32: CM mac address 00:E0:1E:B2:BB:B1
19:46:32: Timing offset is FFFFFFFF
19:46:32: Power value is 1890, or -1 dB
```

```
19:46:32: Freq Error = 0, Freq offset is 0
19:46:32: Ranging has been successful for SID 3 on Interface Cable6/0/U0

19:46:34: Ranging Modem with Sid 5 on i/f : Cable6/0/U0
```

Tip To monitor a specific CM when it comes online, use the **debug cable mac-address, debug cable mac-protocol,** and **debug cable registration** commands.

Table 0-257 debug cable mac-protocol Command Field Descriptions

Field	Description
SID value is....	Reports the service ID of the modem. The range is from 1 through 8191. The information on this line should agree with the first line of the return (that is, Ranging Modem with Sid...).
CM mac address....	The MAC address of the specified CM.
Timing offset is....	The time by which to offset the frame transmission upstream so that the frame arrives at the expected minislot time at the CMTS.
Power value is FE0, or 0 dB	The raw value derived from the Broadcom chip. Alternately, the dB value specifies the relative change in the transmission power level that the CM needs to make so that transmissions arrive at the CMTS at the desired power level.
Freq Error = ...	The raw value derived from the Broadcom chip.
Freq offset is	Specifies the relative change in the transmission frequency that the CM will make to match the CMTS.

Related Commands

Command	Description
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-address	Displays debugging output for the CMs that match the specified address or range of addresses.
debug cable mac-scheduler	Displays debugging output for the MAC layer scheduler and admission control activities.
debug cable registration	Displays debugging output for the registration messages sent when a CM comes online with the CMTS.
show controllers cable	Displays interface controller information for the specified slot.

debug cable mac-scheduler

To display information for the MAC layer's scheduler and admission control activities, use the **debug cable mac-scheduler** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cable mac-scheduler [**admission control | upstream-utilization | ubg**]

no debug cable mac-scheduler [**admission control | upstream-utilization | ubg**]

Syntax Description

admission-control	(Optional) Displays debugging output for the MAC scheduler's admission control activities, which controls the percentage of overbooking allowed on the upstream channel.
upstream-utilization	(Optional) Displays debugging output for upstream utilization.
ubg	(Optional) Displays debugging output for all upstream bonding groups.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(4)CX	This command was introduced for the Cisco uBR10012 router.
12.2(4)BC1	Support for this command was added to the Release 12.2 BC train.
12.2(33)SCC	A new keyword, **ubg**, was added to this command in Cisco IOS Release 12.2(33)SCC.

Usage Guidelines

Do not use this command if you have a large number of CMs on your network. The Cisco CMTS will become flooded with console printouts.

Caution The debug commands are primarily intended for use in controlled test and troubleshooting situations with a limited volume of traffic. In particular, avoid using the **debug cable mac-scheduler** command on an interface with a significant number of CMs, because the resulting volume of debug output could impact system performance. Cisco recommends that when you use the **debug cable mac-scheduler** command, you limit its output to a particular interface or CM, using the **debug cable mac-address** or **debug cable interface** commands.

Examples

The following example shows debugging being turned on for the MAC scheduler:

```
Router# debug cable mac-scheduler
CMTS scheduler debugging is on

Router#
```

The following example shows admission-control debugging being turned on for a particular CM with the MAC address of 000C.1234.5678, and the typical messages that appear if the CM cannot register because the channel is oversubscribed.

```
Router# debug cable mac-address 000C.1234.5678 verbose
Router# debug cable mac-scheduler admission-control
CMTS scheduler debugging is on

Router#

Oct 23 09:38:56.701: MSCHED_CAC: Admit req for US service flow.
Oct 23 09:38:56.701:    SFID=3 SID=1 AdmQoS=5
Oct 23 09:38:56.701:    Admission check for BE service
Oct 23 09:38:56.701:            Min rsvd rate : 1800000 bps
Oct 23 09:38:56.701:            Failed to allocate bandwidth
Oct 23 09:38:56.701:    Admit req rejected.
```

The following example shows the debugging output for upstream bonding groups on a particular CM with the MAC address of 001e.6bfb.153c:

```
Router# debug cable mac-address 001e.6bfb.153c verbose
Router# debug cable mac-scheduler ubg

CMTS mac-scheduler ubg debugging is on

5:38:17 PM?Jul 21 12:07:01.318: cmts_schedule_solc_bgrant:Sid=8 US[2] sc=0/0 type=5
bytes:1605 cbi_p:0xE326494
Jul 21 12:07:01.318: cmts_schedule_solc_bgrant: num_chan: 4, avail_chan: 4
Jul 21 12:07:01.318: cmts_schedule_solc_bgrant: shp_delay = 0 partial_len = 0 num_bytes =
6420
Jul 21 12:07:01.318: cmts_sched_alloc_sbi_p: Alloc sbi_p: 0x1FB7F984
Jul 21 12:07:01.318: cmts_sched_alloc_bg_pgrant_p: alloc bg_pg_p: 0x1FB87938
Jul 21 12:07:01.318: cmts_sched_slotq_enq_bpgrant:bg_pg_p:0x1FB87938 sid:8 US[2]
sbip:0x1FB7F984 bytes:0 ref:1
Jul 21 12:07:01.318: cmts_sched_slotq_enq_breq: US[1]: sbip:0x1FB7F984 sbi.tbytes:1605
ref:2 Sid=8 bytes:1605, mslots:75
Jul 21 12:07:01.318: cmts_sched_slotq_enq_breq: US[2]: sbip:0x1FB7F984 sbi.tbytes:3210
ref:3 Sid=8 bytes:1605, mslots:75
Jul 21 12:07:01.318: cmts_sched_slotq_enq_breq: US[3]: sbip:0x1FB7F984 sbi.tbytes:4815
ref:4 Sid=8 bytes:1605, mslots:75
Jul 21 12:07:01.318: cmts_sched_slotq_enq_breq: US[4]: sbip:0x1FB7F984 sbi.tbytes:6420
ref:5 Sid=8 bytes:1605, mslots:75
Jul 21 12:07:01.318: cmts_sched_free_sbi_p: sbi_p:0x1FB7F984 Sid=8 ref:5
Jul 21 12:07:01.318: cmts_sched_free_sbi_p: sbi_p:0x1FB7F984 Sid=8 ref:4
Jul 21 12:07:01.318: cmts_sched_free_sbi_p: sbi_p:0x1FB7F984 Sid=8 ref:3
Jul 21 12:07:01.318: cmts_sched_free_sbi_p: sbi_p:0x1FB7F984 Sid=8 ref:2
Jul 21 12:07:01.322: cmts_sched_free_sbi_p: sbi_p:0x1FB7F984 Sid=0 ref:1
Jul 21 12:07:01.322: cmts_sched_free_sbi_p: Free sbi_p: 0x1FB7F984
Jul 21 12:07:01.322: cmts_sched_free_bg_pgrant_p: free bg_pg_p: 0x1FB87938
```

Related Commands

cable upstream admission-control	Determines the percentage of overbooking allowed on the upstream channel.
debug cable mac-address	Displays debugging output for the CMs that match the specified address or range of addresses.
debug cable mac-protocol	Displays debug messages for the MAC-layer protocol.
debug cable us-adm-ctrl	Displays debug messages for upstream admission control activity.

debug cable map

To display debugging messages for DOCSIS MAC-layer MAP messages, use the **debug cable map** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cable map [**error** | **sid** [*sid-num*]]

no debug cable map [**error** | **sid** [*sid-num*]]

Syntax Description

error	Displays debugging messages about DOCSIS MAP messages that were received with errors.
sid [*sid-num*]	Specifies the specific service ID (SID) to be debugged in DOCSIS MAP messages. The valid range for the optional *sid-num* value is 1 to 8191. If sid-num is not specified, debugging messages are shown for all SIDs.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1 T	This command was introduced.
12.2(15)BC1	The **error** option was added.

Examples

The following example shows how to display all MAP messages, with and without data grants:

```
Router# debug cable map

19:41:53: On interface Cable6/0, sent 5000 MAPs, 1321 MAPs had grant(s)Long Grants
13256993, Total Short Grants 223

------------------ MAP MSG  --------------------
us_ch_id: 1   ucd_count: 5  num_elems: 9  reserved: 0
Alloc Start Time: 33792        Ack Time: 33618
Rng_bkoff_start: 0    Rng_bkoff_end: 2
Data_bkoff_start: 1   Data_bkoff_end: 3:
sid:16383   iuc:1   mslot_offset:0
sid:0   iuc:7   mslot_offset:40

------------------ MAP MSG  --------------------
us_ch_id: 1   ucd_count: 5  num_elems: 7  reserved: 0
Alloc Start Time: 33712        Ack Time: 33578
Rng_bkoff_start: 0    Rng_bkoff_end: 2
Data_bkoff_start: 1   Data_bkoff_end: 3
sid:2   iuc:6   mslot_offset:0
sid:16383   iuc:1   mslot_offset:16
sid:0   iuc:7   mslot_offset:40
```

Table 0-258 describes the fields displayed by the **debug cable map** command.

Table 0-258 debug cable map Field Descriptions

Field	Description
sent 5000 MAPs	Total number of maps transmitted.
MAPs had grant(s) Long Grants	Total number of grants considered long-sized by CMTS.
Total Short Grants	Total number of grants considered short-sized by CMTS.
us_ch_id	Identifies the upstream channel ID for this message.
ucd_count	Number of upstream channel descriptors (UCDs).
num_elems	Number of information elements in the map.
reserved	Reserved for alignment.
Alloc Start Time	Start time from CMTS initialization (in minislots) for assignments in this map.
Ack Time	Latest time from CMTS initialization (in minislots) processed in upstream. The CMs use this time for collision detection.
Rng_bkoff_start	Initial backoff window for initial ranging contention, expressed as a power of 2. Valid values are from 0 to 15.
Rng_bkoff_end	Final backoff window for initial ranging contention, expressed as a power of 2. Valid values are from 0 to 15.
Data_bkoff_start	Initial backoff window for contention data and requests, expressed as a power of 2. Valid values are from 0 to 15.
Data_bkoff_end	Final backoff window for contention data and requests, expressed as a power of 2. Valid values are from 0 to 15.
sid	Service ID.
iuc	Interval usage code (IUC) value.
mslot_offset	Minislot offset.

The following shows typical output for the **debug cable map error** command:

```
Router# debug cable int cable 4/0
Router# debug cable map error
CMTS map errors debugging is ON

Router#

00:11:21: ##### Bad IE-offset for prenull IE[1]
#####----------------------  MAP MSG  ----------------------
us_ch_id: 1   ucd_count: 11  num_elems: 4  reserved: 0
Alloc Start Time: 20007713        Ack Time: 20007291
Rng_bkoff_start: 0    Rng_bkoff_end: 3
Data_bkoff_start: 0   Data_bkoff_end: 4
sid:16383   iuc:1   mslot_offset:0
sid:16383   iuc:1   mslot_offset:0
sid:16383   iuc:1   mslot_offset:158
sid:0   iuc:7   mslot_offset:160
```

Related Commands

Command	Description
show controllers cable	Displays interface controller information for the specified slot.

debug cable metering

To enable debugging of usage-based billing operations, use the **debug cable metering** command in privileged EXEC mode. To turn off debugging messages, use the **no** form of this command.

debug cable metering

no debug cable metering

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(9a)BC	This command was introduced.

Examples

The following example shows how to enable debugging for usage-based billing on the Cisco CMTS, and then shows examples of the debugging messages that can be displayed.

```
Router# debug cable metering
CMTS metering debug is ON

Router#


Cannot get Metering CMTS IP/MAC Addresses.
CMTS Billing ip address interface name c6/0
CMTS SFLOG File BBCMTS_20000708-120931 open failed (23)
CMTS Metering open file CMTS0120000708-120931 successfully.
CMTS Metering LOCAL data to write 5600, wrote 4800 to file local file
CMTS Metering data to write 6400, wrote 6400 to file B_20000708-120931
CMTS Metering invalid FD for metering file
CMTS Metering Produce Metering - END Time= 2002-05-25T14:41:29Z
CMTS Metering closed file
CMTS Metering: xml end queue data malloc failed.
CMTS Metering file header len=480
CMTS Metering: xml header enqueue failed.
CMTS Metering: xml header enqueue OK.
CMTS Metering: xml end queue data malloc failed.
CMTS Metering xml_ending len 128
CMTS Metering: xml end enqueue OK.
CMTS Metering element header length 440
CMTS Metering sflog read buf = 4800
CMTS Metering: malloc failed.
CMTS Metering: SFLOG file 4800 size tx failed.
CMTS Metering: SFLOG tx OK 4800 size.
Cmts Metering: SFLOG file doesn't exist
CMTS Metering abort producing billing due to file open failure.
CMTS Metering: enqueue failed
CMTS Metering get invalid CM.
CMTS Metering valid cm 192.168.100.101, prim sid 13
CMTS Metering interval 30
Invalid Metering Timer Expired...Stop Timer.
```

Related Commands

Command	Description
cable metering destination	Enables usage-based billing and streams the billing records to an external server.
cable metering filesystem	Enables usage-based billing and writes the billing records to a file on a local file system.
show cable metering-status	Displays information about the most recent usage-based billing operation.
snmp-server enable traps cable	Enables the sending of Simple Network Management Protocol (SNMP) traps for cable-related events.

debug cable mdd

To display debugging messages of the MAC domain descriptor, use the **debug cable mdd** command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

debug cable mdd

no debug cable mdd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced in Cisco IOS Release 12.2(33)SCC.

Examples

The following is a sample output of the **debug cable mdd** command:

```
Router# debug cable mdd

CMTS mdd debugging is on
Router#
Jul  7 20:57:27.038: Cable8/0/0: size 228 mdd_tlv_size 198 num_frag 1 seq_num 1
test_mdd_tlv_length 0
Jul  7 20:57:27.038: Cable8/0/0 MDD datagram size 228, msg len 226, ehdr type_or_
len 208, tlv_size 198 max_pak_size 1518
Jul  7 20:57:27.038: MDD MESSAGE
Jul  7 20:57:27.038:   FRAME HEADER
Jul  7 20:57:27.038:     FC, MAC_PARM, LEN       - 0xC2, 0x00, 0x00E2
Jul  7 20:57:27.038:   MAC MANAGEMENT MESSAGE HEADER
Jul  7 20:57:27.038:     DA, SA                  - 01E0.2F00.0001, 0014.F1E5.381
8
Jul  7 20:57:27.038:     msg LEN                 - 0x00D0
Jul  7 20:57:27.038:     DSAP, SSAP              - 0, 0
Jul  7 20:57:27.038:     control, version, type  - 0x03, 0x04, 0x21
Jul  7 20:57:27.038:     0x00D0: 01 02 07 07 01 01 01 02 02 00 00 08 01 01 09 01

Jul  7 20:57:27.038:     0x00E0: 00 0C 01 01
Jul  7 20:57:35.038: Non-primary MDD from Cable8/0/0 to RFID 961:
Jul  7 20:57:35.038:   message dump:
Jul  7 20:57:35.038:   0x0000: C2 00 00 1C 9C 24 01
E0 2F 00 00 01 00 14 F1 E5
Jul  7 20:57:35.038:   0x0010: 38 18 00 0A 00 00 03 04 21 00 01 01 01 62
Jul  7 20:57:35.038: Non-primary MDD from Cable8/0/0 to RFID 962:
Jul  7 20:57:35.038:   message dump:
Jul  7 20:57:35.038:   0x0000: C2 00 00 1C 9C 24 01 E0 2F 00 00 01 00 14 F1 E5
Jul  7 20:57:35.038:   0x0010: 38 18 00 0A 00 00 03 04 21 00 01 01 01 63
Jul  7 20:57:35.038: Non-primary MDD from Cable8/0/0 to RFID 963:
Jul  7 20:57:35.038:   message dump:
Jul  7 20:57:35.038:   0x0000: C2 00 00 1C 9C 24 01 E0 2F 00 00 01 00 14 F1 E5
Jul  7 20:57:35.038:   0x0010: 38 18 00 0A 00 00 03 04 21 00 01 01 01 64
```

Related Commands

Command	Description
debug cable md-sg	Displays the MAC domain service group debugging messages.

debug cable md-sg

To enable debugging information for MAC domain service group messages, use the **debug cable md-sg** command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

debug cable md-sg

no debug cable md-sg

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced in Cisco IOS Release 12.2(33)SCC.

Examples

The following is a sample output of the **debug cable md-sg** command:

```
Router# debug cable md-sg

CMTS md-sg debugging is on

Router#
Jul  7 21:04:24.938: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:04:24.938: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:04:24.938: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:04:38.858: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:04:38.858: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:04:38.858: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:04:50.438: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:04:50.438: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:04:50.438: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:04:52.898: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:04:52.898: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:04:52.898: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:05:06.938: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:05:06.938: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:05:06.938: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:05:08.378: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:05:08.378: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:05:08.378: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:05:22.538: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:05:22.538: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:05:22.538: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:05:24.998: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:05:24.998: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
```

```
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:05:24.998: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
Jul  7 21:05:36.818: Ambiguity Resolution: B_INIT_RNG_REQ notified (us = 1).
Jul  7 21:05:36.818: Ambiguity Resolution Validate Candidate: (B_INIT_RNG_REQ) f
ound[ 1 ] uschan = 0x1, reachable = 0x1, failed = 0x0.
Jul  7 21:05:36.818: Ambiguity Resolution: Done (case 1 b_init) with sg_id = 1.
```

Related Commands

Command	Description
debug cable mdd	Displays debugging messages of the MAC domain descriptor.

debug cable multicast counter clear

To reset debugging of multicast counters, use the **debug cable multicast counter clear** command in privileged EXEC mode.

debug cable multicast counter clear {**all** | *multicast-group-address* | *mac-address*}

Syntax Description

all	Resets all debug counters.
multicast-group-address	IP address of the multicast group.
mac-address	MAC address of the cable modem.

Command Default

None

Command Modes

Privileged EXEC(#)

Command History

Release	Modification
12.2(33)SCE	This command was introduced.

Usage Guidelines

When enabled, this debug command does not provide any debugging output for counter values. When you use this command to reset debug counters associated with a multicast group IP address or MAC address, the command does not return any debugging messages. You will have to use the **show cable multicast debug** command to verify the debug counters.

Examples

The following is a sample output of the **debug cable multicast counter clear** command with the keyword **all**:

```
Router# debug cable multicast counter clear all

All multicast debug counters cleared.
```

Related Commands

Command	Description
debug cable multicast counter start	The Cisco CMTS router starts collecting multicast debug counters based on a particular multicast group or a cable modem.
debug cable multicast counter stop	The Cisco CMTS router stops collecting multicast debug counters.
show cable multicast debug	Displays information about debug counters.

debug cable multicast counter start

To enable the Cisco CMTS router to start collecting multicast debug counters based on a particular multicast group or a cable modem, use the **debug cable multicast counter start** command in privileged EXEC mode.

debug cable multicast counter start {*multicast-group-address* | *mac-address*}

Syntax Description

multicast-group-address	IP address of the multicast group.
mac-address	MAC address of the cable modem.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCE	This command was introduced.

Usage Guidelines

When enabled, this debug command does not provide any debugging output or messages. You must use the **show cable multicast debug** command to verify the debug counters. If you want to verify multicast group specific or MAC specific debug counters, you must turn on debugging of multicast counters using the **debug cable multicast counter start** command before using the **show cable multicast debug** command.

Examples

The following is a sample output of the **debug cable multicast counter start** command on the Cisco uBR10012 router:

```
Router# debug cable multicast counter start 001a.c3ff.d41a
```

Related Commands

Command	Description
debug cable multicast counter clear	Resets debugging of multicast counters.
debug cable multicast counter stop	The Cisco CMTS router stops collecting multicast debug counters.
show cable multicast debug	Displays information about debug counters.

debug cable multicast counter stop

To stop the Cisco CMTS router from collecting multicast debug counters, use the **debug cable multicast counter stop** command in privileged EXEC mode.

debug cable multicast counter stop

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC(#)

Command History

Release	Modification
12.2(33)SCE	This command was introduced.

Usage Guidelines

When enabled, this debug command does not provide any debugging output or messages. You must use the **show cable multicast debug** command to verify the debug counters.

Examples

The following is a sample output of the **debug cable multicast counter stop** command on the Cisco uBR10012 router:

```
Router# debug cable multicast counter stop
```

Related Commands

Command	Description
debug cable multicast counter clear	Resets debugging of multicast counters.
debug cable multicast counter start	The Cisco CMTS router starts collecting multicast debug counters based on a particular multicast group or a cable modem.
show cable multicast debug	Displays information about debug counters.

debug cable multicast forwarding

To display debugging messages about downstream forwarding interfaces for cable modems that are wideband online (w-online) and multicast quality of service (MQoS) enabled, use the **debug cable multicast forwarding** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug cable multicast forwarding [**verbose**]

no debug cable multicast forwarding

Syntax Description

verbose	(Optional) Enables debugging with verbose description that provides detailed information about downstream forwarding interfaces.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCE	This command was introduced.

Usage Guidelines

Ensure that you turn on this debug command in a controlled environment where not too many multicast requests are being handled by the Cisco CMTS router, because this debug command will log a lot of debugging messages.

Examples

The following is a sample output of the **debug cable multicast forwarding** command that displays debugging messages about downstream forwarding interfaces for cable modems that are w-online with MQoS-enabled multicast requests on the Cisco uBR10012 router:

```
Router# debug cable multicast forwarding

! IGMP join

04:01:03: MCAST DS: [N/A:230.4.4.4:N/A]: No broadcast forwarding interface available.
Looking for default forwarding interface
04:01:03: MCAST DS: Req Attr: 0x0 Forb Attr: 0x0 CM [001c.eaa5.06ce:20].
04:01:03: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: Default Forwarding interface
found  Checking for MQos, if MQos not configured use the default.
04:01:03: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: CM [001c.eaa5.06ce]: No
attribute masks configured for GC [1], GQC [1], SC[2]
04:01:03: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: CM [001c.eaa5.06ce]: Forwarding
Interface selected, Bundle [Bundle123]
04:01:03: MCAST DS: [N/A:230.4.4.4:N/A]: No broadcast forwarding interface available.
Looking for default forwarding interface
04:01:03: MCAST DS: Req Attr: 0x0 Forb Attr: 0x0 CM [001c.eaa5.06ce:20].
04:01:03: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: Default Forwarding interface
found  Checking for MQos, if MQos not configured use the default.
```

```
04:01:03: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: CM [001c.eaa5.06ce]: No
attribute masks configured for GC [1], GQC [1], SC[2]
04:01:03: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: CM [001c.eaa5.06ce]: Forwarding
Interface selected, Bundle [Bundle123]
.
.
.
! IGMP leave

04:04:00: MCAST DS: [N/A:230.4.4.4:N/A]: No broadcast forwarding interface available.
Looking for default forwarding interface
04:04:00: MCAST DS: Req Attr: 0x0 Forb Attr: 0x0 CM [001c.eaa5.06ce:20].
04:04:00: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: Default Forwarding interface
found  Checking for MQos, if MQos not configured use the default.
04:04:00: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: CM [001c.eaa5.06ce]: No
attribute masks configured for GC [1], GQC [1], SC[2]
04:04:00: MCAST DS: [N/A:230.4.4.4:Wideband-Cable5/1/2:1]: CM [001c.eaa5.06ce]: Forwarding
Interface selected, Bundle [Bundle123]
.
.
.
```

The following is a sample output of the **debug cable multicast forwarding** command with the **verbose** option that provides information about the forwarding interface selection related to service flow attributes on the Cisco uBR10012 router:

```
Router# debug cable multicast forwarding verbose

*Jun  2 00:45:32.679 UTC: MCAST DS: [N/A:231.1.1.2:N/A]: No broadcast forwarding interface
available. Looking for default forwarding interface
*Jun  2 00:45:32.679 UTC: MCAST DS: Req Attr: 0x0 Forb Attr: 0x0 CM [001a.c3ff.d824:1].
*Jun  2 00:45:32.679 UTC: MCAST DS: [N/A:231.1.1.2:Wideband-Cable7/0/0:0]: Default
Forwarding interface found  Checking for MQos, if MQos not configured use the default.
*Jun  2 00:45:32.679 UTC: MCAST DS: Req Attr: 0x800000F0 Forb Attr: 0x0 CM
[001a.c3ff.d824:1].
*Jun  2 00:45:32.679 UTC: MCAST DS: Checking interface [Wideband-Cable1/0/0:0] Attr:
0x8000000F, for CM [001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: Checking interface [Wideband-Cable1/0/0:1] Attr:
0x80000000, for CM [001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: Checking interface [Wideband-Cable1/0/0:2] Attr:
0x80000000, for CM [001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: Checking interface [Wideband-Cable7/0/0:0] Attr:
0x80000000, for CM [001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: Checking interface [Wideband-Cable7/0/0:2] Attr:
0x800000F0, for CM [001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: Checking interface [Wideband-Cable7/0/1:0] Attr:
0x8000000F, for CM [001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: Selected interface [Wideband-Cable7/0/0:2], for CM
[001a.c3ff.d824:1]
*Jun  2 00:45:32.679 UTC: MCAST DS: [N/A:231.1.1.2:Wideband-Cable7/0/0:2]: CM
[001a.c3ff.d824]: Picked interface after attribute matching
*Jun  2 00:46:07.991 UTC: MCAST DS: [N/A:230.1.1.1:N/A]: No broadcast forwarding interface
available. Looking for default forwarding interface
*Jun  2 00:46:07.991 UTC: MCAST DS: Req Attr: 0x0 Forb Attr: 0x0 CM [001a.c3ff.d824:1].
*Jun  2 00:46:07.991 UTC: MCAST DS: [N/A:230.1.1.1:Wideband-Cable7/0/0:0]: Default
Forwarding interface found  Checking for MQos, if MQos not configured use the default.
*Jun  2 00:46:07.991 UTC: MCAST DS: Req Attr: 0x8000000F Forb Attr: 0x0 CM
[001a.c3ff.d824:1].
*Jun  2 00:46:07.991 UTC: MCAST DS: Checking interface [Wideband-Cable1/0/0:0] Attr:
0x8000000F, for CM [001a.c3ff.d824:1]
*Jun  2 00:46:07.991 UTC: MCAST DS: Checking interface [Wideband-Cable1/0/0:1] Attr:
0x80000000, for CM [001a.c3ff.d824:1]
*Jun  2 00:46:07.991 UTC: MCAST DS: Checking interface [Wideband-Cable1/0/0:2] Attr:
0x80000000, for CM [001a.c3ff.d824:1]
```

```
*Jun  2 00:46:07.991 UTC: MCAST DS: Checking interface [Wideband-Cable7/0/0:0] Attr:
0x80000000, for CM [001a.c3ff.d824:1]
*Jun  2 00:46:07.991 UTC: MCAST DS: Checking interface [Wideband-Cable7/0/0:2] Attr:
0x800000F0, for CM [001a.c3ff.d824:1]
*Jun  2 00:46:07.991 UTC: MCAST DS: Checking interface [Wideband-Cable7/0/1:0] Attr:
0x8000000F, for CM [001a.c3ff.d824:1]
*Jun  2 00:46:07.991 UTC: MCAST DS: Selected interface [Wideband-Cable7/0/1:0], for CM
[001a.c3ff.d824:1]
```

Related Commands

Command	Description
debug cable multicast counter clear	Resets debugging of multicast counters.
debug cable multicast counter start	The Cisco CMTS router starts collecting multicast debug counters based on a particular multicast group or a cable modem.
debug cable multicast counter stop	The Cisco CMTS router stops collecting multicast debug counters.
show cable multicast debug	Displays information about debug counters.

debug cable multicast latency

To display debugging messages about multicast latency, use the **debug cable multicast latency** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug cable multicast latency

no debug cable multicast latency

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCE	This command was introduced.

Examples

The following is a sample output of the **debug cable multicast latency** command that displays latency information related to IGMP and PCMM supported multicast sessions:

```
Router# debug cable multicast latency

! IGMP join

4:31:59 PM 00:35:53: MCAST LATENCY: Sending IPC RP->GUR for indices: Group 230.1.1.2
Source N/A sid 65535 RP-LC IPC flags 1 if_num 1832, mc_rfid 720, sfid 0, gcfgid 0, wb_chid
720 7/0/0
00:35:53: MCAST LATENCY:  Indices received from Guardian for Group:230.1.1.2 Source:N/A
Bundle:Bundle1 Interface:Integrated-Cable7/0/0:0 Sid65535 StatIndex:23 KeyIndex:0

! IGMP leave

4:33:16 PM 00:37:22: MCAST LATENCY:  Indices received from Guardian for Group:230.1.1.2
Source:N/A Bundle:Bundle1 Interface:Integrated-Cable7/0/0:0 Sid0 StatIndex:23 KeyIndex:0
```

The following is a sample output of the **debug cable multicast latency** command that displays latency information related to PCMM supported multicast sessions:

```
Router# debug cable multicast latency

! PCMM join

00:13:44: MCAST LATENCY: Sending IPC RP->GUR for indices: Group 230.1.1.1 Source N/A sid
65535 RP-LC IPC flags 1 if_num 1800, mc_rfid 0, sfid 0, gcfgid 0, wb_chid 1800 7/0/0
00:13:44: MCAST LATENCY: Sending IPC RP->GUR for indices: Group 230.1.1.1 Source N/A sid
8198 RP-LC IPC flags 1 if_num 1800, mc_rfid 0, sfid 0, gcfgid 1, wb_chid 1800 7/0/0
00:13:44: MCAST LATENCY: Group 230.1.1.1 Source N/A Sending to Guardian
WB:Wideband-Cable7/0/0:0 ID:1800 MCAST_SID:8198 ServiceClass:200 Flag:1 mqos_gc:1
```

```
00:13:44: MCAST LATENCY:  Indices received from Guardian for Group:230.1.1.1 Source:N/A
Bundle:Bundle1 Interface:Wideband-Cable7/0/0:0 Sid65535 StatIndex:11 KeyIndex:0
00:13:44: MCAST LATENCY:  Indices received from Guardian for Group:230.1.1.1 Source:N/A
Bundle:Bundle1 Interface:Wideband-Cable7/0/0:0 Sid8198 StatIndex:12 KeyIndex:0
00:10:25: MCAST LATENCY: Wideband-Cable7/0/0: IPC LC-RP CMTS_MQOS MCAST_SID 8198 SCLASS
200, DS_SFID 5, Flag 1

! PCMM leave

00:16:16: MCAST LATENCY:  Indices received from Guardian for Group:230.1.1.1 Source:N/A
Bundle:Bundle1 Interface:Wideband-Cable7/0/0:0 Sid0 StatIndex:13 KeyIndex:0
00:16:16: MCAST LATENCY: Group N/A Source N/A Sending to Guardian WB:Wideband-Cable7/0/0:0
ID:1800 MCAST_SID:8199 ServiceClass:200 Flag:0 mqos_gc:1
```

Related Commands

Command	Description
debug cable multicast forwarding	Displays debugging messages about downstream forwarding interfaces for cable modems that are wideband online (w-online).

debug cable phs

To display the activities of the payload header suppression and restoration (PHS) driver, use the **debug cable phs** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cable phs

no debug cable phs

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(4)CX	This command was introduced.
12.2(4)BC1	Support was added to the Release 12.2 BC train.

Usage Guidelines

Do not use this command when you have a large number of active CMs on your network, because it could generate a huge amount of output to the console port.

This command displays the output for both the upstream and downstream drivers. The upstream receive driver restores headers that have been suppressed by CMs, and the downstream driver suppresses specific fields in packet header before forwarding a frame to the CM.

Examples

The following example shows typical output from PHS debugging:

```
Router# debug cable phs
CMTS payload header suppression debugging is on

00:02:55: New PHS rule: 1 (SFID: 9)
   size : 34
   mask : 00 00 00 03 FC 00 00 00  field: 00 00 00 00 00 00 00 00
 Add PHS rule 1 to CFR ID 1

 00:02:57: New PHS rule: 1 (SFID: 11)
   size : 34
   mask : 00 00 00 03 FC 00 00 00  field: 00 00 00 00 00 00 00 00
 Add PHS rule 1 to CFR ID 1
Router#
```

debug cable phy

To activate debugging of messages generated in the cable physical layer, use the **debug cable phy** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable phy

no debug cable phy

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of messages generated in the PHY system, which is the physical layer where upstream and downstream activity between the Cisco CMTS and the HFC network is controlled. When this command is activated, any messages generated in the PHY system are displayed on the Cisco CMTS console.

Examples

The following is typical output from the **debug cable phy** command.

```
cmts_phy_init: mac_version == BCM3210_FPGA
bcm3033_set_tx_sym_rate(5056941)
stintctl = 0x54484800
bcm3033_set_tx_if_freq(44000000)
stfreqctl = 0x5BAAAAAA
cmts_phy_init_us: U0 part_id = 0x3136, revid = 0x05, rev_id2 = 0x64
cmts_phy_init: mac_version == BCM3210_FPGA
Media access controller chip version.
bcm3033_set_tx_sym_rate(5056941)
stintctl = 0x54484800
Physical layer symbol rate register value.
00:51:49: bcm3033_set_tx_if_freq(44000000)
00:51:49:  stfreqctl = 0x5BAAAAAA
Physical layer intermediate frequency (IF) register value.
00:51:49: cmts_phy_init_us: U0 part_id = 0x3136, revid = 0x05, rev_id2 = 0x64
Physical layer receiver chip part version.
```

debug cable privacy

To activate debugging of baseline privacy, use the **debug cable privacy** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cable privacy

no debug cable privacy

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3 XA	This command was introduced.

Usage Guidelines

This command activates debugging of the Baseline Privacy Interface (BPI) feature. When this command is activated, the BPI handler generates a debugging message whenever the BPI state changes or a BPI-related event occurs.

Tip Debugging must be enabled for one or more cable interfaces, using the **debug cable interface** command, before the **debug cable privacy** command displays any output. BPI debugging can be done only on a per-interface basis, not on the basis of a CM's MAC address.

Note This command is supported only on images that support BPI or BPI+ encryption.

Examples

The following is typical output from the **debug cable privacy** command:

```
02:32:08: CMTS Received AUTH REQ.
02:32:08: Created a new CM key for 0030.96f9.65d9.
02:32:08: CMTS generated AUTH_KEY.
02:32:08: Input : 70D158F106B0B75
02:32:08: Public Key:
02:32:08: 0x0000: 30 68 02 61 00 DA BA 93 3C E5 41 7C 20 2C D1 87
02:32:08: 0x0010: 3B 93 56 E1 35 7A FC 5E B7 E1 72 BA E6 A7 71 91
02:32:08: 0x0020: F4 68 CB 86 A8 18 FB A9 B4 DD 5F 21 B3 6A BE CE
02:32:08: 0x0030: 6A BE E1 32 A8 67 9A 34 E2 33 4A A4 0F 8C DB BD
02:32:08: 0x0040: D0 BB DE 54 39 05 B0 E0 F7 19 29 20 8C F9 3A 69
02:32:08: 0x0050: E4 51 C6 89 FB 8A 8E C6 01 22 02 34 C5 1F 87 F6
02:32:08: 0x0060: A3 1C 7E 67 9B 02 03 01 00 01
02:32:08: RSA public Key subject:
02:32:08: 0x0000: 30 7C 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05
02:32:08: 0x0010: 00 03 6B 00 30 68 02 61 00 DA BA 93 3C E5 41 7C
02:32:08: 0x0020: 20 2C D1 87 3B 93 56 E1 35 7A FC 5E B7 E1 72 BA
02:32:08: 0x0030: E6 A7 71 91 F4 68 CB 86 A8 18 FB A9 B4 DD 5F 21
```

```
02:32:08: 0x0040: B3 6A BE CE 6A BE E1 32 A8 67 9A 34 E2 33 4A A4
02:32:08: 0x0050: 0F 8C DB BD D0 BB DE 54 39 05 B0 E0 F7 19 29 20
02:32:08: 0x0060: 8C F9 3A 69 E4 51 C6 89 FB 8A 8E C6 01 22 02 34
02:32:08: 0x0070: C5 1F 87 F6 A3 1C 7E 67 9B 02 03 01 00 01
02:32:08: RSA encryption result = 0
02:32:08: RSA encrypted output:
02:32:08: 0x0000: B6 CA 09 93 BF 2C 05 66 9D C5 AF 67 0F 64 2E 31
02:32:08: 0x0010: 67 E4 2A EA 82 3E F7 63 8F 01 73 10 14 4A 24 ED
02:32:08: 0x0020: 65 8F 59 D8 23 BC F3 A8 48 7D 1A 08 09 BF A3 A8
02:32:08: 0x0030: D6 D2 5B C4 A7 36 C4 A9 28 F0 6C 5D A1 3B 92 A2
02:32:08: 0x0040: BC 99 CC 1F C9 74 F9 FA 76 83 ED D5 26 B4 92 EE
02:32:08: 0x0050: DD EA 50 81 C6 29 43 4F 73 DA 56 C2 29 AF 05 53
02:32:08: CMTS sent AUTH response.
02:32:08: CMTS Received TEK REQ.
02:32:08: Created a new key for SID 2.
02:32:08: CMTS sent KEY response.
```

Related Commands

Command	Description
debug cable bpiatp	Displays debugging information about the BPI-related messages that the CMTS sends or receives.
debug cable interface	Enables debugging output for a specific cable interface.
debug cable keyman	Displays debugging information about BPI key management.

debug cable qos

To activate quality-of-service (QoS) debugging, use the **debug cable qos** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable qos

no debug cable qos

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of QoS. When this command is activated, any messages related to QoS parameters are displayed on the Cisco CMTS console.

Examples

The following is typical output from the **debug cable qos** command:

```
CMTS_QOS_LOG_NO_MORE_QOS_INDEX
Modems cannot add more entries to the class of service table.
CMTS_QOS_LOG_NOMORE_QOSPRF_MEM
Memory allocation error when creating class of service table entry.
CMTS_QOS_LOG_NO_CREATION_ALLOWED
Class of service entry cannot be created by modem.  Use CLI or SNMP
interface instead of the modem's TFTP configuration file.
CMTS_QOS_LOG_CANNOT_REGISTER_COS_SID
A service identifier (SID) could not be assigned to the registering modem.
CMTS_QOS_LOG_CANNOT_DEREGISTER_COS_SID
The modem's service identifier (SID) was already removed.
CMTS_QOS_LOG_MSLOT_TIMEBASE_WRAPPED
The 160 KHz timebase clock drives a 26-bit counter which wraps around
approximately every 7 minutes.  This message is generated every time it
wraps around.
```

debug cable range

To display ranging messages from CMs on the HFC network, use the **debug cable range** command in privileged EXEC mode. To disable debugging output, the **no** form of the command.

debug cable range

no debug cable range

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of ranging messages from CMs on the HFC network. When this command is activated, any ranging messages generated when CMs request or change their upstream frequencies are displayed on the Cisco CMTS console. Use this command to display the details of the initial and station maintenance procedures. The initial maintenance procedure is used for link establishment. The station maintenance procedure is used for link keepalive monitoring.

Examples

The following shows typical output from the **debug cable range** command.

```
Got a ranging request
SID value is 0 on Interface Cable3/0/U0
CM mac address 00:10:7B:43:AA:21 Timing offset is 3312
3E 1E 3F FF 00 00 59 BF 01 15 F8 01 A7 00 0C F0
```

The following output shows typical output when a CM first seeks to establish a link to the Cisco CMTS. The service identifier (SID) value of 0 indicates that the modem has no assigned SID. The CM mac address is the MAC address of the modem's radio frequency (RF) interface, not its Ethernet interface. The Timing offset is a measure of the distance between the modem and the Cisco CMTS, expressed in 10.24 MHz clocks. This value is adjusted down to zero by the maintenance procedures. The first 16 bytes of the prepended header of the message are dumped in hexadecimal.

```
CM mac address 0010.7b43.aa21
found..Assigned SID #2 on Interface Cable3/0/U0
Timing offset is CF0
Power value is 15F8, or -1 dB
Freq Error = 423, Freq offset is 1692
Ranging Modem with Sid 2 on i/f : Cable3/0/U0
```

The following is typical output when the CM is first assigned a SID during initial maintenance:

```
Initial Range Message Received on Interface Cable3/0/U0
CMTS reusing old sid : 2 for modem : 0010.7b43.aa21
Timing offset is CF0
Power value is 15F8, or -1 dB
```

```
Freq Error = 423, Freq offset is 1692
Ranging Modem with Sid 2 on i/f : Cable3/0/U0
```

The following is typical output when the modem is reassigned the same SID during initial maintenance.

```
Ranging Modem with Sid 2 on i/f : Cable3/0/U0

Got a ranging request
SID value is 2 on Interface Cable3/0/U0
CM mac address 00:10:7B:43:AA:21
Timing offset is 0
Power value is 1823, or -1 dB
Freq Error = 13, Freq offset is 0
Ranging has been successful for SID 2 on Interface Cable3/0/U0
```

Output occurs when the modem is polled by the CMTS during station maintenance. Polling happens at a minimum rate of once every 10 seconds.

debug cable receive

To display debug messages for messages received on the upstream from a CM, use the **debug cable receive** command in privileged EXEC mode. To stop displaying debug messages, use the **no** form of this command.

debug cable receive

no debug cable receive

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR	This command was introduced.
12.1(1)T, 12.1(2)EC1	This command was disabled.
12.1(11b)EC	This command was removed.

Examples

The following command enables debugging of upstream messages:

```
router# debug cable receive
CMTS debug Rx debugging is on
```

The **debug cable receive** command was disabled in Cisco IOS Release 12.1 T, 12.1 EC, and later releases, although it still appears in the CLI until Cisco IOS Release 12.1(11b)EC. To monitor the packets received from the CMs, use the **debug ip packet** command instead.

Because the **debug ip packet** command produces a large volume of messages, you must use it together with an access list to restrict the messages to a particular CM or CPE device to avoid affecting system performance. For example, the following commands would set up an access list that monitors all traffic being sent between the hosts with the two specified IP addresses:

```
router# configure terminal
router(config)# access-list 150 permit ip host ip-address-1 host ip-address-2
router(config)# access-list 150 permit ip host ip-address-2 host ip-address-1
router(config)# exit
router# debug ip packet 150 detail
```

Related Commands

Command	Description
debug cable transmit	Enables debugging for transmitted packets.
debug ip packet	Enables debugging of IP packets received and transmitted by the router.

debug cable registration

To display debug messages for the CM registration process, use the **debug cable registration** command in privileged EXEC mode. To stop displaying debug messages, use the **no** form of this command.

debug cable registration

no debug cable registration

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR	This command was introduced.

Usage Guidelines

The **debug cable registration** command displays messages about the registration request and response messages sent and received by the CMTS when a CM initiates the DOCSIS registration process. To see the relevant messages, you must also enable debugging for a particular interface or CM using the **debug cable interface** or **debug cable mac-address** commands.

Examples

The following shows typical debugging output from a registration process:

```
router# debug cable interface c3/0 verbose
router# debug cable registration
CMTS registration debugging is on

Jul 4 14:31:37.471: Registration request from 0001.9659.3ef7, SID 3 on Cable3/0/U2
Jul 4 14:31:37.471: Found a network access control parameter
Jul 4 14:31:37.471: The Ntw Access Control is 1
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found a class of service block
Jul 4 14:31:37.475: The CLASS ID : 5
Jul 4 14:31:37.475: The MAX DS RATE : 3000000
Jul 4 14:31:37.475: The MAX US RATE : 2000000
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found vendor extensions
Jul 4 14:31:37.475: Cisco Vendor ID Field found(ok)
Jul 4 14:31:37.475: No. of requested phone lines is 2
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found vendor extensions
Jul 4 14:31:37.475: Cisco Vendor ID Field found(ok)
Jul 4 14:31:37.475: IP precedence specific subtype found
Jul 4 14:31:37.475: IP Precedence value: 0 Rate Limit: 10000
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found Max CPE
Jul 4 14:31:37.475: The Max CPE is 10
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found CM MIC
Jul 4 14:31:37.475: CM Mic: 5A D5 58 DC E8 2B 5B 24 6E 4E 69 84 17 B9 AB 36
```

```
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found CMTS MIC
Jul 4 14:31:37.475: CMTS Mic: A7 E4 15 6 F9 2F BA 81 FE 22 E4 92 5F 81 D4 BB
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found modem ip
Jul 4 14:31:37.475: The modem ip value is 10.200.69.90
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Found modem capabilities
Jul 4 14:31:37.475: Modem Caps Length is 18
Jul 4 14:31:37.475: Modem Caps values:
Jul 4 14:31:37.475: 0x0000: 01 01 01 02 01 00 03 01 00 04 01 00 05 01 00 06
Jul 4 14:31:37.475: 0x0010: 01 00
Jul 4 14:31:37.475: Concatenation is on for this CM
Jul 4 14:31:37.475: Unknown capability type 2: Ignored
Jul 4 14:31:37.475: Unknown capability type 3: Ignored
Jul 4 14:31:37.475: Unknown capability type 4: Ignored
Jul 4 14:31:37.475: Unknown capability type 5: Ignored
Jul 4 14:31:37.475: Unknown capability type 6: Ignored
Jul 4 14:31:37.475: This TLV is GOOD
Jul 4 14:31:37.475: Finished parsing REG Request
Jul 4 14:31:37.475: Sec sids obtained for all requested classes of service
Jul 4 14:31:37.475: Performing connection admission control (CAC) for each Sid
Jul 4 14:31:37.475: CAC Status for ClassID:5 is CAC_SUCCESS
Jul 4 14:31:37.475: Building a GOOD REG-RSP
Jul 4 14:31:37.475: Adding Modem Caps to the Response to REG RSP
Jul 4 14:31:37.475: Registration Status: ok (0)
Jul 4 14:31:37.475: ClassId:5 assigned QoS Sid:3
Jul 4 14:31:37.475: Adding Service Class Data TLV:
Jul 4 14:31:37.475: 0x0000: 01 07 01 01 05 02 02 00 03
Jul 4 14:31:37.475: Adding Modem Caps to Response:
Jul 4 14:31:37.475: 0x0000: 05 03 01 01 01
Jul 4 14:31:37.475: Registration Response:
Jul 4 14:31:37.475: 0x0000: C2 00 00 29 00 00 00 01 96 59 3E F7 00 30 7B F9
Jul 4 14:31:37.475: 0x0010: 40 54 00 17 00 00 03 01 07 00 00 03 00 01 07 01
Jul 4 14:31:37.479: 0x0020: 01 05 02 02 00 03 05 03 01 01 01
Jul 4 14:31:37.479: Registration Response Transmitted
```

The following example shows what can occasionally occur when a shared secret has been implemented. Some CMs calculate the MD5 Message Integrity Check (MIC) over the length of the registration TLV parameters as well as the length of the shared secret itself, while other CMS calculate the MIC value only over the length of the registration TLV parameters. The Cisco CMTS attempts to verify the MIC and shared secret values using the first technique, and if that fails, the CMTS then uses the second technique to verify the CM.

```
Jun 28 12:17:36.171: Registration request from 00c0.abcd.ef01, SID 58 on Cable5/0/U0
Jun 28 12:17:36.171: Found Network Access TLV
Jun 28 12:17:36.171: Found Class Of Service TLV Block
Jun 28 12:17:36.171: Found Max CPEs TLV
Jun 28 12:17:36.171: Found TFTP Server Provisioned CM Address TLV
Jun 28 12:17:36.171: Found CM-MIC TLV
Jun 28 12:17:36.171: Found CMTS-MIC TLV
Jun 28 12:17:36.171: Found Modem Capabilities TLV
Jun 28 12:17:36.171: Found CM IP Address TLV
Jun 28 12:17:36.171: Computing CMTS-MIC to validate REG-REQ data.
Jun 28 12:17:36.171: CMTS_MIC(rfc2104) failed text + key
Jun 28 12:17:36.171: CMTS_MIC(rfc2104) passed
Jun 28 12:17:36.171: REG-RSP Status : ok (0)
Jun 28 12:17:36.171: Registration Response Transmitted
```

The following example shows typical output when a CM attempts to come online without downloading a DOCSIS configuration file from a TFTP server through the Cisco CMTS cable interface, when the **cable tftp-enforce** command has been used.

```
Router# debug cable interface c3/0 verbose
Router# debug cable registration
CMTS registration debugging is on

May 14 22:33:21.474: Registration request from 5555.7366.12fb, SID 2 on Cable3/0/U0
May 14 22:33:21.474: Cable Modem sent Registration Request without attempting TFTP

SLOT 3/0: May 14 22:33:21.474: %UBR7200-4-REGISTRATION_BEFORE_TFTP:
<133>CMTS[Cisco]:<????????> Registration request unexpected: Cable Modem did not attempt
TFTP. CM Mac Addr <5555.7366.12fb>
Registration failed for Cable Modem 5555.7366.12fb on interface Cable3/0/U0:
               CoS/Sflow/Cfr/PHS failed in REG-REQ
May 14 22:33:21.474: REG-RSP Status : failure (2)
May 14 22:33:21.474: Registration Response:
May 14 22:33:21.474: 0x0000: C2 00 00 1B 00 00 00 50 73 4E B4 19 00 05 00 E0
May 14 22:33:21.474: 0x0010: 56 AC 00 09 00 00 03 01 07 00 00 02 02
May 14 22:33:21.474: Registration Response Transmitted
```

Tip To monitor a specific CM when it comes online, use the **debug cable mac-address, debug cable mac-protocol,** and **debug cable registration** commands.

Related Commands

Command	Description
debug cable interface	Enables debugging output for a specific cable interface.
debug cable mac-protocol	Displays debugging output for the MAC layer protocol.
debug cable mac-scheduler	Displays debugging output for the MAC layer scheduler and admission control activities.
show controllers cable	Displays interface controller information for the specified slot.

debug cable remote-query

To display debug messages for remote modem queries, use the **debug cable remote-query** command in privileged EXEC mode. To stop displaying debug messages, use the **no** form of this command.

debug cable remote-query

no debug cable remote-query

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR, 12.1(2)T	This command was introduced.
12.1(2)EC1	Support for this command was added to the 12.1 EC train.
12.2(4)BC1b	Support for this command was added to the 12.2 BC train.

Examples

The following example shows typical debugging output for a successful poll of the CMs:

```
router# debug cable remote-query
remote-query debugging is on
.
For IP address 209.165.200.223
Nov 10 15:56:50.241: docsIfSignalQualityEntry.5.4 = 380
Nov 10 15:56:50.241: docsIfMibObjects.2.2.1.3.2 = 360
Nov 10 15:56:50.245: docsIfDownstreamChannelEntry.6.4 = -30
Nov 10 15:56:50.245: docsIfUpstreamChannelEntry.6.3 = 12422
Nov 10 15:56:50.249: docsIfSignalQualityEntry.6.4 = 0
Nov 10 15:56:50.477:
```

The following example shows typical debugging output when the waiting queue at the CMTS is empty:

```
SNMP proxy exec got event, but queue is empty
```

The following example shows typical debugging output when you try to modify the polling interval or community string while the polling in is progress:

```
Community string if modified will not be reflected
```

Note The polling interval will be changed but to change the community string, you must unconfigure the **snmp-server community** command and reconfigure it with the new community string.

Related Commands

Command	Description
cable modem remote-query	Enables and configures the remote-query feature to gather CM performance statistics on the CMTS.
show cable modem remote-query	Displays the statistics accumulated by the remote-query feature.
snmp-server enable traps cable	Enables traps that are sent when the remote polling of CMs has been completed.

debug cable reset

To display debugging messages when cable interfaces are reset due to the complete loss of received packets, use the **debug cable reset** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable reset

no debug cable reset

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates display of reset messages from cable interfaces.

Examples

The following shows typical output from the **debug cable reset** command:

```
Router# debug cable reset
CMTS reset debugging is on
Router#

Resetting CMTS interface.  Num SIDs = 32 Ranging count = 10
    Elapsed time: 21 seconds
```

debug cable rfmib

To display debugging messages about when the DOCSIS-IF-MIB MIB is updated, use the **debug cable rfmib** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable rfmib

no debug cable rfmib

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)BC1	This command was introduced.

Usage Guidelines

This command displays debugging messages when the counters in the DOCSIS-IF-MIB are updated.

Note For more information about the information displayed by this command, see RFC 2670 and the DOCS-IF-MIB MIB at the following URL:

http://www.cisco.com/go/mibs

Examples

The following shows typical output from the **debug cable rfmib** command:

```
Router# debug cable rfmib
CMTS RFMIB debugging is on
Router#

CMTS is calculating utilization.

Calculating all utilization data.

Polling and calculating all channels utilization data.

Polling and calculating all channels utilization data.

Cable7/0/0 DS
cur_snmp_ifOutOctets 29282580
cur_snmp_total_bytes 3348841605
last_snmp_ifOutOctets 27327636
last_snmp_total_bytes 3126121018
Cable7/0/0 DS, delta_outoctets 1954944, delta_total_bytes 222720587, chan_utilization 1

Cable7/0/0 US 2
cur_snmp_total_ms 36065552
cur_snmp_ucast_grnt_ms 11287
```

```
cur_snmp_used_cntn_ms 1238
last_snmp_total_ms 33665428
last_snmp_ucast_grnt_ms 10807
last_snmp_used_cntn_ms 1238
Cable7/0/0 US 2 delta_total_ms 2400124, delta_ucast_grnt_ms 480, delta_used_cntn_ms 0,
chan_utilization 0
```

debug cable service-ds-selection

To enable the debugging for downstream selection, use the **debug cable service** command in privileged EXEC mode. To disable debugging for downstream-selection, use the **no** form of the command.

debug cable service-ds-selection {**event**|**timer**}

no debug cable service {**event**|**timer**}

Syntax Description

event	Enables debug messages for Cable downstream selection events.
timer	Enables debug messages for Cable downstream selection timer.

Command Default

No debug messages for downstream selection are enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(23)BC	This command was introduced for the Cisco uBR10012 router.

Usage Guidelines

The **debug cable service** command is intended for use by Cisco Systems technical support personnel.

Examples

The following example shows how to enable debugging for downstream selection events:

```
Router debug cable service-ds-selection event
```

Related Commands

Command	Description
test cable voice	Manually set voice tag of a cable modem to test downstream channel selection for a voice-enabled modem.

debug cable specmgmt

To debug spectrum management (frequency agility) on the HFC network, use the **debug cable specmgmt** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable specmgmt

no debug cable specmgmt

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.
12.0(4)XI	This command was removed and replaced by the d**ebug cable hw-spectrum** command.

Usage Guidelines

This command activates debugging of spectrum management (frequency agility) on the HFC network. When this command is activated, any messages generated due to spectrum group activity will be displayed on the Cisco CMTS console. Spectrum group activity can be additions or changes to spectrum groups, or frequency and power lever changes controlled by spectrum groups.

Examples

The following shows sample output from the **debug cable specmgmt** command:

```
Router# debug cable specmgmt
 CMTS specmgmt debugging is on
Router#
reassign-blind: U0 not present or shutdown
```

debug cable startalloc

To debug channel allocations on the HFC network, use the **debug cable startalloc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable startalloc

no debug cable startalloc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of the channel allocations on the HFC network. When this command is activated, any messages generated when channels are allocated to CMs on the HFC network are displayed on the Cisco CMTS console.

Caution This command should be used for development testing only, not in production setting.

Examples

The following shows sample output from the **debug cable startalloc** command:

```
Router# debug cable startalloc
Router#
 00:53:27: Cable3/U0 MAP startalloc adjusted by 1 mslots
 00:53:27: Cable4/U0 MAP startalloc adjusted by 3 mslots
 00:53:27: Cable3/U0 MAP startalloc adjusted by 5 mslots
 00:53:27: Cable4/U0 MAP startalloc adjusted by 4 mslots
 00:53:28: Cable3/U0 MAP startalloc adjusted by 4 mslots
 00:53:28: Cable4/U0 MAP startalloc adjusted by 5 mslots
```

debug cable subscriber-monitoring

To display enforce-rule debug messages for subscriber traffic management on the Cisco CMTS routers, use the **debug cable subscriber-monitoring** command in privileged EXEC mode. To stop the display of debug messages, use the **no** form of this command.

debug cable subscriber-monitoring

no debug cable subscriber-monitoring

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(15)BC1	This command was introduced.
12.3(9a)BC	This command was integrated into Cisco IOS Release 12.3(9a)BC.
12.2(33)SCA	This command was integrated into Cisco IOS Release 12.2(33)SCA. Support for the Cisco uBR7225VXR router was added.

Usage Guidelines

Because this command can produce a large volume of debug information, use this command only when you have also enabled debugging for a particular interface or MAC address, using the **debug cable interface** and **debug cable mac-address** commands, respectively.

Examples

The following example shows how to enable debugging output using the **debug cable subscriber-monitoring** command:

```
Router# debug cable subscriber-monitoring

subscriber monitoring debugging is on

cmts_enf_map_sm_to_qos: enforced=9, penalty_life_time=10080
cmts_enf_map_sm_to_qos: Found rule #=9, rule_name=name, dir=US
cmts_enf_map_sm_to_registered_qos1: us smp=0x00, ds smp=0x1F
```

Related Commands

Command	Description
cable qos enforce-rule	Creates an enforce-rule to enforce a particular QoS profile for subscriber traffic management and enters enforce-rule configuration mode.
debug cable interface	Enables debugging output for a specific cable interface.

Command	Description
debug cable mac-address	Enables debugging output for the cable modems that match the specified hardware MAC address or range of addresses.
show cable qos enforce-rule	Displays the QoS enforce-rules that are currently defined.

debug cable telco-return

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

debug cable telco-return

no debug cable telco-return

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

This command is supported only in images that support telco-return operation (-t-).

Examples

The following is sample output from the **debug cable telco-return** and **debug cable telco-return msg** commands:

```
Router# debug cable telco-return
Router# debug cable telco-return msg
01:17:31:Sending TCD message:
  TLV type = 1
  TLV len = 56
   Factory default flag: 1
   Phone number 1:       5551212
   Service provider name:uBR7246
   Connection threshold: 10
   Username:             guest
   Password:             password
   DHCP authenticate:    1
   DHCP server:          10.10.255.255
   PPP authentication:   2
   Manual dial:          1
Sending TSI message:
   DS channel IP address:  10.10.10.10
   Registration IP address:10.10.10.10
   CMTS boot time:         3080626752
   DS channel ID:          0
   Epoch:                  1
```

Related Commands

Command	Description
debug cable telco-return msg	Displays the Telephony Channel Descriptor (TCD) and Termination System Information (TSI) messages.

debug cable telco-return msg

To display the Telephony Channel Descriptor (TCD) and Termination System Information (TSI) messages that are sent downstream to the telco-return CMs, use the **debug cable telco-return msg** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable telco-return msg

no debug cable telco-return msg

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

This command is supported only in images that support telco-return operation (-t-).

Examples

The following is sample output from the **debug cable telco-return msg** command:

```
Router# debug cable telco-return
Router# debug cable telco-return msg
01:17:31:Sending TCD message:
  TLV type = 1
  TLV len = 56
   Factory default flag: 1
   Phone number 1:       5551212
   Service provider name:uBR7246
   Connection threshold: 10
   Username:             guest
   Password:             password
   DHCP authenticate:    1
   DHCP server:          10.10.255.255
   PPP authentication:   2
   Manual dial:          1
Sending TSI message:
   DS channel IP address:  10.10.10.10
   Registration IP address:10.10.10.10
   CMTS boot time:         3080626752
   DS channel ID:          0
   Epoch:                  1
```

Related Commands

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

debug cable tlvs

To display the Type/Length/Value encodings (TLVs) parsed by the DOCSIS 1.1 TLV parser/encoder, use the **debug cable tlvs** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cable tlvs

no debug cable tlvs

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(4)CX	This command was introduced.
12.2(4)BC1	This command was supported on the Cisco uBR7100 series and Cisco uBR10012 universal broadband routers.
12.2(33)SCC	This command was integrated into Cisco IOS Release 12.2(33)SCC.

Usage Guidelines

This command displays the TLVs for service flow encodings, classifier encodings, and PHS rules. Do not use this command when you have a large number of active CMs on your network because it could generate a huge amount of output to the console port.

Examples

The following example shows typical output for the **debug cable tlvs** command:

```
Router# debug cable tlvs
CMTS TLV encodings debugging is on

 00:02:06: Registration request from 0003.e350.9b8d, SID 3 on Cable3/0/U0
 00:02:06: TLV-Block Bytes:
 00:02:06: 0x0000: 03 01 01 12 01 10 1D 01 00 16 0F 01 01 01 03 02
 00:02:06: 0x0010: 00 04 09 06 03 04 0A 0A 00 02 18 07 01 02 00 01
 00:02:06: 0x0020: 06 01 07 18 07 01 02 00 02 06 01 07 18 07 01 02
 00:02:06: 0x0030: 00 03 06 01 01 18 07 01 02 00 04 06 01 01 19 07
 00:02:06: 0x0040: 01 02 00 09 06 01 07 19 07 01 02 00 0A 06 01 01
 00:02:06: 0x0050: 19 07 01 02 00 0B 06 01 01 19 07 01 02 00 0C 06
 00:02:06: 0x0060: 01 01 06 10 33 E0 BA 7A DA 81 1B 9B 8E 37 F5 33
 00:02:06: 0x0070: 1C 84 E7 4D 07 10 01 0C C8 DB F9 26 B7 D2 DD 0A
 00:02:06: 0x0080: 00 58 1E 14 15 FD 0C 04 0A 0A 00 02 08 03 00 03
 00:02:06: 0x0090: E3 05 21 02 01 01 03 01 01 04 01 01 05 01 00 06
 00:02:06: 0x00A0: 01 01 07 01 00 08 01 04 09 01 00 0A 01 01 0B 01
 00:02:06: 0x00B0: 08 01 01 01
 00:02:06: Found Network Access TLV
 00:02:06: Ntw Access Control : 1
 00:02:06: Found Max CPEs TLV
 00:02:06: Maximum Number Of CPEs : 16
 00:02:06: Found Privacy Enable TLV
 00:02:06: Privacy Enable : 0
```

```
00:02:06: Found Upstream Packet Classifier TLV
00:02:06:       Classifier Reference : 1
00:02:06:       Service-Flow Reference : 4
00:02:06:       Found IP Packet Classifier Sub-TLV
00:02:06:               Source Address : 10.10.0.2
00:02:06: Found Upstream Service Flow TLV
00:02:06:       Service Flow Reference : 1
00:02:06:       QoS Parameter Set Type : 0x7
00:02:06: Found Upstream Service Flow TLV
00:02:06:       Service Flow Reference : 2
00:02:06:       QoS Parameter Set Type : 0x7
00:02:06: Found Upstream Service Flow TLV
00:02:06:       Service Flow Reference : 3
00:02:06:       QoS Parameter Set Type : 0x1
00:02:06: Found Upstream Service Flow TLV
00:02:06:       Service Flow Reference : 4
00:02:06:       QoS Parameter Set Type : 0x1
00:02:06: Found Downstream Service Flow TLV
00:02:06:       Service Flow Reference : 9
00:02:06:       QoS Parameter Set Type : 0x7
00:02:06: Found Downstream Service Flow TLV
00:02:06:       Service Flow Reference : 10
00:02:06:       QoS Parameter Set Type : 0x1
00:02:06: Found Downstream Service Flow TLV
00:02:06:       Service Flow Reference : 11
00:02:06:       QoS Parameter Set Type : 0x1
00:02:06: Found Downstream Service Flow TLV
00:02:06:       Service Flow Reference : 12
00:02:06:       QoS Parameter Set Type : 0x1
00:02:06: Found CM-MIC TLV
00:02:06: CM MIC:
00:02:06: 0x0000: 33 E0 BA 7A DA 81 1B 9B 8E 37 F5 33 1C 84 E7 4D
00:02:06: Found CMTS-MIC TLV
00:02:06: CMTS MIC:
00:02:06: 0x0000: 01 0C C8 DB F9 26 B7 D2 DD 0A 00 58 1E 14 15 FD
00:02:06: Found CM IP Address TLV
00:02:06: Modem IP Address : 10.10.0.2
00:02:06: Vendor Id:
00:02:06: 0x0000: 00 03 E3
00:02:06: Found Modem Capabilities TLV
00:02:06:       DOCSIS Version : 1
00:02:06:       Fragmentation Support : 1
00:02:06:       Payload Header Suppresion Support : 1
00:02:06:       IGMP Support : 0
00:02:06:       Privacy Support : 1
00:02:06:       Downstream SAID Support : 0
00:02:06:       Upstream SID Support : 4
00:02:06:       Optional Filtering Support : 0
00:02:06:       Tx Equalizer Taps Per Symbol : 1
00:02:06:       Tx Equalizer Taps Support : 8
00:02:06:       Concatenation Support : 1
00:02:06: Performing admission control check
00:02:06: Added Modem Capabilities TLV:
00:02:06: 0x0000: 05 21 02 01 01 03 01 01 04 01 01 05 01 00 06 01
00:02:06: 0x0010: 01 07 01 00 08 01 04 09 01 00 0A 01 01 0B 01 08
00:02:06: 0x0020: 01 01 01
00:02:06: Sfref = 1, SFID = 7
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 18 11 01 02 00 01 02 04 00 00 00 07 03 02 00 03
00:02:06: 0x0010: 06 01 07
00:02:06: Sfref = 2, SFID = 43
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 18 11 01 02 00 02 02 04 00 00 00 2B 03 02 00 0B
00:02:06: 0x0010: 06 01 07
```

```
00:02:06: Sfref = 3, SFID = 44
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 18 0D 01 02 00 03 02 04 00 00 00 2C 06 01 01
00:02:06: Sfref = 4, SFID = 45
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 18 0D 01 02 00 04 02 04 00 00 00 2D 06 01 01
00:02:06: Sfref = 9, SFID = 8
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 19 0D 01 02 00 09 02 04 00 00 00 08 06 01 07
00:02:06: Sfref = 10, SFID = 46
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 19 0D 01 02 00 0A 02 04 00 00 00 2E 06 01 01
00:02:06: Sfref = 11, SFID = 47
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 19 0D 01 02 00 0B 02 04 00 00 00 2F 06 01 01
00:02:06: Sfref = 12, SFID = 48
00:02:06: Added Service Flow Parameters TLV:
00:02:06: 0x0000: 19 0D 01 02 00 0C 02 04 00 00 00 30 06 01 01
00:02:06: Cfr-ref = 1, CFID = 1, SF-ref 4, SFID 45
00:02:06: Added Classifier Parameters TLV:
00:02:06: 0x0000: 16 19 01 01 01 03 02 00 04 02 02 00 01 04 04 00
00:02:06: 0x0010: 00 00 2D 09 06 03 04 0A 0A 00 02
00:02:06: REG-RSP Status : ok (0), REG-ACK required from CM (0)
00:02:06: Reg-Ack wait state successfully created
00:02:06: Registration Response:
00:02:06: 0x0000: C2 00 00 D9 00 00 00 03 E3 50 9B 8D 00 00 00 00
00:02:06: 0x0010: 30 30 00 C7 00 00 03 01 07 00 00 03 00 05 21 02
00:02:06: 0x0020: 01 01 03 01 01 04 01 01 05 01 00 06 01 01 07 01
00:02:06: 0x0030: 00 08 01 04 09 01 00 0A 01 01 0B 01 08 01 01 01
00:02:06: 0x0040: 18 11 01 02 00 01 02 04 00 00 00 07 03 02 00 03
00:02:06: 0x0050: 06 01 07 18 11 01 02 00 02 02 04 00 00 00 2B 03
00:02:06: 0x0060: 02 00 0B 06 01 07 18 0D 01 02 00 03 02 04 00 00
00:02:06: 0x0070: 00 2C 06 01 01 18 0D 01 02 00 04 02 04 00 00 00
00:02:06: 0x0080: 2D 06 01 01 19 0D 01 02 00 09 02 04 00 00 00 08
00:02:06: 0x0090: 06 01 07 19 0D 01 02 00 0A 02 04 00 00 00 2E 06
00:02:06: 0x00A0: 01 01 19 0D 01 02 00 0B 02 04 00 00 00 2F 06 01
00:02:06: 0x00B0: 01 19 0D 01 02 00 0C 02 04 00 00 00 30 06 01 01
00:02:06: 0x00C0: 16 19 01 01 01 03 02 00 04 02 02 00 01 04 04 00
00:02:06: 0x00D0: 00 00 2D 09 06 03 04 0A 0A 00 02
00:02:06: Registration Response Transmitted
00:02:06: Registration acknowledgement from 0003.e350.9b8d, SID 3 on Cable3/0/U0
00:02:06: REG-ACK confirmation code : 0
```

The following is a sample output displaying the TLVs on a Cisco uBR10012 router:

```
Router#debug cable tlvs

CMTS TLV encodings debugging is on
*Jul  7 19:39:28.366: Found Modem Capabilities TLV
*Jul  7 19:39:28.366: Found Upstream Service Flow TLV
*Jul  7 19:39:28.366: Found Downstream Service Flow TLV
*Jul  7 19:39:28.366: Found Privacy Enable TLV
*Jul  7 19:39:28.366: Subscriber Mgmt Filter Group Length: 20
*Jul  7 19:39:28.366: Found Subscriber Mgmt Filter Group Type
*Jul  7 19:39:28.366: Subscriber Mgmt Filter Group Length >= 20
*Jul  7 19:39:28.366: Found Subscriber Mgmt. Control TLV: 35
*Jul  7 19:39:28.366: Found Subscriber Mgmt IP table TLV
*Jul  7 19:39:28.366: Found Vendor Specific Information TLV
Jul  7 19:39:28.394: Found CMTS-MIC TLV
Jul  7 19:39:28.394: Found Network Access TLV
Jul  7 19:39:28.394: Found Max CPEs TLV
Jul  7 19:39:28.394: Found Upstream Service Flow TLV
Jul  7 19:39:28.394: Found Downstream Service Flow TLV
Jul  7 19:39:28.394: Found Privacy Enable TLV
```

```
Jul  7 19:39:28.394: Found CM-MIC TLV
Jul  7 19:39:28.394: Found Modem Capabilities TLV
Jul  7 19:39:28.394: Found CM IP Address TLV
```

Related Commands

debug cable dynsrv	Displays debugging information about DOCSIS 1.1 dynamic service flow messages.
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for MAC-layer information for a specific CM.

debug cable tod

To display debugging for the local time-of-day (ToD) server on the Cisco CMTS, use the **debug cable tod** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable tod

no debug cable tod

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3 NA	This command was introduced.
12.2(11)BC2	Support for this command was added to the Release 12.2 BC train.

Usage Guidelines

This command displays information about the operation of the ToD server that is onboard the Cisco CMTS. Before this command will display any output, you must also use the **debug cable interface** command to enable debugging operations on each interface that has CMs you wish to monitor.

Examples

The following example shows typical output for the **debug cable tod** command:

```
Router# debug cable interface c3/0
Router# debug cable tod
TOD server debugging is on
Router#

000104: Jun  8 05:48:08.783: tod: Received request from 10.1.1.27 (0001.9659.4411) on
Cable3/0
000105: Jun  8 05:48:15.015: tod: Received request from 10.1.1.32 (0030.96f9.65d9) on
Cable3/0
000106: Jun  8 05:48:16.363: tod: Received request from 10.3.3.10 (0001.9659.43fd) on
Cable3/0
000107: Jun  8 05:48:17.335: tod: Received request from 10.2.2.43 (0002.fdfa.0a35) on
Cable3/0
```

For telco-return CMs, the ToD server verifies that the IP address for the CM making the ToD request is on the correct cable interface, so as to prevent IP spoofing. If the IP address and interface do not match, the following message is printed:

```
000104: Jun  8 05:48:08.783: tod: Cant match output i/f for telco return
```

Related Commands

debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for MAC-layer information for a specific CM.

debug cable transmit

To display debug messages for messages that the CMTS transmits on the cable interface, use the **debug cable transmit** command in privileged EXEC mode. To stop displaying debug messages, use the **no** form of this command.

debug cable transmit

no debug cable transmit

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR	This command was introduced.
12.1(1)T, 12.1(2)EC1	This command was disabled.
12.1(11b)EC	This command was removed.

Examples

The following command enables debugging of messages that the CMTS transmits on the cable interface:

```
Router# debug cable transmit
CMTS debug transmit debugging is on
Router#
```

The **debug cable transmit** command was disabled in Cisco IOS Release 12.1 T, 12.1 EC, and later releases, although it still appears in the CLI until Cisco IOS Release 12.1(11b)EC. To monitor the packets transmitted by the CMs, use the **debug ip packet** command instead.

Because the **debug ip packet** command produces a large volume of messages, you must use it together with an access list to restrict the messages to a particular CM or CPE device to avoid affecting system performance. For example, the following commands would set up an access list that monitors all traffic being sent between the hosts with the two specified IP addresses:

```
router# configure terminal
router(config)# access-list 150 permit ip host ip-address-1 host ip-address-2
router(config)# access-list 150 permit ip host ip-address-2 host ip-address-1
router(config)# exit
router# debug ip packet 150 detail
```

Related Commands

Command	Description
debug cable receive	Enables debugging for received packets.
debug ip packet	Enables debugging of IP packets received and transmitted by the router.

debug cable ubg

To enable debugging information for upstream bonding groups, use the **debug cable ubg** command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

debug cable ubg

no debug cable ubg

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced in Cisco IOS Release 12.2(33)SCC.

Examples

The following is a sample output of the **debug cable ubg** command:

```
Router#debug cable ubg

CMTS UBG assignment info debugging is on

Apr 27 02:28:57.140: CM 001a.c3ff.d59e UBG input TCS: 0x000000FF; mtc_cap: 4
Apr 27 02:28:57.140: CM 001a.c3ff.d59e UBG trace: Configure UBG list
Apr 27 02:28:57.140:  UBG 1(4): bitmap 0x0000000F, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 2(4): bitmap 0x000000F0, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 3(1): bitmap 0x00000004, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 4(2): bitmap 0x00000003, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 5(1): bitmap 0x00000001, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 6(1): bitmap 0x00000002, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 7(2): bitmap 0x00000030, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 8(1): bitmap 0x00000010, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 9(1): bitmap 0x00000020, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 65536(1): bitmap 0x00000001, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65537(1): bitmap 0x00000002, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65538(1): bitmap 0x00000004, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65539(1): bitmap 0x00000008, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65540(1): bitmap 0x00000010, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65541(1): bitmap 0x00000020, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65542(1): bitmap 0x00000040, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65543(1): bitmap 0x00000080, attrib 0x20000000
Apr 27 02:28:57.140: CM 001a.c3ff.d59e UBG trace: ubg after MC & mtc cap match
Apr 27 02:28:57.140:  UBG 1(4): bitmap 0x0000000F, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 2(4): bitmap 0x000000F0, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 3(1): bitmap 0x00000004, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 4(2): bitmap 0x00000003, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 5(1): bitmap 0x00000001, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 6(1): bitmap 0x00000002, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 7(2): bitmap 0x00000030, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 8(1): bitmap 0x00000010, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 9(1): bitmap 0x00000020, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 65536(1): bitmap 0x00000001, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65537(1): bitmap 0x00000002, attrib 0x20000000
```

```
Apr 27 02:28:57.140:  UBG 65538(1): bitmap 0x00000004, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65539(1): bitmap 0x00000008, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65540(1): bitmap 0x00000010, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65541(1): bitmap 0x00000020, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65542(1): bitmap 0x00000040, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65543(1): bitmap 0x00000080, attrib 0x20000000
Apr 27 02:28:57.140: CM 001a.c3ff.d59e
SF_REQ: 0x00000000, SF_FOR: 0x00000000,
CM_REQ: 0x00000000, CM_FOR: 0x00000000
Apr 27 02:28:57.140: CM 001a.c3ff.d59e UBG trace: ubg after SF attrib match
Apr 27 02:28:57.140:      UBG trace: input queue
Apr 27 02:28:57.140:  UBG 1(4): bitmap 0x0000000F, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 2(4): bitmap 0x000000F0, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 3(1): bitmap 0x00000004, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 4(2): bitmap 0x00000003, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 5(1): bitmap 0x00000001, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 6(1): bitmap 0x00000002, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 7(2): bitmap 0x00000030, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 8(1): bitmap 0x00000010, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 9(1): bitmap 0x00000020, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 65536(1): bitmap 0x00000001, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65537(1): bitmap 0x00000002, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65538(1): bitmap 0x00000004, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65539(1): bitmap 0x00000008, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65540(1): bitmap 0x00000010, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65541(1): bitmap 0x00000020, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65542(1): bitmap 0x00000040, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65543(1): bitmap 0x00000080, attrib 0x20000000
Apr 27 02:28:57.140: CM 001a.c3ff.d59e UBG trace: ubg after CM attrib match
Apr 27 02:28:57.140:      UBG trace: input queue
Apr 27 02:28:57.140:  UBG 1(4): bitmap 0x0000000F, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 2(4): bitmap 0x000000F0, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 3(1): bitmap 0x00000004, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 4(2): bitmap 0x00000003, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 5(1): bitmap 0x00000001, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 6(1): bitmap 0x00000002, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 7(2): bitmap 0x00000030, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 8(1): bitmap 0x00000010, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 9(1): bitmap 0x00000020, attrib 0xA0000000
Apr 27 02:28:57.140:  UBG 65536(1): bitmap 0x00000001, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65537(1): bitmap 0x00000002, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65538(1): bitmap 0x00000004, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65539(1): bitmap 0x00000008, attrib 0x20000000
Apr 27 02:28:57.140:  UBG 65540(1): bitmap 0x00000010, attrib 0x20000000
Apr 27 02:28:57.160: UBG 65543 bg_bitrate 2560001 avail_bitrate 2560000
Apr 27 02:28:57.160: UBG 65543 Scale Factor 9 Rank 23040000
Apr 27 02:28:57.160: CM 001a.c3ff.d59e Output UBG 7: 0x00000030
Apr 27 02:28:57.160: UBG mgmt: UBG 7 SF Queue
Apr 27 02:28:57.160:  CM 001a.c3ff.d59e SF 22
```

debug cable ucc

To debug upstream channel change (UCC) messages generated when CMs request or are assigned a new channel, use the **debug cable ucc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable ucc

no debug cable ucc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of any upstream channel change (UCC) messages generated when CMs request or are assigned a new channel. When this command is activated, any messages related to upstream channel changes are displayed on the Cisco CMTS console.

Examples

The following is typical output from the **debug cable ucc** command:

```
Router# debug cable ucc
Router#
SID 2 has been registered
Mac Address of CM for UCC
  00:0E:1D:D8:52:16
UCC Message Sent to CM
Changing SID 2 from upstream channel 1 to upstream channel 2
```

Related Commands

Command	Description
debug cable ucd	Enables debugging for UCD messages.

debug cable ucd

To debug upstream channel descriptor (UCD) messages, use the **debug cable ucd** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable ucd

no debug cable ucd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3NA	This command was introduced.

Usage Guidelines

This command activates debugging of any upstream channel descriptor (UCD) messages. UCD messages contain information about upstream channel characteristics and are sent to the CMs on the HFC network. Cable modems that are configured to use enhanced upstream channels use these UCD messages to identify and select an enhanced upstream channel to use. When this command is activated, any messages related to upstream channel descriptors are displayed on the Cisco CMTS console.

Examples

The following is typical output from the **debug cable ucd** command:

```
UCD MESSAGE
-----------
  FRAME HEADER
    FC                        - 0xC2 ==
    MAC_PARM                  - 0x00
    LEN                       - 0xD3
  MAC MANAGEMENT MESSAGE HEADER
    DA                        - 01E0.2F00.0001
    SA                        - 0009.0CEF.3730
    msg LEN                   - C1
    DSAP                      - 0
    SSAP       t               - 0
    control                   - 03
    version                   - 01
    type                      - 02 ==
  US Channel ID               - 1
  Configuration Change Count  - 5
  Mini-Slot Size              - 4
  DS Channel ID               - 1
  Symbol Rate                 - 8
  Frequency                   - 10000000
  Preamble Pattern            - CC CC CC CC CC CC CC CC CC CC CC CC CC CC
 CC 0D 0D
  Burst Descriptor 0
    Interval Usage Code       - 1
    Modulation Type           - 1 == QPSK
```

```
    Differential Encoding     - 2 == OFF
    Preamble Length           - 64
    Preamble Value Offset     - 56
    FEC Error Correction      - 0
    FEC Codeword Length       - 16
    Scrambler Seed            - 0x0152
    Maximum Burst Size        - 2
    Guard Time Size           - 8
    Last Codeword Length      - 1 == FIXED
    Scrambler on/off          - 1 == ON
  Burst Descriptor 1
    Interval Usage Code       - 3
    Modulation Type           - 1 == QPSK
    Differential Encoding     - 2 == OFF
    Preamble Length           - 128
    Preamble Value Offset     - 0
    FEC Error Correction      - 5
    FEC Codeword Length       - 34
    Scrambler Seed            - 0x0152
    Maximum Burst Size        - 0
    Guard Time Size           - 48
    Last Codeword Length      - 1 == FIXED
    Scrambler on/off          - 1 == ON
  Burst Descriptor 2
    Interval Usage Code       - 4
    Modulation Type           - 1 == QPSK
    Differential Encoding     - 2 == OFF
    Preamble Length           - 128
    Preamble Value Offset     - 0
    FEC Error Correction      - 5
    FEC Codeword Length       - 34
    Scrambler Seed            - 0x0152
    Maximum Burst Size        - 0
    Guard Time Size           - 48
    Last Codeword Length      - 1 == FIXED
    Scrambler on/off          - 1 == ON
  Burst Descriptor 3
    Interval Usage Code       - 5
    Modulation Type           - 1 == QPSK
    Differential Encoding     - 2 == OFF
    Preamble Length           - 72
    Preamble Value Offset     - 48
    FEC Error Correction      - 5
    FEC Codeword Length       - 75
    Scrambler Seed            - 0x0152
    Maximum Burst Size        - 0
    Guard Time Size           - 8
    Last Codeword Length      - 1 == FIXED
    Scrambler on/off          - 1 == ON

The UCD MESSAGE is :
0xC2 0x00 0x00 0xD3 0x00 0x00 0x01 0xE0
0x2F 0x00 0x00 0x01 0x00 0x09 0x0C 0xEF
0x37 0x30 0x00 0xC1 0x00 0x00 0x03 0x01
0x02 0x00 0x01 0x05 0x04 0x01 0x01 0x01
0x08 0x02 0x04 0x00 0x98 0x96 0x80 0x03
0x10 0xCC 0xCC 0xCC 0xCC 0xCC 0xCC 0xCC
0xCC 0xCC 0xCC 0xCC 0xCC 0xCC 0xCC 0x0D
0x0D 0x04 0x25 0x01 0x01 0x01 0x01 0x02
0x01 0x02 0x03 0x02 0x00 0x40 0x04 0x02
0x00 0x38 0x05 0x01 0x00 0x06 0x01 0x10
0x07 0x02 0x01 0x52 0x08 0x01 0x02 0x09
0x01 0x08 0x0A 0x01 0x01 0x0B 0x01 0x01
0x04 0x25 0x03 0x01 0x01 0x01 0x02 0x01
```

```
0x02 0x03 0x02 0x00 0x80 0x04 0x02 0x00
0x00 0x05 0x01 0x05 0x06 0x01 0x22 0x07
0x02 0x01 0x52 0x08 0x01 0x00 0x09 0x01
0x30 0x0A 0x01 0x01 0x0B 0x01 0x01 0x04
0x25 0x04 0x01 0x01 0x01 0x02 0x01 0x02
0x03 0x02 0x00 0x80 0x04 0x02 0x00 0x00
0x05 0x01 0x05 0x06 0x01 0x22 0x07 0x02
0x01 0x52 0x08 0x01 0x00 0x09 0x01 0x30
0x0A 0x01 0x01 0x0B 0x01 0x01 0x04 0x25
0x05 0x01 0x01 0x01 0x02 0x01 0x02 0x03
0x02 0x00 0x48 0x04 0x02 0x00 0x30 0x05
0x01 0x05 0x06 0x01 0x4B 0x07 0x02 0x01
0x52 0x08 0x01 0x00 0x09 0x01 0x08 0x0A
0x01 0x01 0x0B 0x01 0x01
```

Related Commands

Command	Description
debug cable ucc	Enables debugging for UCC messages.

debug cable upconverter

To enable hardware debugging of an internal upconverter, use the **debug cable upconverter** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable upconverter

no debug cable upconverter

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(5)EC	This command was introduced for the Cisco uBR7100 series routers.
12.2(11)CY	Support was added for the Cisco uBR10-MC5X20S cable interface line card on the Cisco uBR10012 router.

Usage Guidelines

This command activates hardware debugging of the internal upconverter that is onboard the Cisco uBR7100 series or the Cisco uBR10-MC5X20S cable interface line card on the Cisco uBR10012 router.

Examples

The following command shows how to enable the **debug cable upconverter** output:

```
Router# debug cable upconverter
Upconverter programming debugging is on
Router# config t
Router(config)# int c1/0
Router(config-if)# no cable downstream rf-shutdown
Ack failed from upconverter
Card does not have upconverter?
Error in upconverter!
Router(config-if)#
```

Related Commands

Command	Description
cable downstream rf-power	Configures the desired RF output power on the integrated upconverter.
cable downstream rf-shutdown	Enables or disables the RF output from the integrated upconverter.
show controllers cable	Displays status and configuration information for the cable interface. On Cisco uBR7100 series routers, this includes information about the integrated upconverter.

debug cable us-adm-ctrl

To enable debugging of upstream admission control activity, use the **debug cable us-adm-ctrl** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable us-adm-ctrl

no debug cable us-adm-ctrl

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(4)CX	This command was introduced for the Cisco uBR10012 router.
12.2(4)BC1	Support for this command was added to the Release 12.2 BC train for the Cisco uBR7100 series, Cisco uBR7200 series, and Cisco uBR10012 routers.
12.2(11)CY	Support was added for the Cisco uBR10-MC5X20S cable interface line card on the Cisco uBR10012 router.

Examples

The following command shows how to enable debugging output for upstream admission control:

```
Router# debug cable us-adm-ctrl
CMTS upstream admission control debugging is on

Router# show debug
CMTS:
   CMTS upstream admission control debugging is on
Router#
```

The following are the debug messages that are displayed when the CMTS does not have an admission policy, scheduling instance, or service class for a registering CM:

```
Null service class
Null admission policy
Null scheduler instance
```

The CMTS displays the following debug message when a CM requests a scheduling type that is either not defined, not recognized, or not supported:

```
Undefined scheduling Type
```

Related Commands

Command	Description
cable upstream admission-control	Determines the percentage of overbooking allowed on the upstream channel.
debug cable mac-scheduler	Displays information for the MAC layer's scheduler and admission control activities.

debug cable wbcmts

To enable debugging information for the wideband CMTS, use the **debug cable wbcmts** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable wbcmts [**association** | **bpi** | **cli** | **guardian** | **statistics** | **wbka** | **wcd**]

no debug cable wbcmts [**association** | **bpi** | **cli** | **guardian** | **statistics** | **wbka** | **wcd**]

Syntax Description

association	(Optional) Enables debug messages for RF to fiber-node associations.
bpi	(Optional) Enables debug messages for Baseline Privacy Interface (BPI).
cli	(Optional) Enables debug messages for wideband-related Cisco IOS CLI activity.
guardian	(Optional) Enables debug messages for DOCSIS 3.0 Downstream Channel Bonding operations performed by the modular-host line card.
ha	(Optional) Enables debug messages for wideband high availability (HA).
statistics	(Optional) Enables debug messages for wideband-related statistics.
wbka	(Optional) Enables debug messages for wideband keepalive (WBKA) operations.
wcd	(Optional) Enables debug messages with wideband channel information as conveyed in the Bonded Downstream Channel Descriptor (BDCD).

Command Default

No wideband-related debug messages are enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(21)BC	This command was introduced for the Cisco uBR10012 router.
12.2(33)SCA	This command was integrated into Cisco IOS Release 12.2(33)SCA.

Usage Guidelines

The **debug cable wbcmts** command is intended for use by Cisco technical support personnel.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use **debug** commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support personnel. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased **debug** command processing overhead will affect system use.

If you specify **debug cable wbcmts** with no argument, generic wideband debugging messages are enabled.

Before issuing the **debug cable wbcmts wcd** command to display BDCD information, you must issue the **debug cable interface** command to enable debugging on the cable interface line card configured for DOCSIS 3.0 protocol operations. For example:

```
Router# debug cable interface Cable slot/subslot/port
Router# debug cable wbcmts wcd
```

In the preceding example, *slot/subslot/port* specifies a port on a cable interface line card that has been configured for DOCSIS 3.0 protocol operations with the **modular-host subslot** command. The line card cable interface for which you enable debug messages with **debug cable interface** is the first cable interface that is up. For example, if the line card configured for DOCSIS 3.0 protocol operations is located in slot 7/0, specify interface 7/0/1 if 7/0/0 is down and 7/0/1 is up.

For information on configuring a cable interface line card for DOCSIS 3.0 protocol operations, see the **modular-host subslot** command.

Examples

The following example shows how to enable debug messages for RF to fiber-node associations, and shows sample output:

```
Router# debug cable wbcmts association

WBCMTS Association debugging is on
...
Router(config-fiber-node)# downstream modular-cable 3/0/0 rf-channel 0-1
Router(config-fiber-node)# end
...
Router#
*Nov 7 17:16:08.835: Remove Wideband-Cable3/0/0:0 nb_info 8/1 index 0,chanID 203 bundle 1,
Index 203
*Nov 7 17:16:08.835: Remove Cable8/1/0 wb_info 8/1 wb_index 0, chanID 72, Index 72
*Nov 7 17:16:08.835: %SYS-5-CONFIG_I: Configured from console by console

Router#
```

The following example shows how to enable debug messages for wideband CMTS statistics, and shows sample output.

```
Router# debug cable wbcmts statistics

WBCMTS statistics debugging is on

Router#
Router# show interface cable 8/1/0 service-flow 19 counters
Sfid Packets Bytes PacketDrop Bits/Sec Packet/Sec
19   0       0     0          0        0
Router#
*Nov 7 17:28:09.815: LC SFID=19, SFID=19, Blaze_Index=23 Reading stats for SFID 1 @ index
23 on 3/0/0
```

Related Commands

Command	Description
debug c10k-jacket	Enables debugging information for the Wideband SIP.
debug cable fn	Enables debugging information for cable fiber nodes.
debug hw-module bay	Enables debugging information for a Wideband SPA.

debug cable wbcmts admission-control

To enable debugging of the wideband interface admission control on the Cisco CMTS, use the **debug cable wbcmts admission-control** command. To disable debugging, use the **no** form of this command.

debug cable wbcmts admission-control

no debug cable wbcmts admission-control

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced.

Usage Guidelines

This command enables debugging of the wideband interface.

Examples

The following example shows a sample output of the **debug cable wbcmts admission-control** command.

```
Router> enable
Router# debug cable wbcmts admission-control
Oct  5 15:43:32.230: Wideband-Cable1/0/0:0    NB 6/1/0 app 1, nb cir = 0, total bkt cir =
0
Oct  5 15:43:32.230:  total_cfg_non_ex_pct: 0, prev_bkt_resv: 0
Oct  5 15:43:32.230: total_cfg_ex_pct: 100, total_cfg_non_ex_pct: 0, total_ex_cir_cfg_bps:
72000000, total bkt resv 0
Oct  5 15:43:32.230: Wideband-Cable1/0/0:0    app 1, per_bucket_cfg_excl_bps: 0,
max_non_ex_bps: 0,
total_nonex_resvd_bps: 0, bkt type: 0
```

Related Commands

Command	Description
cable admission-control	Configures the CPU and memory thresholds for the Cisco CMTS router and supporting broadband processing engines (BPEs).
cable admission-control event	Configures and enables admission control event types on the Cisco CMTS router.
cable admission-control ds-bandwidth	Configures admission control downstream bandwidth thresholds on the Cisco CMTS router.
cable admission-control us-bandwidth	Configures admission control upstream bandwidth thresholds on the Cisco CMTS router.

Command	Description
debug cable admission-control	Enables automatic admission control troubleshooting processes on the Cisco CMTS router.
show cable admission-control	Displays the current admission control configuration and status on the Cisco CMTS router or on a specified interface.

debug cable wbcmts resiliency

To enable debugging of the resiliency operation, use the **debug cable wbcmts resiliency** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cable wbcmts resiliency

no debug cable wbcmts resiliency

Syntax Description

This command has no arguments or keywords.

Command Default

No wideband-related debug messages are enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCB	This command was introduced.

Usage Guidelines

The **debug cable wbcmts** command is intended for use by Cisco technical support personnel.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use **debug** commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support personnel. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased **debug** command processing overhead will affect system use.

Related Commands

Command	Description
debug c10k-jacket	Enables debugging information for the Wideband SIP.
debug cable fn	Enables debugging information for cable fiber nodes.
debug hw-module bay	Enables debugging information for a Wideband SPA.
debug cable wbcmts	Enables debugging information for the wideband CMTS.

debug cable-modem bpkm

To display information about Baseline Privacy Interface (BPI) key management, use the **debug cable-modem bpkm** command in privileged EXEC mode. To disable BPI debugging, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem bpkm {**errors** | **events** | **packets**}

no debug cable-modem bpkm {**errors** | **events** | **packets**}

Syntax Description:

errors	Debugs CM privacy errors.
events	Debugs events related to cable baseline privacy.
packets	Debugs baseline privacy packets.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.0(5), 12.0(5)T	This command was removed from normal access.

Usage Guidelines

Baseline privacy key management exchanges take place only when both the router and the CMTS are running code images that support baseline privacy, and the privacy class of service is enabled via the configuration file that is downloaded to the cable access router. Baseline privacy code images for the router contain k1, k8, or k9 in the code image name.

In Cisco IOS Release 12.0(5) and later releases, this command is available only under the direction of TAC or field service, and should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

The following example shows typical debug output when the CMTS does not have privacy enabled:

```
Router# debug cable-modem bpkm errors

cm_bpkm_fsm(): machine: KEK, event/state: EVENT_4_TIMEOUT/STATE_B_AUTH_WAIT, new state:
STATE_B_AUTH_WAIT

cm_bpkm_fsm(): machine: KEK, event/state: EVENT_4_TIMEOUT/STATE_B_AUTH_WAIT, new state:
STATE_B_AUTH_WAIT

%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to down
cm_bpkm_fsm(): machine: KEK, event/state: EVENT_1_PROVISIONED/STATE_A_START, new state:
STATE_B_AUTH_WAIT
```

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to up
Router#
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information.
debug cable-modem error	Displays debugging messages for the cable interface driver.
debug cable-modem interrupts	Displays information about CM interrupts.
debug cable-modem mac log	Displays comprehensive debugging messages for the cable interface MAC layer.
debug cable-modem map	Displays the timing of MAP and sync messages.

debug cable-modem bridge

To display bridge filter processing information, use the **debug cable-modem bridge** command in Privileged EXEC mode. To disable bridge filter debugging, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem bridge

no debug cable-modem bridge

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.0(5), 12.0(5)T	This command was removed from normal access.

Usage Guidelines

When the interface is down, all bridge table entries learned on the Ethernet interface are set to discard because traffic is not bridged until the cable interface has completed initialization. After the interface (the line protocol) is completely up, bridge table entries learned on the Ethernet interface program the cable's MAC data filters. The cable MAC hardware filters out any received packets whose addresses are not in the filters. In this way, the cable interface only receives packets addressed to its own MAC address or an address it has learned on the Ethernet interface.

In Cisco IOS Release 12.0(5) and later releases, this command is available only under the direction of TAC or field service, and should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

The following shows typical output for the **debug cable-modem bridge** command:

```
Router# debug cable-modem bridge
Router#
%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to downshut
cm_tbridge_add_entry(): MAC not initialized, discarding entry: 00e0.fe7a.186fno shut
cm_tbridge_add_entry(): MAC not initialized, discarding entry: 00e0.fe7a.186f
%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to up
cm_tbridge_add_entry(): Adding entry 00e0.fe7a.186f to filter 2
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays Baseline Privacy Interface (BPI) information.
debug cable-modem error	Displays debugging messages for the cable interface driver.
debug cable-modem interrupts	Displays information about CM interrupts.
debug cable-modem mac log	Displays comprehensive debugging messages for the cable interface MAC layer.
debug cable-modem map	Displays the timing of MAP and sync messages.

debug cable-modem error

To display debugging messages for the cable interface driver, use the **debug cable-modem error** command in privileged EXEC mode. To turn off cable interface debugging, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem error

no debug cable-modem error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.0(5), 12.0(5)T	This command was removed from normal access.

Usage Guidelines

This command displays detailed output about the sanity checking of received frame formats, the acquisition of downstream QAM/FEC lock, the receipt or non-receipt of SYNC messages from the CMTS, reception errors, and bandwidth request failures.

In Cisco IOS Release 12.0(5) and later releases, this command is available only under the direction of TAC or field service, and should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

The following shows typical output for the **debug cable-modem error** command:

```
Router# debug cable-modem error
Router#
*Mar  7 20:16:29: AcquireSync(): Update rate is 100 Hz
*Mar  7 20:16:30: 1st Sync acquired after 1100 ms.
*Mar  7 20:16:30: Recovery loop is locked (7/9)
*Mar  7 20:16:30: 2nd Sync acquired after 100 ms.
*Mar  7 20:16:30: Recovery loop is locked (10/15)
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays information about Baseline Privacy Interface (BPI) key management.
debug cable-modem bridge	Displays bridge filter processing information.

Command	Description
debug cable-modem interrupts	Displays information about CM interrupts.
debug cable-modem mac log	Displays comprehensive debugging messages for the cable interface MAC layer.
debug cable-modem map	Displays the timing of MAP and sync messages.

debug cable-modem interrupts

To display information about cable interface interrupts, use the **debug cable-modem interrupts** command in privileged EXEC mode. To turn off cable interface interrupt debugging, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem interrupts

no debug cable-modem interrupts

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.0(5), 12.0(5)T	This command was removed from normal access.

Usage Guidelines

In Cisco IOS Release 12.0(5) and later releases, this command is available only under the direction of TAC or field service, and should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

The following shows typical output for the **debug cable-modem interrupts** command:

```
Router# debug cable-modem interrupts
Router#
*** bcm3220_rx_mac_msg_interrupt ***
*** bcm3220_rx_mac_msg_interrupt ***
### bcm3220_tx_interrupt ###
*** bcm3220_rx_mac_msg_interrupt ***
### bcm3220_tx_interrupt ###
*** bcm3220_rx_mac_msg_interrupt ***
### bcm3220_tx_interrupt ###
### bcm3220_tx_interrupt ###
### bcm3220_tx_interrupt ###
### bcm3220_tx_interrupt ###
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays information about Baseline Privacy Interface (BPI) key management.
debug cable-modem bridge	Displays bridge filter processing information.
debug cable-modem error	Displays debugging messages for the cable interface driver.
debug cable-modem mac log	Displays comprehensive debugging messages for the cable interface MAC layer.
debug cable-modem map	Displays the timing of MAP and sync messages.

debug cable-modem mac log

To display comprehensive debugging messages for the cable interface MAC layer, use the **debug cable-modem mac log** command in privileged EXEC mode. To turn off debugging for the MAC layer, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem mac log [**verbose**]

no debug cable-modem mac log [**verbose**]

Syntax Description

verbose	(Optional) Displays periodic MAC layer events, such as ranging.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.1(3)XL	Support was added for the Cisco uBR905 cable access router.
12.1(5)XU1	Support was added for the Cisco CVA122 Cable Voice Adapter.
12.2(2)XA	Support was added for the Cisco uBR925 cable access router.

Usage Guidelines

The **debug cable-modem mac log** command is one of the most useful debugging tools because MAC-layer log messages are written to a circular log file even when debugging is not turned on. These messages include timestamps, events, and information pertinent to these events. Enter the **debug cable-modem mac log** command to view MAC log messages. If you want to view this information without entering debug mode, use the **show controllers cable-modem** *number* **mac log** command. The same information is displayed by both commands.

Tip The **debug cable-modem mac log** command displays details about the MAC-layer processes. In particular, this command displays the details of the CM's registration and initialization process.

If the router interface fails to come up or resets periodically, the MAC log will show what happened. For example, if an address is not obtained from the DHCP server, an error is logged, initialization starts over, and the CM scans for a downstream frequency. The **debug cable-modem mac log** command displays the log from oldest entry to newest entry.

After initial ranging is successful (dhcp_state has been reached), further RNG-REQ/RNG-RSP messages and watchdog timer entries are suppressed from output unless the **verbose** keyword is used. Note that CMAC_LOG_WATCHDOG_TIMER entries while in the maintenance_state are normal when using the **verbose** keyword.

Note This command should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

This example shows typical output from the **debug cable-modem mac log** command. The fields of the output are the time since bootup, the log message, and in some cases a parameter that gives more detail about the log entry.

```
uBR924# debug cable-modem mac log
Cable Modem mac log debugging is on

*Mar  7 01:42:59: 528302.040 CMAC_LOG_LINK_DOWN
*Mar  7 01:42:59: 528302.042 CMAC_LOG_RESET_FROM_DRIVER
*Mar  7 01:42:59: 528302.044 CMAC_LOG_STATE_CHANGE                      wait_for_link_up_state
*Mar  7 01:42:59: 528302.046 CMAC_LOG_DRIVER_INIT_IDB_SHUTDOWN          0x08098D02
*Mar  7 01:42:59: 528302.048 CMAC_LOG_LINK_DOWN
*Mar  7 01:43:05: 528308.428 CMAC_LOG_DRIVER_INIT_IDB_RESET             0x08098E5E
*Mar  7 01:43:05: 528308.432 CMAC_LOG_LINK_DOWN
*Mar  7 01:43:05: 528308.434 CMAC_LOG_LINK_UP
*Mar  7 01:43:05: 528308.436 CMAC_LOG_STATE_CHANGE                      ds_channel_scanning_state
*Mar  7 01:43:05: 528308.440 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     88/453000000/855000000/6000000
*Mar  7 01:43:05: 528308.444 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     89/93000000/105000000/6000000
*Mar  7 01:43:05: 528308.448 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     90/111250000/117250000/6000000
*Mar  7 01:43:05: 528308.452 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     91/231012500/327012500/6000000
*Mar  7 01:43:05: 528308.456 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     92/333015000/333015000/6000000
*Mar  7 01:43:05: 528308.460 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     93/339012500/399012500/6000000
*Mar  7 01:43:05: 528308.462 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     94/405000000/447000000/6000000
*Mar  7 01:43:05: 528308.466 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     95/123015000/129015000/6000000
*Mar  7 01:43:05: 528308.470 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     96/135012500/135012500/6000000
*Mar  7 01:43:05: 528308.474 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     97/141000000/171000000/6000000
*Mar  7 01:43:05: 528308.478 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     98/219000000/225000000/6000000
*Mar  7 01:43:05: 528308.482 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND     99/177000000/213000000/6000000
*Mar  7 01:43:05: 528308.486 CMAC_LOG_WILL_SEARCH_SAVED_DS_FREQUENCY    663000000
*Mar  7 01:43:05: 528308.488 CMAC_LOG_WILL_SEARCH_USER_DS_FREQUENCY     663000000
*Mar  7 01:43:07: 528310.292 CMAC_LOG_DS_64QAM_LOCK_ACQUIRED            663000000
.
.
.
528383.992 CMAC_LOG_STATE_CHANGE                    registration_state
528384.044 CMAC_LOG_REG_REQ_MSG_QUEUED
528384.050 CMAC_LOG_REG_REQ_TRANSMITTED
528384.052 CMAC_LOG_REG_RSP_MSG_RCVD
528384.078 CMAC_LOG_COS_ASSIGNED_SID                1/4
528384.102 CMAC_LOG_RNG_REQ_QUEUED                  4
528384.102 CMAC_LOG_REGISTRATION_OK
528384.102 CMAC_LOG_STATE_CHANGE                    establish_privacy_state
528384.102 CMAC_LOG_STATE_CHANGE                    maintenance_state
528388.444 CMAC_LOG_RNG_REQ_TRANSMITTED
528388.444 CMAC_LOG_RNG_RSP_MSG_RCVD
528398.514 CMAC_LOG_RNG_REQ_TRANSMITTED
528398.516 CMAC_LOG_RNG_RSP_MSG_RCVD
528408.584 CMAC_LOG_RNG_REQ_TRANSMITTED
528408.586 CMAC_LOG_RNG_RSP_MSG_RCVD
528414.102 CMAC_LOG_WATCHDOG_TIMER
528418.654 CMAC_LOG_RNG_REQ_TRANSMITTED
528418.656 CMAC_LOG_RNG_RSP_MSG_RCVD
528428.726 CMAC_LOG_RNG_REQ_TRANSMITTED
528428.728 CMAC_LOG_RNG_RSP_MSG_RCVD
528438.796 CMAC_LOG_RNG_REQ_TRANSMITTED
```

```
528438.798 CMAC_LOG_RNG_RSP_MSG_RCVD
528444.102 CMAC_LOG_WATCHDOG_TIMER
528444.492 CMAC_LOG_LINK_DOWN
528444.494 CMAC_LOG_RESET_FROM_DRIVER
528444.494 CMAC_LOG_STATE_CHANGE                        wait_for_link_up_state
528444.494 CMAC_LOG_DRIVER_INIT_IDB_SHUTDOWN            0x08098D02
528444.494 CMAC_LOG_LINK_DOWN
528474.494 CMAC_LOG_WATCHDOG_TIMER
528504.494 CMAC_LOG_WATCHDOG_TIMER
528534.494 CMAC_LOG_WATCHDOG_TIMER

0 events dropped due to lack of a chunk
```

Note The line "`0 events dropped due to lack of a chunk`" at the end of the display indicates that no log entries were discarded due to a temporary lack of memory. This means the log is accurate and reliable.

The following example shows typical output of the **debug cable-modem mac log verbose** command. The **verbose** keyword displays periodic events such as ranging.

```
Router# debug cable mac log verbose
Cable Modem mac log debugging is on (verbose)
Router#
574623.810 CMAC_LOG_RNG_REQ_TRANSMITTED
574623.812 CMAC_LOG_RNG_RSP_MSG_RCVD
574627.942 CMAC_LOG_WATCHDOG_TIMER
574633.880 CMAC_LOG_RNG_REQ_TRANSMITTED
574633.884 CMAC_LOG_RNG_RSP_MSG_RCVD
574643.950 CMAC_LOG_RNG_REQ_TRANSMITTED
574643.954 CMAC_LOG_RNG_RSP_MSG_RCVD
574654.022 CMAC_LOG_RNG_REQ_TRANSMITTED
574654.024 CMAC_LOG_RNG_RSP_MSG_RCVD
574657.978 CMAC_LOG_WATCHDOG_TIMER
574664.094 CMAC_LOG_RNG_REQ_TRANSMITTED
574664.096 CMAC_LOG_RNG_RSP_MSG_RCVD
574674.164 CMAC_LOG_RNG_REQ_TRANSMITTED
574674.166 CMAC_LOG_RNG_RSP_MSG_RCVD
```

The following example shows how to disable verbose debugging:

```
Router# no debug cable mac log verbose
Cable Modem mac log debugging is off
Router#
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays information about Baseline Privacy Interface (BPI) key management.
debug cable-modem bridge	Displays bridge filter processing information.
debug cable-modem error	Displays debugging messages for the cable interface driver.
debug cable-modem interrupts	Displays information about CM interrupts.
debug cable-modem mac messages	Displays the MAC-layer messages.
debug cable-modem mac messages dynsrv	Displays the dynamic service MAC-layer messages.
debug cable-modem map	Displays the timing of MAP and sync messages.

debug cable-modem mac messages

To display debugging messages for specific MAC-layer messages, use the **debug cable-modem mac messages** command in privileged EXEC mode. To turn off debugging for the MAC layer, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem mac messages *message-type*

no debug cable-modem mac messages *message-type*

Syntax Description

message-type	Selects the specific type of MAC-layer message type to display: • **dcc-ack**—Displays the Dynamic Channel Change Acknowledge (DCC-ACK) messages received by the CM. • **dcc-req**—Displays the Dynamic Channel Change Request (DCC-REQ) messages received by the CM. • **dcc-rsp**—Displays the Dynamic Channel Change Response (DCC-RSP) messages transmitted by the CM. • **dynsrv**—Displays the Dynamic Service messages transmitted and received by the CM (for more information, see the **debug cable-modem mac messages dynsrv** command). • **map**—Displays the MAP messages received by the CM. • **reg-ack**—Displays the Registration Acknowledge (REG-ACK) messages transmitted by the CM. • **reg-req** —Displays the Registration Request (REG-REQ) messages transmitted by the CM. • **reg-rsp**—Displays the Registration Response (REG-RSP) messages received by the CM. • **rng-req**—Displays the Ranging Request (RNG-REQ) messages transmitted by the CM. • **rng-rsp**—Displays the Ranging Response (RNG-RSP) messages received by the CM. • **sync**—Displays the Time Synchronization (SYNC) messages received by the CM. • **ucc-req**—Displays the Upstream Channel Change Request (UCC-REQ) messages received by the CM. • **ucc-rsp**—Displays the Upstream Channel Change Response (UCC-RSP) messages transmitted by the CM. • **ucd—**Displays the Upstream Channel Descriptor (UCD) messages received by the CM. • **up-dis**—Displays the Upstream Transmitter Disable (UP-DIS) messages received by the CM.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.1(3)XL	Support was added for the Cisco uBR905 cable access router.
12.1(5)XU1	Support was added for the Cisco CVA122 Cable Voice Adapter.
12.2(2)XA	Support was added for the Cisco uBR925 cable access router.
12.2(15)CZ	The **dcc-ack**, **dcc-req**, and **dcc-rsp** options were added to support DOCSIS 1.1 operations.
12.3(2)T	The **up-dis** option was supported.

Usage Guidelines

The output from this command is very verbose, displaying the details of the DOCSIS MAC-layer messages, and is usually not needed for normal interface debugging. The command is most useful when attempting to attach a router to a CMTS that is not DOCSIS-qualified.

Caution This command should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

The following example shows typical output from the **debug cable-modem mac messages** command. A separate command must be given for each message type to be displayed.

Much of the information, such as REG-REQ messages, is displayed in hexadecimal dump format, using the Type/Length/Value (TLV) format required by the DOCSIS specification.

Note For complete descriptions of the MAC-layer management messages, see the DOCSIS Radio Frequency Interface Specification (SP-RFIv1.1-I07-010829 or later), available from CableLabs at http://www.cablemodem.com/

```
Router# debug cable mac messages ucd
ucd message debugging is on
Router# debug cable mac messages map
map message debugging is on
Router# debug cable mac messages rng-rsp
rng-rsp message debugging is on
Router#
*Mar  7 01:44:06:
*Mar  7 01:44:06: UCD MESSAGE
*Mar  7 01:44:06: -----------
*Mar  7 01:44:06:   FRAME HEADER
*Mar  7 01:44:06:     FC                        - 0xC2 == MAC Management
*Mar  7 01:44:06:     MAC_PARM                  - 0x00
*Mar  7 01:44:06:     LEN                       - 0xD3
*Mar  7 01:44:06:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:06:     DA                        - 01E0.2F00.0001
*Mar  7 01:44:06:     SA                        - 00E0.1EA5.BB60
```

```
*Mar  7 01:44:06:     msg LEN                  - C1
*Mar  7 01:44:06:     DSAP                     - 0
*Mar  7 01:44:06:     SSAP                     - 0
*Mar  7 01:44:06:     control                  - 03
*Mar  7 01:44:06:     version                  - 01
*Mar  7 01:44:06:     type                     - 02 == UCD
*Mar  7 01:44:06:     RSVD                     - 0
*Mar  7 01:44:06:   US Channel ID              - 1
*Mar  7 01:44:06:   Configuration Change Count  - 4
*Mar  7 01:44:06:   Mini-Slot Size             - 8
*Mar  7 01:44:06:   DS Channel ID              - 1
*Mar  7 01:44:06:   Symbol Rate                - 8
*Mar  7 01:44:06:   Frequency                  - 20000000
*Mar  7 01:44:06:   Preamble Pattern           - CC CC CC CC CC CC CC CC CC CC CC 0D 0D
*Mar  7 01:44:06:   Burst Descriptor 0
*Mar  7 01:44:06:     Interval Usage Code      - 1
*Mar  7 01:44:06:     Modulation Type          - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding    - 2 == OFF
*Mar  7 01:44:06:     Preamble Length          - 64
*Mar  7 01:44:06:     Preamble Value Offset    - 56
*Mar  7 01:44:06:     FEC Error Correction     - 0
*Mar  7 01:44:06:     FEC Codeword Info Bytes  - 16
*Mar  7 01:44:06:     Scrambler Seed           - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size       - 1
*Mar  7 01:44:06:     Guard Time Size          - 8
*Mar  7 01:44:06:     Last Codeword Length     - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off         - 1 == ON
*Mar  7 01:44:06:   Burst Descriptor 1
*Mar  7 01:44:06:     Interval Usage Code      - 3
*Mar  7 01:44:06:     Modulation Type          - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding    - 2 == OFF
*Mar  7 01:44:06:     Preamble Length          - 128
*Mar  7 01:44:06:     Preamble Value Offset    - 0
*Mar  7 01:44:06:     FEC Error Correction     - 5
*Mar  7 01:44:06:     FEC Codeword Info Bytes  - 34
*Mar  7 01:44:06:     Scrambler Seed           - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size       - 0
*Mar  7 01:44:06:     Guard Time Size          - 48
*Mar  7 01:44:06:     Last Codeword Length     - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off         - 1 == ON
*Mar  7 01:44:06:   Burst Descriptor 2
*Mar  7 01:44:06:     Interval Usage Code      - 4
*Mar  7 01:44:06:     Modulation Type          - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding    - 2 == OFF
*Mar  7 01:44:06:     Preamble Length          - 128
*Mar  7 01:44:06:     Preamble Value Offset    - 0
*Mar  7 01:44:06:     FEC Error Correction     - 5
*Mar  7 01:44:06:     FEC Codeword Info Bytes  - 34
*Mar  7 01:44:06:     Scrambler Seed           - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size       - 0
*Mar  7 01:44:06:     Guard Time Size          - 48
*Mar  7 01:44:06:     Last Codeword Length     - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off         - 1 == ON
*Mar  7 01:44:06:   Burst Descriptor 3
*Mar  7 01:44:06:     Interval Usage Code      - 5
*Mar  7 01:44:06:     Modulation Type          - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding    - 2 == OFF
*Mar  7 01:44:06:     Preamble Length          - 72
*Mar  7 01:44:06:     Preamble Value Offset    - 48
*Mar  7 01:44:06:     FEC Error Correction     - 5
*Mar  7 01:44:06:     FEC Codeword Info Bytes  - 75
*Mar  7 01:44:06:     Scrambler Seed           - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size       - 0
*Mar  7 01:44:06:     Guard Time Size          - 8
```

```
*Mar  7 01:44:06:     Last Codeword Length      - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off          - 1 == ON
*Mar  7 01:44:06:
*Mar  7 01:44:06:
*Mar  7 01:44:06: MAP MESSAGE
*Mar  7 01:44:06: -----------
*Mar  7 01:44:06:   FRAME HEADER
*Mar  7 01:44:06:     FC                        - 0xC3 == MAC Mement with Extended Header
*Mar  7 01:44:06:     MAC_PARM                  - 0x02
*Mar  7 01:44:06:     LEN                       - 0x42
*Mar  7 01:44:06:     EHDR                      - 0x00 0x00
*Mar  7 01:44:06:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:06:     DA                        - 01E0.2F00.0001
.
*Mar  7 01:44:17: RNG-RSP MESSAGE
*Mar  7 01:44:17: ---------------
*Mar  7 01:44:17:   FRAME HEADER
*Mar  7 01:44:17:     FC                        - 0xC2 == MAC Management
*Mar  7 01:44:17:     MAC_PARM                  - 0x00
*Mar  7 01:44:17:     LEN                       - 0x2B
*Mar  7 01:44:17:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:17:     DA                        - 00F0.1EB2.BB61
.
*Mar  7 01:44:20: REG-REQ MESSAGE
*Mar  7 01:44:20: ---------------
*Mar  7 01:44:20: C20000A5 000000E0  1EA5BB60 00F01EB2
*Mar  7 01:44:20: BB610093 00000301  06000004 03010104
*Mar  7 01:44:20: 1F010101 0204003D  09000304 001E8480
*Mar  7 01:44:20: 04010705 04000186  A0060200 0C070101
*Mar  7 01:44:20: 080300F0 1E112A01  04000000 0A020400
*Mar  7 01:44:20: 00000A03 04000002  58040400 00000105
*Mar  7 01:44:20: 04000000 01060400  00025807 04000000
*Mar  7 01:44:20: 3C2B0563 6973636F  06105E4F C908C655
*Mar  7 01:44:20: 61086FD5 5C9D756F  7B730710 434D5453
*Mar  7 01:44:20: 204D4943 202D2D2D  2D2D2D2D 0C040000
*Mar  7 01:44:20: 00000503 010100
*Mar  7 01:44:20:
*Mar  7 01:44:20:
*Mar  7 01:44:20: REG-RSP MESSAGE
*Mar  7 01:44:20: ---------------
*Mar  7 01:44:20:   FRAME HEADER
*Mar  7 01:44:20:     FC                        - 0xC2 == MAC Management
*Mar  7 01:44:20:     MAC_PARM                  - 0x00
*Mar  7 01:44:20:     LEN                       - 0x29
*Mar  7 01:44:20:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:20:     DA                        - 00F0.1EB2.BB61
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays information about Baseline Privacy Interface (BPI) key management.
debug cable-modem bridge	Displays bridge filter processing information.
debug cable-modem error	Displays debugging messages for the cable interface driver.
debug cable-modem interrupts	Displays information about CM interrupts.
debug cable-modem map	Displays the timing of MAP and sync messages.

Command	Description
debug cable-modem mac messages dynsrv	Displays the dynamic service messages.
debug cable-modem up-dis	Displays debugging for Upstream Transmitter Disable (UP-DIS) messages.

debug cable-modem mac messages dynsrv

To display debug messages for the dynamic service MAC-layer messages that are generated when voice calls are made using the dynamic SID feature, use the **debug cable-modem mac messages dynsrv** command in privileged EXEC mode. To turn off debugging for the dynamic service MAC-layer messages, use the **no** form of this command.

Cisco uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem mac messages dynsrv

no debug cable-modem mac messages dynsrv

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XR and 12.1(1)T	This command was introduced for the Cisco uBR924 cable access router.
12.1(3)XL	Support was added for the Cisco uBR905 cable access router.
12.1(5)XU1	Support was added for the Cisco CVA122 Cable Voice Adapter.
12.2(2)XA	Support was added for the Cisco uBR925 cable access router.

Usage Guidelines

This command begins the display of debug messages that show the dynamic service MAC messages that are generated when a voice call is made using the dynamic SID feature. Dynamic SIDs use the following DOCSIS MAC-layer messages to create a new SID when a voice call is made and to delete it when the call is over:

- DSA-REQ—Dynamic Service Addition Request, sent to establish a new service flow.
- DSA-RSP—Dynamic Service Addition Response, sent in reply to DSA-REQ to confirm or deny the new service flow.
- DSA-ACK—Dynamic Service Addition Acknowledge, sent in reply to DSA-RSP to acknowledge the creation of the service flow.
- DSD-REQ—Dynamic Service Deletion Request, sent to delete an existing service flow when it is no longer needed (for example, when a voice call has terminated).
- DSD-RSP—Dynamic Service Deletion Response, sent in response to DSD-REQ to delete an existing service flow.

Note Dynamic Services are described in the DOCSIS 1.1 specification (SP-RFIv1.1-I03-991105 or later revision), available from CableLabs at http://www.cablemodem.com/

Examples

The following example shows how to enable the display of debug messages related to dynamic service operations:

```
Router# debug cable-modem mac messages dynsrv
Router#
```

The following example shows how to turn off the display of debug messages related to dynamic service operations:

```
Router# no debug cable-modem mac messages dynsrv
Router#
```

The following example shows the types of debug messages that are displayed when a voice call is made. This example shows that dynamic SID 52 is created for this particular call.

```
DSA-REQ TLV's:
--------------
US Flow Scheduler(24):
 Unsolicited Grant Size               - 19:2:89
 Nominal Grant Interval               - 20:4:20000
Created New Dynamic Service State, Transaction_id = 3

DSA-REQ MESSAGE TLVS
--------------------
C2000026 00010010  07DF6854 00507366
23270014 00000301  0F000003 180A1302
00591404 00004E20

   597.721 CMAC_LOG_DSA_REQ_MESSAGE_EVENT

DSA-REQ MESSAGE
---------------
  FRAME HEADER
    FC                         - 0xC2 == MAC Management
    MAC_PARM                   - 0x00
    LEN                        - 0x26
  MAC MANAGEMENT MESSAGE HEADER
    DA                         - 0010.abcd.ef00
    SA                         - 0050.abcd.ef00
    msg LEN                    - 14
    DSAP                       - 0
    SSAP                       - 0
    control                    - 03
    version                    - 01
    type                       - 0F == DSA-REQ
    RSVD                       - 0
  Transaction ID               - 3

   597.725 CMAC_LOG_DSA_RSP_MSG_RCVD

DSA-RSP MESSAGE
---------------
  FRAME HEADER
    FC                         - 0xC2 == MAC Management
    MAC_PARM                   - 0x00
    LEN                        - 0x26
  MAC MANAGEMENT MESSAGE HEADER
    DA                         - 0050.abcd.ef00
    SA                         - 0010.abcd.ef00
    msg LEN                    - 14
    DSAP                       - 0
    SSAP                       - 0
    control                    - 03
    version                    - 01
```

```
    type                        - 10 == DSA-RSP
    RSVD                        - 0
  Transaction ID                - 3
  Response                      - 0 == DSA-RSP-OK
  SID                           - 52

Adding sid = 52 to sid_index = 1
   597.729 CMAC_LOG_QOS_ADD_FLOW_SID                   52
```

Related Commands

Command	Description
debug cable-modem mac log	Displays debug messages for other types of MAC-layer messages, including MAP messages, upstream request messages, and sync messages.
show controllers cable-modem qos	Displays detailed information about the QoS configuration for the router.

debug cable-modem map

To display the timing of MAP messages to sync messages, and the timing between MAP messages, use the **debug cable-modem map** command in privileged EXEC mode. To turn off debugging of MAP messages, use the **no** form of this command.

Cisco uBR904, uBR905, uBR924, uBR925 cable access routers, Cisco CVA122 Cable Voice Adapter

debug cable-modem map

no debug cable-modem map

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(4)NA	This command was introduced for the Cisco uBR904 cable access router.
12.0(4)XI1	Support was added for the Cisco uBR924 cable access router.
12.0(5), 12.0(5)T	This command was removed from normal access.

Usage Guidelines

In Cisco IOS Release 12.0(5) and later releases, this command is available only under the direction of TAC or field service, and should be used only while debugging CM operation. Displaying debugging messages consumes system resources, and turning on too many messages could negatively affect system performance.

Examples

The following shows typical output from the **debug cable map** command:

```
Router# debug cable-modem map
Cable Modem MAP debugging is on
Router#
*Mar  7 20:12:08: 595322.942: Min MAP to sync=72
*Mar  7 20:12:08: 595322.944: Max map to map time is 40
*Mar  7 20:12:08: 595322.982: Min MAP to sync=63
*Mar  7 20:12:08: 595323.110: Max map to map time is 41
*Mar  7 20:12:08: 595323.262: Min MAP to sync=59
*Mar  7 20:12:08: 595323.440: Max map to map time is 46
*Mar  7 20:12:09: 595323.872: Min MAP to sync=58
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays information about Baseline Privacy Interface (BPI) key management.
debug cable-modem bridge	Displays bridge filter processing information.
debug cable-modem error	Displays debugging messages for the cable interface driver.

Command	Description
debug cable-modem interrupts	Displays information about cable interface interrupts.
debug cable-modem mac log	Displays comprehensive debugging messages for the cable interface MAC layer.

debug cable-modem up-dis

To display information about DOCSIS 1.1 Upstream Transmitter Disable (UP-DIS) messages received by the cable modem, use the **debug cable-modem up-dis** command in Privileged EXEC mode. To disable debugging output for UP-DIS messages, use the **no** form of this command.

debug cable-modem up-dis

no debug cable-modem up-dis

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.

Usage Guidelines

This command shows debugging messages about the Upstream Transmitter Disable (UP-DIS) messages that the CMTS sends to a CM. The CMTS can send an UP-DIS message to a cable modem to suspend its operation on the cable network. The UP-DIS message can instruct the cable modem to suspend its operations for a limited period of time or permanently.

After sending the UP-DIS message, the CMTS allows the cable modem back on the network only if the cable modem is power-cycled, or if the CMTS sends another UP-DIS message re-enabling the cable modem's upstream transmitter. If a cable modem is already in the suspended state, it must ignore any further UP-DIS messages except a UP-DIS message re-enabling its operations.

Note See the DOCSIS 1.1 specification (revision SP-RFIv1.1-I09-020830 and above) for information on the content and format of the UP-DIS messages.

Examples

The following examples show typical output for the **debug cable up-dis** command, which displays the MAC frame header and UP-DIS Timeout Interval value (in milliseconds):

```
Router# debug cable up-dis
up-dis message debugging is on

Router#

UP-DIS MESSAGE");
--------------");
*Mar  7 01:44:06:   FRAME HEADER
*Mar  7 01:44:06:     FC                          - 0xC2 == MAC Management
*Mar  7 01:44:06:     MAC_PARM                    - 0x00
```

```
*Mar  7 01:44:06:     LEN                       - 0xD3
*Mar  7 01:44:06:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:06:     DA                        - 01E0.2F00.0001
*Mar  7 01:44:06:     SA                        - 00E0.1EA5.BB60
*Mar  7 01:44:06:     msg LEN                   - 10
*Mar  7 01:44:06:     DSAP                      - 0
*Mar  7 01:44:06:     SSAP                      - 0
*Mar  7 01:44:06:     control                   - 03
*Mar  7 01:44:06:     version                   - 01
*Mar  7 01:44:06:     type                      - 28 == UP-DIS
*Mar  7 01:44:06:     RSVD                      - 0

*Mar  7 01:44:06:  Timer Value - 20000
```

Related Commands

debug cable-modem mac messages	Displays debugging messages for specific MAC-layer messages.

debug checkpoint

To enable debugging of the Checkpointing Facility (CF) subsystem on the Cisco uBR10012 router, use the **debug checkpoint** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug checkpoint [**all** | **errors** | **messages** | **temporary** | **timers**]

no debug checkpoint [**all** | **errors** | **messages** | **temporary** | **timers**]

Syntax Description

all	(Optional) Enables all checkpoint debugging types.
errors	(Optional) Enables debugging of any checkpoint errors that occur.
messages	(Optional) Enables debugging for the messages that are sent during checkpoint operations.
temporary	(Optional) Enables basic checkpoint debugging (default).
timers	(Optional) Enables debugging of the checkpoint timers.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)BC3	This command was introduced for the Cisco uBR10012 universal broadband router.

Usage Guidelines

The **debug checkpoint** command enables debugging of the Checkpoint Facility (CF) subsystem, which passes messages from the Active processor to the Standby processor. It also handles sequencing and throttling, as needed, during redundancy operations.

Examples

The following example shows how to enable debugging messages for the CF subsystem:

```
Router# debug checkpoint

Router#
```

Related Commands

Command	Description
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for a specific CM, as identified by its hardware (MAC) address.

debug cmts ipc-cable base

To debug the base layer cable inter-processor communication (IPC) software, use the **debug cmts ipc-cable base** command from the Privileged EXEC mode. To disable the debug flag, use the **no** form of the command.

debug cmts ipc-cable base {**client** {**all** | **id** *cid*} |

entity client-id *cid* **entity-id** *eid* |

service |

session client-id *cid* **entity-id** *eid* **slot** *slot* **subslot** *subslot*}

no debug cmts ipc-cable base {**client** {**all** | **id** *cid*} |

entity client-id *cid* **entity-id** *eid* |

service |

session client-id *cid* **entity-id** *eid* **slot** *slot* **subslot** *subslot*}

Syntax Description

client	Displays all the clients or the specified client-related IPC information
entity	Displays the IPC information of the entity.
service	Displays the IPC information related to service.
session	Displays the IPC information related to this session.
all	Displays the IPC information for all clients.
client-id *cid*	Specifies the client ID of the cable IPC layer. The valid client ID range is from 0 to 8.
entity-id *eid*	Specifies the entity ID od the cable IPC layer. The valid entity ID range is from 0 and 2.
slot *slot*	Specifies the cable interface slot. The valid range is from 5 to 8.
subslot *subslot*	Specifies the cable interface subslot. The valid values are 0 and 1.

Command Default

Debug is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(33)SCF	This command was introduced.

Usage Guidelines

Each debug log has a debug-enable flag that is set by the **debug cmts ipc-cable** command. Each cable IPC debug log has a severity level. The **logging cmts ipc-cable** command allows you to selectively enable cable IPC debug logs based on the log severities. A debug log is recorded only if its severity is equal to or higher than the configured log-level, and if its debug enable flag is set.

Examples

The following example shows the IPC base layers service debugging message:

```
Router(config)# logging cmts ipc-cable log-level debugging
Router# debug cmt ipc-cable base service
Router# hw-module subslot 8/0 reset


Proceed with reset? [confirm]
*Feb 11 05:24:44.101: CLNT DOCSIS C8/0 is down for apps
*Feb 11 05:24:44.101: CLNT HCCP C8/0 is down for apps
*Feb 11 05:24:44.101: CLNT PKTCBL C8/0 is down for apps
*Feb 11 05:24:44.101: CLNT PLFM C8/0 is down for apps
*Feb 11 05:24:44.101: CLNT SNMP C8/0 is down for apps
*Feb 11 05:24:44.101: CLNT GUARDIAN C8/0 is down for apps
*Feb 11 05:24:44.101: CLNT TEST C8/0 is down for apps
*Feb 11 05:24:44.101: Default IPC service 8/0 deleted
*Feb 11 05:24:44.101: Remote IPC service 8/0 svc_type 2 closed
*Feb 11 05:24:44.101: Remote IPC service 8/0 svc_type 3 closed
*Feb 11 05:24:44.101: Remote IPC service 8/0 svc_type 1 closed
```

The following example shows the IPC base layer all clients debugging message:

Note When client debugging is enabled, all entity and session debugging messages are also enabled.

```
Router(config)# logging cmts ipc-cable log-level debugging
Router# debug cmts ipc-cable base client all

*Feb 11 06:00:19.345: CLNT DOCSIS C8/1 is down for apps
*Feb 11 06:00:19.345: CLNT HCCP C8/1 is down for apps
*Feb 11 06:00:19.345: CLNT PKTCBL C8/1 is down for apps
*Feb 11 06:00:19.345: CLNT PLFM C8/1 is down for apps
*Feb 11 06:00:19.345: CLNT SNMP C8/1 is down for apps
*Feb 11 06:00:19.345: CLNT GUARDIAN C8/1 is down for apps
*Feb 11 06:00:19.345: CLNT TEST C8/1 is down for apps
*Feb 11 06:00:19.345: cr10k_ipc_flush_req_xmt_queue -> Flush the packets in xmt queue 8/1
*Feb 11 06:00:19.345: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 0 has removed
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 1 has removed
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 2 has removed
```

```
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 3 has removed
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 4 has removed
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 5 has removed
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 6 has removed
*Feb 11 06:00:19.349: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC

*Feb 11 06:00:19.349: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:00:19.349: Session 8/1 for client 7 has removed
*Feb 11 06:00:19.349: cr10k_ipc_flush_req_xmt_queue -> Flush the packets in xmt queue 8/1
*Feb 11 06:00:19.349: cr10k_ipc_flush_req_xmt_queue -> Flush the packets in xmt queue 8/1
*Feb 11 06:00:19.349: cr10k_ipc_flush_req_xmt_queue -> Flush the packets in xmt queue 8/1
*Feb 11 06:00:19.349: %IPCOIR-3-TIMEOUT: Timeout waiting for a response from slot 8/1.
*Feb 11 06:00:19.349: %IPCOIR-2-CARD_UP_DOWN: Card in slot 8/1 is down.  Notifying
5cable-mc520h-d driver.
```

The following example shows the IPC base layer DOCSIS client and route processor-line card (RP-CLC) entity debugging message:

Note When entity debugging is enabled, session debugging messages are also enabled.

```
Router# debug cmts ipc-cable base entity client-id 0 entity-id 0

*Feb 11 06:05:48.961: CLNT DOCSIS C5/1 is down for apps
*Feb 11 06:05:48.961: CLNT HCCP C5/1 is down for apps
*Feb 11 06:05:48.961: CLNT PKTCBL C5/1 is down for apps
*Feb 11 06:05:48.961: CLNT PLFM C5/1 is down for apps
*Feb 11 06:05:48.961: CLNT SNMP C5/1 is down for apps
*Feb 11 06:05:48.961: CLNT GUARDIAN C5/1 is down for apps
*Feb 11 06:05:48.961: CLNT TEST C5/1 is down for apps
*Feb 11 06:05:48.961: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 5/1, RP-CLC
*Feb 11 06:05:48.961: cr10k_ipc_remove_session -> Start calling get session instance for
slot 5/1
*Feb 11 06:05:48.961: Session 5/1 for client 0 has removed
*Feb 11 06:05:48.965: %IPCOIR-3-TIMEOUT: Timeout waiting for a response from slot 5/1.
*Feb 11 06:05:48.965: %IPCOIR-2-CARD_UP_DOWN: Card in slot 5/1 is down.  Notifying
5cable-mc520h-d driver.
```

The following example shows the IPC base layer session debugging message:

```
Router# debug cmts ipc-cable base session client 0 entity 0 slot 8 subslot 1

*Feb 11 06:11:25.809: CLNT DOCSIS C8/1 is down for apps
*Feb 11 06:11:25.809: CLNT HCCP C8/1 is down for apps
*Feb 11 06:11:25.809: CLNT PKTCBL C8/1 is down for apps
*Feb 11 06:11:25.809: CLNT PLFM C8/1 is down for apps
*Feb 11 06:11:25.809: CLNT SNMP C8/1 is down for apps
*Feb 11 06:11:25.809: CLNT GUARDIAN C8/1 is down for apps
*Feb 11 06:11:25.809: CLNT TEST C8/1 is down for apps
*Feb 11 06:11:25.813: cr10k_ipc_notify_close_conn_event -> Notifying client of IPC Port
CLOSED for Slot 8/1, RP-CLC
*Feb 11 06:11:25.813: cr10k_ipc_remove_session -> Start calling get session instance for
slot 8/1
*Feb 11 06:11:25.813: %IPCOIR-3-TIMEOUT: Timeout waiting for a response from slot 8/1.
plfm-10k-4#
*Feb 11 06:11:25.813: %IPCOIR-2-CARD_UP_DOWN: Card in slot 8/1 is down.  Notifying
5cable-mc520h-d driver.
```

Related Commands

logging cmts ipc-cable	Enables debug logging for the cable IPC software.
show cmts ipc-cable	Displays information about the cable IPC layer on the Cisco CMTS routers.

debug cmts ipc-cable client

To debug the client cable inter-processor communication (IPC) software, use the **debug cmts ipc-cable client** command from the Privileged EXEC mode. To disable the debug flag, use the **no** form of the command.

debug cmts ipc-cable client {**all** |

client client-id *cid*

entity client-id *cid* **entity-id** *eid* |

session client-id *cid* **entity-id** *eid* **slot** *slot* **subslot** *subslot* |

message client-id *cid* **entity-id** *eid* **message-type** *mtype* |

packing client-id *cid* **entity-id** *eid* **packing-type** *ptype*}

no debug cmts ipc-cable client {**all** |

client client-id *cid*

entity client-id *cid* **entity-id** *eid* |

session client-id *cid* **entity-id** *eid* **slot** *slot* **subslot** *subslot* |

message client-id *cid* **entity-id** *eid* **message-type** *mtype* |

packing client-id *cid* **entity-id** *eid* **packing-type** *ptype*}

Syntax Description

all	Displays all information about IPC for all the clients including base layer IPC information.
client	Displays the specified client-related IPC information inlcuding base layer IPC information.
entity	Displays the IPC information of the entity, including base layer IPC information.
session	Displays the IPC information related to the specified session, inlcuding base layer IPC information.
message	Displays the IPC information related to the particular message type.
packing	Displays the IPC information related to this type of packing message.
client-id *cid*	Specifies the client ID of the cable IPC layer. The valid client ID range is from 0 to 8.
entity-id *eid*	Specifies the entity ID od the cable IPC layer. The valid entity ID range is from 0 and 2.
slot *slot*	Specifies the cable interface slot. The valid range is from 5 to 8.
subslot *subslot*	Specifies the cable interface subslot. The valid values are 0 and 1.
message type *mtype*	Specifies the message-type. The valid value is decided by the client and entity.
packaging type *ptype*	Specifies the packing-type. The valid values are 0 and 1.

Command Default Debug is disabled.

Command Modes Privileged EXEC

Command History

Release	Modification
12.2(33)SCF	This command was introduced.

Usage Guidelines Each debug log has a debug-enable flag that is set by the **debug cmts ipc-cable** command. Each cable IPC debug log has a severity level. The **logging cmts ipc-cable** command allows you to selectively enable cable IPC debug logs based on the log severities. A debug log is recorded only if its severity is equal to or higher than the configured log-level, and if its debug enable flag is set.

Examples The following example shows the client PLFM entity of the route processor-cable line card (RP-CLC) session in the logging message:

```
Router# debug cmts ipc-cable client session client-id 4 entity-id 0 slot 5 subslot 0
Router# show cable modem

*Feb 11 07:33:50.628: cr10k_plfm_rsv_buf_guts(): Start C5/0 T=17 SVC=0 L=1612
*Feb 11 07:33:50.628: cr10k_plfm_rsv_buf_guts(): Done
*Feb 11 07:33:50.628: cr10k_plfm_get_unit_ptr(): start B=0x128862D4 Bv=1 Br=0)
*Feb 11 07:33:50.628: cr10k_plfm_get_unit_ptr(): done(P=0xC0E2ACD0)
*Feb 11 07:33:50.628: cr10k_plfm_tx_req_w_rsp(): Start C5/0 T1=17
*Feb 11 07:33:50.628: cr10k_plfm_xform(): Start C5/0 T1=17 Tx=1
*Feb 11 07:33:50.628: cr10k_plfm_get_unit_ptr(): start B=0x128862D4 Bv=1 Br=0)
*Feb 11 07:33:50.628: cr10k_plfm_get_unit_ptr(): done(P=0xC0E2ACD0)
*Feb 11 07:33:50.628: cr10k_plfm_issu_xform_guts() C5/0 T17: p=0xC0E2ACD0
*Feb 11 07:33:50.628: cr10k_plfm_issu_xform_guts() C5/0 T17: xmit xform ok
*Feb 11 07:33:50.628: cr10k_plfm_get_unit_ptr(): start B=0x128862D4 Bv=1 Br=0)
*Feb 11 07:33:50.628: cr10k_plfm_get_unit_ptr(): done(P=0x0)
*Feb 11 07:33:50.628: cr10k_ipc_send_rpc_message_blocked -> Sending blocking message for
client 4 session 5/0
*Feb 11 07:33:50.628: cr10k_message_dump_msg_buffer -> dump buffer size = 1668, client_id
= 4
*Feb 11 07:33:50.628: Dump before RPC blocking send:
*Feb 11 07:33:50.628: 0x0000: 00 00 28 0C 04 03 02 02 06 6C 00 18 05 00 05 00
*Feb 11 07:33:50.628: 0x0010: 52 50 00 01 00 01 00 00 00 0C 80 04 06 64 00 08
*Feb 11 07:33:50.628: 0x0020: 00 0C 06 58 00 01 00 00 00 00 00 00 00 0C 06 4C
*Feb 11 07:33:50.628: 0x0030: 00 00 00 00 00 00 00 11 10 1C 00 25 44 73 62 88
*Feb 11 07:33:50.628: 0x0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.628: 0x0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.628: 0x0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x00E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
*Feb 11 07:33:50.632: 0x0120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x01A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x01B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x01C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x01D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x01E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x01F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.632: 0x0200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.636: 0x0610: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.636: 0x0620: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.636: 0x0630: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.640: 0x0640: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.640: 0x0650: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.640: 0x0660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.640: 0x0670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*Feb 11 07:33:50.640: 0x0680: 00 0F 0F 0F
*Feb 11 07:33:50.640: cr10k_ipc_send_rpc_message_blocked -> RPC blocking message for
client 4 session 5/0 has sent successfully
*Feb 11 07:33:50.640: cr10k_plfm_tx_req_w_rsp() Inf: Done S=0
*Feb 11 07:33:50.640: cr10k_ipc_free_buffer -> Done deallocating IPC buffer for client 4.
Now pending count = 0
```

The following example shows the debug logging message:

```
Router# debug cmts ipc-cable client message client-id 7 entity-id 0 message-type 12
Router# test cmts ipc-cable client test nonblocked 5/0

TEST IPC-CABLE TEST APIs: nonblocked tx/rx
*Feb 11 07:52:11.364: cr10k clnt hdr: pack type(0), hdr len(12), msgs (1), total pyld(28)
*Feb 11 07:52:11.364: cr10k clnt payload:
*Feb 11 07:52:11.364: 0x0000: 00 0C 00 10 00 00 00 00 00 00 00 0C 00 01 02 03
*Feb 11 07:52:11.364: 0x0010: 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
*Feb 11 07:52:11.364: cr10k_ipc_send_ipc_message_nonblocked -> Sending non-blocking
message for client 7 session 5/0
*Feb 11 07:52:11.364: cr10k_message_dump_msg_buffer -> dump buffer size = 72, client_id =
7
*Feb 11 07:52:11.364: Dump before IPC nonblocking send:
*Feb 11 07:52:11.364: 0x0000: 00 00 28 0C 04 03 02 02 00 30 00 18 05 00 05 00
*Feb 11 07:52:11.364: 0x0010: 52 50 00 02 00 01 00 00 00 0C 80 07 00 28 00 08
*Feb 11 07:52:11.364: 0x0020: 00 0C 00 1C 00 01 00 00 00 00 00 00 00 0C 00 10
*Feb 11 07:52:11.364: 0x0030: 00 00 00 00 00 00 00 00 0C 00 01 02 03 04 05 06 07
*Feb 11 07:52:11.364: 0x0040: 08 09 0A 0B 0C 0D 0E 0F
*Feb 11 07:52:11.364: cr10k_ipc_send_ipc_message_nonblocked -> IPC non-blocking message
for client 7 session 5/0 has sent successfully
*Feb 11 07:52:11.364: cr10k_ipc_free_buffer -> Done deallocating IPC buffer for client 7.
Now pending count = 0
*Feb 11 07:52:11.364: cr10k_ipc_send_req_status_notify_inline -> Message sent to slot 5/0
for client 7 has ack. Msg Len = 40
TEST IPC-CABLE TEST APIs: OK!
```

The following example shows the PNEGO client and RP-CLC entity logging message:

Note When entity logging is enabled, all session, message, and packaging logging messages for that entity are enabled.

```
Router# debug cmts ipc-cable client entity client 3 entity-id 0

*Feb 11 08:02:42.708: cr10k_pnego_get_buf_guts() L317 C5/1 T20: Start L=0 SVC=704
*Feb 11 08:02:42.708: cr10k_pnego_get_buf_guts() L374 C5/1 T20: Done
*Feb 11 08:02:42.708: cr10k_pnego_get_data_ptr_guts() L609 T20: start B=0x12886654 Bv=1
Br=0)
*Feb 11 08:02:42.708: cr10k_pnego_get_data_ptr_guts() L621 T20: done(P=0xC0DBCF50)
*Feb 11 08:02:42.708: cr10k_pnego_tx_urgent_req() L756 C5/1 T20: Start
*Feb 11 08:02:42.708: cr10k_pnego_xform() L668 C5/1 T20: Start Tx=1
*Feb 11 08:02:42.708: cr10k_pnego_get_data_ptr_guts() L609 T20: start B=0x12886654 Bv=1
Br=0)
*Feb 11 08:02:42.708: cr10k_pnego_get_data_ptr_guts() L621 T20: done(P=0xC0DBCF50)
*Feb 11 08:02:42.708: cr10k_pnego_issu_xform_guts() L657 C5/1 T20: p=0xC0DBCF50
*Feb 11 08:02:42.708: cr10k_pnego_issu_xform_guts() L701 C5/1 T20: xmit xform ok
*Feb 11 08:02:42.708: cr10k_pnego_get_data_ptr_guts() L609 T20: start B=0x12886654 Bv=1
Br=0)
*Feb 11 08:02:42.708: cr10k_pnego_get_data_ptr_guts() L621 T20: done(P=0x0)
*Feb 11 08
Router#:02:42.708: cr10k_ipc_send_ipc_message_urgent -> Sending non-blocking message for
client 3 session 5/1
*Feb 11 08:02:42.708: cr10k_message_dump_msg_buffer -> dump buffer size = 760, client_id =
3
```

The following example shows the debugging message when the TEST client is enabled:

```
Router# debug cmts ipc-cable client client-id 7
Router# test cmts ipc-cable client test blocked 8/0

TEST IPC-CABLE TEST APIs: blocked tx/rx
*Feb 11 08:08:29.128: test_cr10k_clnt_test_blocked() Info: Start(C=8/0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_req_buf(): start(C8/0 typ=10 svc=0 len=16)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_req_buf(): done
*Feb 11 08:08:29.128: cr10k_clnt_test_get_ptr(): start(B=0x12886A20 Bv=1 Br=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_ptr(): done(P=0xC0D4BA50)
*Feb 11 08:08:29.128: cr10k clnt payload:
*Feb 11 08:08:29.128: 0x0000: 00 0C 00 10 00 00 00 00 00 00 00 0A 00 01 02 03
*Feb 11 08:08:29.128: 0x0010: 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
*Feb 11 08:08:29.128: cr10k_clnt_test_send_blocked(): start(C8/0 MC=0 TL=16571 T1=10 L1=16
B=0x12886A20)
*Feb 11 08:08:29.128: cr10k_clnt_test_xform(): Start C8/0 T1=10 Tx=1
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): start(B=0x12886A20 Bv=1 Br=0
Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): done(P=0xC0D4BA50)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_next
plfm-10k-4#_data_ptr(): start(B=0x12886A20 Bv=1 Br=0 Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_next_data_ptr(): done(P=0x0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): start(B=0x12886A20 Bv=1 Br=0
Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): done(P=0xC0D4BA50)
*Feb 11 08:08:29.128: cr10k clnt hdr: pack type(0), hdr len(12), msgs (1), total pyld(28)
*Feb 11 08:08:29.128: cr10k clnt payload:
*Feb 11 08:08:29.128: 0x0000: 00 0C 00 10 00 00 00 00 00 00 00 0A 00 01 02 03
*Feb 11 08:08:29.128: 0x0010: 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
*Feb 11 08:08:29.128: cr10k_ipc_send_rpc_message_blocked -> Sending blocking message for
client 7 session 8/0
*Feb 11 08:08:29.128: cr10k_message_dump_msg_buffer -> dump buffer size = 72, client_id =
7
*Feb 11 08:08:29.128: Dump before RPC blocking send:
```

```
*Feb 11 08:08:29.128: 0x0000: 00 00 28 0C 04 03 02 02 00 30 00 18 08 00 08 00
1 00 00 00 0C 80 07 00 28 00 08  50 00 01 00 0
*Feb 11 08:08:29.128: 0x0020: 00 0C 00 1C 00 01 00 00 00 00 00 00 00 0C 00 10
*Feb 11 08:08:29.128: 0x0030: 00 00 00 00 00 00 00 0A 00 01 02 03 04 05 06 07
*Feb 11 08:08:29.128: 0x0040: 08 09 0A 0B 0C 0D 0E 0F
*Feb 11 08:08:29.128: cr10k_ipc_send_rpc_message_blocked -> RPC blocking message for
client 7 session 8/0 has sent success
*Feb 11 08:08:29.128: cr10k_clnt_test_get_bndl_payload_size(): start(B=0x12886AB4 Bv=1
Br=1 Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_bndl_payload_size(): done(16)
*Feb 11 08:08:29.128: cr10k_clnt_test_xform(): Start C8/0 T1=11 Tx=0
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): start(B=0x12886AB4 Bv=1 Br=1
Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): done(P=0xC014AB0C)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_next_data_ptr(): start(B=0x12886AB4 Bv=1 Br=1
Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_next_data_ptr(): done(P=0x0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): start(B=0x12886AB4 Bv=1 Br=1
Bh=0)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_first_data_ptr(): done(P=0xC014AB0C)
*Feb 11 08:08:29.128: cr10k_clnt_test_send_blocked(): done(REQ B=0x12886A20 RSP S=0 MC=1
TL=16 T1=11 L1=16 B=0x12886AB4)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_ptr(): start(B=0x12886AB4 Bv=1 Br=1)
*Feb 11 08:08:29.128: cr10k_clnt_test_get_ptr(): done(P=0xC014AB0C)
*Feb 11 08:08:29.128: cr10k_clnt_test_free_buf(): B=0x12886AB4
*Feb 11 08:08:29.128: cr10k_ipc_free_buffer -> Done deallocating IPC buffer for client 7.
Now pending count = 0
*Feb 11 08:08:29.128: test_cr10k_clnt_test_blocked() Info: Done(1)
TEST IPC-CABLE TEST APIs: OK!
```

Related Commands

logging cmts ipc-cable	Enables debug logging for the cable IPC software.
show cmts ipc-cable	Displays information about the cable IPC layer on the Cisco CMTS routers.

debug cpd

To debug the CPD feature, use the **debug cpd** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug cpd verbose

no debug cpd verbose

Syntax Description

verbose	(Optional) Displays detailed debugging information.

Command Default

Debug is disabled and CPD request and response messages are not displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(21a)BC3	This command was introduced.

Examples

The following example shows enabling the **debug cpd** command:

```
Router# debug cpd
```

Related Commands

Command	Description
cpd	Enables CPD.

debug cr10k-rp

To enable debugging of the subsystems on the active Performance Routing Engine (PRE1) module on the Cisco uBR10012 router, use the **debug cr10k-rp pkt** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug cr10k-rp [**cli** | **drv** | **ha-all** | **ha-error** | **ha-msg** | **ha-recovery** | **ha-timing** | **ipc** | **oir** | **pkt** [**conditional** [**peek** *byte*]] | **sid** | **spec**]

no debug cr10k-rp [**cli** | **drv** | **ha-all** | **ha-error** | **ha-msg** | **ha-recovery** | **ha-timing** | **ipc** | **oir** | **pkt** [**conditional** [**peek** *byte*]] | **sid** | **spec**]

Syntax Description

cli	(Optional) Displays debugging messages for the command-line interface (CLI) commands run on the processor.
drv	(Optional) Displays debugging messages for the processor's driver software.
ha-all	(Optional) Displays debugging messages related to High Availability (HA) redundancy events.
ha-error	(Optional) Displays debugging messages about errors related to the recovery of cable modems during HA switchovers.
ha-msg	(Optional) Displays debugging messages for HA bulk synchronization operations.
ha-recovery	(Optional) Displays debugging messages related to the recovery operations for cable modems after HA switchovers. In particular, these messages concern the recovery of high-priority cable modems that have voice calls.
ha-timing	(Optional) Displays debugging messages related to HA timing events.
ipc	(Optional) Displays debugging messages for the processor's interprocess communications (IPC) system.
oir	(Optional) Displays debugging messages for online insertion and removal (OIR) operations.
pkt	(Optional) Displays debugging messages for the packets that the PRE1 module processes.
conditional	(Optional) Enables conditional debugging for the packets processed by the PRE1 module.
peek *byte*	(Optional) Specifies that debugging should show the value for a specific byte in each packet processed by the PRE. The valid range for *byte* is 1 to 120.
sid	(Optional) Displays debugging messages for the service IDs (SIDs) processed by the PRE1 module.
spec	(Optional) Displays debugging messages for spectrum management operations.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)BC1	This command was introduced for the Cisco uBR10012 universal broadband router.
12.2(11)BC2	The **conditional** and **peek** options were added for the **pkt** keyword.
12.2(11)BC3	The **ha-all**, **ha-msg**, and **ha-timing** options were added to support High-Availability N+1 (1:n) redundancy operations.

Usage Guidelines

The **debug cr10k-rp pkt** command enables debugging of the different subsystems that are active on the PRE1 modules in the Cisco uBR10012 router. You can perform general debugging by giving the command without any options, or you can limit the debugging output to a specific subsystem by specifying one of the optional keywords.

In Cisco IOS Release 12.2(11)BC2 and later releases, you can use the **conditional** option, together with the **debug cable mac-address** and **debug cable interface** commands, to display information about selected packets. The command will display only those packets that match the specified cable interface or MAC address options.

Together with the **conditional** option, you can also optionally specify the **peek** keyword to create specifically match only those packets that contain a matching MAC address at the specified location in the datagram. This can be useful for examining certain types of packets, such as DHCP or ARP packets.

Examples

The following example shows typical output for CLI debugging messages:

```
Router# debug cr10k-rp cli

CR10K RP debug CLI debugging is on

Failed setting clock for slot 2, subunit 1
SNMP Info download failed for slot 2, subunit 1
Config change command for unknown interface!!!
```

The following example shows typical output for IPC debugging messages:

```
Router# debug cr10k-rp ipc

CR10K RP debug IPC debugging is on

plugin_card__c10k_sch_card_event: hwidb=Cable8/1/0 if_num=0 event=6

00:03:14: clc_if_stats_event: If stats from Ca8/1/0
00:03:14: Merge: on Ca8/1/0
00:03:14:         Outputs         Tot_Outs     TotTxBytes
00:03:14:  RP       0            40            7103
00:03:14:  SIC      0            0            0
00:03:14:  LC       0            40            7103

plugin_card__c10k_sch_card_event: hwidb=Cable8/1/1 if_num=1 event=6

00:03:14: clc_if_stats_event: If stats from Ca8/1/1
00:03:14: Merge: on Ca8/1/1plugin_card__c10k_sch_card_event: hwidb=Cable6/0/0 if_num=0
event=6

00:03:15: clc_if_stats_event: If stats from Ca6/0/0
00:03:15:
Merge: on Ca6/0/0
00:03:15:         Outputs         Tot_Outs     TotTxBytes
```

```
00:03:15:  RP       0            0            0
00:03:15:  SIC      0            0            0
00:03:15:  LC       0            0            0
```

The following example shows typical output for IPC debugging messages when you are shutting down and reenabling a cable interface:

```
Router# debug cr10k-rp ipc

CR10K RP debug IPC debugging is on
Router# configure terminal
Router(config)# interface c6/0/0
Router(config-if)# shutdown

schrp_cli_cmd: slot=6 subunit=100 slotunit=8 cmdtype=101F
c10k_card_send_nbcmd_eventrsp: nbcmd_id=0x611CE998 hwidb=6B3DF60
plugin_card__c10k_sch_card_event: hwidb=0x6B3DF60 interface_num=0 event=4096
sch_handle_sch_event: erso_type=0x1001
sch_handle_sch_rp_cfg_ersp(): hwidb=0x6B3DF60 msg_size=0x0 0x78  ersp_size=0x0 0x78
type=0x1001
plugin_card__c10k_sch_card_event: hwidb=0x6B3DF60 interface_num=0 event=4
c10k_sch_link_state_event: hwidb=0x6B3DF60 unit=0 seq=34 reason=2 event_state=1


Router(config-if)# no shutdown

Router(config-if)#schrp_cli_cmd: slot=6 subunit=100 slotunit=8 cmdtype=101F
schrp_cli_cmd: SCH_API_CMD_GET_INIT_DS hwidb=6B3DF60
c10k_card_send_nbcmd_eventrsp: nbcmd_id=0x611CE998 hwidb=6B3DF60
plugin_card__c10k_sch_card_event: hwidb=0x6B3DF60 interface_num=0 event=4096
sch_handle_sch_event: erso_type=0x1001
sch_handle_sch_rp_cfg_ersp(): hwidb=0x6B3DF60 msg_size=0x0 0x950  ersp_size=0x0 0x950
type=0x1001
c10k_card_send_nbcmd_eventrsp: nbcmd_id=0x611CE998 hwidb=6B3DF60
plugin_card__c10k_sch_card_event: hwidb=0x6B3DF60 interface_num=0 event=4096
sch_handle_sch_event: erso_type=0x1001
sch_handle_sch_rp_cfg_ersp(): hwidb=0x6B3DF60 msg_size=0x0 0x20  ersp_size=0x0 0x20
type=0x1001
plugin_card__c10k_sch_card_event: hwidb=0x6B3DF60 interface_num=0 event=4
c10k_sch_link_state_event: hwidb=0x6B3DF60 unit=0 seq=35 reason=2 event_state=1
```

The following example shows a typical display for the **debug cr10k-rp pkt conditional** command, which displays packets for SID 2 on cable interface 6/1/0:

```
Router# debug cable interface c6/1/0 sid 2
Router# debug cr10k-rp pkt conditional
Router# show debug

CR10K PACKET:
  Dump cr10k packets to/from RP conditionally
```

The following example shows how to enable conditional debugging of packets, displaying only those packets that contain the desired mac address at byte 92 in the datagram:

```
Router# debug cr10k-rp pkt conditional peek 92
Router# debug cable interface c6/1/0 mac-address 00C0.abcd.ef00
Router# show debug
CR10K PACKET:
  Dump cr10k packets to/from RP conditionally
    Additionally, peeking inside transmitted pkts at offset 92
```

Both types of packet debugging generate output similar to the following example:

```
Jun 19 13:07:32.316: RPTX: Using Downstream Service Flow ID : 16939, SID : 2  V5
```

```
Jun 19 13:07:32.316: RPTX to Cobalt: 0x801B634, size=111
006F0000 00057010 422B2488 00020000 10000006 10000010 950701DB 00016440
D1420800 4500004B 2B260000 FF11187A A4789781 A478978F CD7E00A1 0037D2CA
302D0201 00040670 75626C69 63A02002 03062B27 02010002 01003013 3011060D
  ....

Jun 19 13:07:32.316: RPTX: Using Downstream Service Flow ID : 16939, SID : 2  V5
Jun 19 13:07:32.316: RPTX to Cobalt: 0x8023834, size=111
006F0000 00057010 422B2488 00020000 10000006 10000010 950701DB 00016440
D1420800 4500004B 2B270000 FF111879 A4789781 A478978F CD7E00A1 0037D2CB
302D0201 00040670 75626C69 63A02002 03062B28 02010002 01003013 3011060D
  ....
```

Table 0-259 explains the information contained in the display for each packet:

Table 0-259 debug cr10k-rp pkt Field Descriptions

Field	Description
RPIX: Using Downstream Service Flow ID	Displays the service flow ID (SFID) for this packet.
SID	Displays the service ID (SID) for this packet.
RPTX or RPRX	Displays whether this packet is transmitted (RPTX) or received (RPRX) by the processor.
Size	Displays the size of the packet's datagram in decimal.
Packet Dump	Displays the first 96 bytes of the packet's datagram in hexadecimal. The command then displays an ellipses (. . .) if the datagram is larger than 96 bytes.

The following example shows typical output for SID debugging messages:

```
Router# debug cr10k-rp sid

CR10K RP debug SID debugging is on

(Cable 6/1/0:2): CM Offline - MAC 00C0.1234.5678, SID 113
(Cable 6/1/0:2): -Shutdown CM- SID 231
(Cable 6/1/0:2): CM Shutdown - MAC 00C0.2210.a01c, SID 231
(Cable 6/1/0:2): CM Remove - MAC 00C0.2210.a01c, SID 231
Call SID replace with old IP addr 10.10.13.18 new IP addr 10.10.13.121
(Cable 5/1/1:1) - New CM 00C0.1122.bcab, SID 396
(Cable 5/1/0:1) - New CM 00C0.8723.11F0, SID 397
(Cable 5/1/0:1) - CM Init FAILED - MAC 00C0.8723.11F0, SID 397
```

The following example shows typical output for spectrum management debugging messages for a particular interface:

```
Router# debug cable interface c6/1/0 sid 2
Router# debug cr10k-rp pkt spec

CR10K RP debug Spectrum debugging is on

(Cable 5/0/0:4) Release frequency (11600000, 3200000) from group 12
(Cable 5/0/0:4) Frequency request (10000000 - 13200000) from group 12 approved
(Cable 7/0/0:1) Frequency request (12000000 - 13600000) from group 2 approved
(Cable 7/0/0:1) Release frequency (12800000, 1600000) from group 2
(Cable 7/0/0:1) Frequency request (12000000 - 15200000) from group 2 approved
(Cable 7/0/0:2) Frequency request (20000000 - 21600000) from group 22 approved
(Cable 7/0/0:2) Release frequency (20800000, 1600000) from group 22
```

```
(Cable 7/0/0:2) Frequency request (20000000 - 23200000) from group 22 approved
(Cable 7/0/0:3) Frequency request (20000000 - 21600000) from group 22 approved
(Cable 7/0/0:3) Release frequency (20800000, 1600000) from group 22
(Cable 7/0/0:4) Release frequency (20800000, 1600000) from group 22
(Cable 7/0/0:4) Frequency request (20000000 - 20400000) from group 22 approved
(Cable 7/0/0:5) Frequency request (20000000 - 21600000) from group 22 approved
(Cable 7/0/0:5) Release frequency (20800000, 1600000) from group 22
(Cable 7/0/0:5) Frequency request (20000000 - 20200000) from group 22 approved
(Cable 7/0/0:6) Frequency request (20000000 - 21600000) from group 22 approved
Release frequency request sent to slot 7 subslot 0
(Cable 7/0/0:6) Frequency request (21400000 - 21800000) from group %d rejected because of
overlapping band
```

The following lists the typical messages that are displayed by the **ha-all** option.

```
PRE Redundancy: failed to get RF buf to send msg to peer
PRE Redundancy: failed to send RF buf to peer
PRE REDUNDANCY: bulk sync in progress
PRE REDUNDANCY: Got RF message when not in Active state.
RP DS SRV FLOW packet sanity check... ok.
Failed to find DS SF with sfid 321
Deleting an old inter_rp_ds_srv_flow
Adding a new inter_rp_ds_srv_flow
Updating an inter_rp_ds_srv_flow
Processing inter_rp_ds_srv_flow
cr10k_rp_ha_parse_sync: Received Packet of 122 len
cr10k_rp_ha_parse_sync:Received
Synching cm instance
CR10K HA:  Standby recv chkpt msg
PRE REDUNDANCY: NO Resources
PRE REDUNDANCY: RF_PROG_STANDBY_BULK received
PRE REDUNDANCY:This is standby from boot
PRE REDUNDANCY: RF_PROG_ACTIVE_FAST received
```

The following example shows the typical messages for the **ha-timing** option. These messages show the total time it takes to synchronize all of the cable modems on a cable interface after a switchover, as well as the total time it takes for all cable modems to recover and come online. These messages also show the total time it took the HCCP and DOCSIS protocol subsystems to synchronize after a switchover.

```
Router# debug cable interface c6/1/0
Router# debug cr10k-rp ha-timing

CR10K RP debug High Availability timing

PRE_HA: c6/1/0 Total modems 234 bulk sync'ed in 531 msec
    (delay: 20 msec; CM's per loop:10)
PRE_HA: c6/1/0 Total modems (234) recovered in 1124 msec
PRE_HA: Completed hccp bulksync in 335 msecs
PRE_HA: Completed docsis bulksync in 751 msecs
```

The following example shows the typical messages for the **ha-msg** option. These messages show the total time it takes to synchronize all of the cable modems on a cable interface after a switchover, as well as the total time it takes for all cable modems to recover and come online. These messages also show the total time it took the HCCP and DOCSIS protocol subsystems to synchronize after a switchover.

```
Router# debug cable interface c6/1/0
Router# debug cr10k-rp ha-msg

CR10K RP debug High Availability msg

PRE_HA_BUGMSG
PRE REDUNDANCY: Bulk sync completed
PRE REDUNDANCY: Recv bulk sync complete - sending ack
```

```
PRE REDUNDANCY: Send bulk sync ack failed
PRE RF: Waiting for bulk sync ack
PRE REDUNDANCY: Bulk sync completed
PRE REDUNDANCY: Recv bulk sync complete - sending ack
PRE REDUNDANCY: Send bulk sync ack failed
PRE RF: Waiting for bulk sync ack
```

The following example shows the typical messages for the **ha-recovery** option. The following messages show the typical recovery process, in which cable modems with active voice calls are recovered first. After these cable modems are online, the remaining, data-only cable modems are recovered.

```
PRE_RECOVERY: <Cable I/F> Received message to recover voice
PRE_RECOVERY: Suspending recovery until higher priority modems are recovered
PRE_RECOVERY: All interfaces recovered. Killing the voice process
PRE_RECOVERY: No events in voice queue. Allow data modems to recover
PRE_RECOVERY: <Cable I/F> Resume data recovery after suspend (modems recovered <# of CMs>)
PRE_RECOVERY:<Cable I/F> Received message to recover data
PRE_RECOVERY: All interfaces recovered. Killing the data process
```

The following are possible messages that can be displayed because of potential error conditions:

```
PRE_RECOVERY:<Cable I/F> No modems to be recovered
PRE_HA:<Cable I/F> Unable to start the voice recovery process
PRE_HA: <Cable I/F> Unable to start the data recovery process
```

Related Commands

Command	Description
debug cable interface	Enables debugging on a specific cable interface.
debug cable mac-address	Enables debugging for a specific CM, as identified by its hardware (MAC) address.

debug cr10k-rp dbs-queue

To display debug information for dynamic bandwidth sharing (DBS) on the Cisco uBR10012 universal broadband router, use the **debug cr10k-rp dbs-queue** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cr10k-rp dbs-queue

no debug cr10k-rp dbs-queue

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(23)BC1	This command was introduced.
12.2(33)SCB	This command was integrated into Cisco IOS Release 12.2(33)SCB.

Usage Guidelines

The **debug cr10k-rp dbs-queue** command is used only with the Cisco uBR10012 universal broadband router.

Note Routine use of the debug cr10k-rp dbs-queue command is not recommended. If you require further information, contact Cisco technical assistance at http://www.cisco.com/techsupport.

Examples

The following example shows that the debug option has been turned on using the **debug cr10k-rp dbs-queue** command :

```
Router# debug cr10k-rp dbs-queue
CR10K RP debug dynamic BG link queue setup debugging is on
```

Related Commands

Command	Description
cable dynamic-bw-sharing	Enables dynamic bandwidth sharing on a specific modular cable or wideband cable interface.
show pxf cable controller	Displays information about the RF channel Versatile Traffic Management System (VTMS) links and link queues.
show pxf cpu queue	Displays parallel express forwarding (PXF) queueing and link queue statistics.

debug docsis ssd

To display debugging information about the parsing and verification of the DOCSIS code verification certificates (CVCs) that are part of a software image downloaded with the Secure Software Download (SSD) procedure, use the **debug docsis ssd** command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

Cisco uBR905 and Cisco uBR925 cable access routers, and Cisco CVA122 Cable Voice Adapter

debug docsis ssd

no debug docsis ssd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)CZ	This command was introduced for the Cisco uBR905 and Cisco uBR925 cable access routers, and the Cisco CVA122 Cable Voice Adapter.

Usage Guidelines

This command displays whether the Secure Software Download procedure could validate the manufacturer's CVC and optional cosigner's CVC (if present) that are part of the downloaded software image.

Examples

The following example shows typical output for a successful Secure Software Download procedure for a software image that has been signed by both the manufacturer and by a cosigner:

```
Router# debug docsis ssd

secure software download debugging is on
Router#
SSD: decrypt process suspended and continued
SSD: decrypt process suspended and continued
Code Verification Successful (Manufacturer CVC/CVS)
Verifying Co-Signer CVC/CVS
SSD: decrypt process suspended and continued
SSD: decrypt process suspended and continued
Co-signer CVC has been validated
Code Verification Successful (Co-Signer CVC/CVS)

Router#
```

If the manufacturer's signature on the software image file cannot be validated using the manufacturer's CVC on the router, the following messages are displayed:

```
MFG code signature does not validate
MFG CVC validation has failed
```

If the MSO cosigner's signature on the software image file cannot be validated using the cosigner's CVC on the router, the following messages are displayed:

```
Co-signer code signature does not validate
Co-signer CVC validation has failed
```

If the software image was signed either before or after the allowable time range specified as part of the manufacturer's CVC, one of the following messages is displayed:

```
signingTime is before saved codeAccessStart
signingTime is before CVC validNotBefore
signingTime is after CVC validNotAfter
CVC validity start is less than save cvcAccessStart
```

Related Commands

Command	Description
docsis cvc mfg	Configures the access start times and organization name for the manufacturer's code verification certificate (CVC).
docsis cvc mso	Configures the access start times and organization name for the optional MSO cosigned code verification certificate (CVC).
docsis cvc test	Tests the root CA public key and CM private key that are installed on the router.

debug ehsa

To enable debug information about enhanced high system availability (EHSA) activity, use the **debug ehsa** command in privileged EXEC mode. To disable debugging output for the EHSA module, use the **no** form of this command.

debug ehsa {**alarms** | **all** | **configsync** | **fsm** | **keepalive** | **peer-monitor** | **services** | **timesync**}

no debug ehsa

Syntax Description

alarms	Enables debugging of EHSA alarms.
all	Enables all EHSA debugging types.
configsync	Enables debugging of EHSA configuration synchronization.
fsm	Enables debugging of EHSA redundancy finite state machine (FSM).
keepalive	Enables debugging of EHSA keepalive messages sent between the active Performance Routing Engine (PRE) and the standby PRE.
peer-monitor	Enables EHSA debugging messages for the standby PRE monitoring of the active PRE.
services	Enables debugging of EHSA services for the services requested during redundancy processing.
timesync	Enables debugging of EHSA time synchronization procedures.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCA	This command was introduced on the Cisco uBR10012 router.

Usage Guidelines

The **debug ehsa** command must be enabled on the active PRE and on the standby PRE to ensure that debug output after a switchover is displayed.

Examples

The following example shows all debugging output during a switchover using the **all** keyword:

```
Router# debug ehsa all

Redundancy All debugging is on
router#
...

router# show log | inc EHSA
*Apr  6 04:43:53.007: EHSA: Process KEEPALIVE(30) bootstrap message.
*Apr  6 04:43:53.007: EHSA: Received keepalive
```

```
*Apr  6 04:43:53.007: EHSA: Sent keepalive
...

*Apr  6 04:43:55.423: EHSA: IPC message request-type SLAVE_SERVICES_OIR_EVENT_REQ(23)
*Apr  6 04:43:55.423: PEHSA:processing OIR event received from Active
...
*Apr  6 04:43:56.007: EHSA: Process KEEPALIVE(30) bootstrap message.
*Apr  6 04:43:56.007: EHSA: Received keepalive
*Apr  6 04:43:56.007: EHSA: Sent keepalive
...

*Apr  6 04:44:13.287: EHSA: Filter event raw-event=FORCE_SWITCHOVER(10), peer ehsa-states:

old=Unknown redundancy state(0) new=Unknown redundancy state(0)
*Apr  6 04:44:13.287: EHSA: Switching Standby to Active
*Apr  6 04:44:13.287: EHSA: Filter event raw-event=FORCE_SWITCHOVER(10), peer ehsa-states:

old=Unknown redundancy state(0) new=Unknown redundancy state(0)
*Apr  6 04:44:13.291: EHSA: Changing from IPC slave to IPC Master
*Apr  6 04:44:15.111: EHSA: Filter event raw-event=PEER_CRASHED(1), peer ehsa-states:

old=Unknown redundancy state(0) new=Unknown redundancy state(0)
*Apr  6 04:44:15.111: EHSA: Filter event raw-event=PEER_REDUNDANCY_STATE_CHANGE(5), peer

ehsa-states: old=ACTIVE(1) new=STANDALONE(3)
*Apr  6 04:44:16.535: EHSA: Exception dump to network is false
*Apr  6 04:44:16.535: EHSA: Created Parser Mode History for tty 0
*Apr  6 04:44:16.535: EHSA: Killing Peer monitor
*Apr  6 04:44:16.567: EHSA: Restarting Peer monitor
*Apr  6 04:44:16.567: EHSA: Asserting alarm : SWITCHOVER
*Apr  6 04:44:16.567: EHSA: state change, events: major=2 minor=0
*Apr  6 04:44:17.623: EHSA: Filter event raw-event=PEER_CRASHED(1), peer ehsa-states:

old=Unknown redundancy state(0) new=Unknown redundancy state(0)
*Apr  6 04:44:17.623: EHSA: Filter event raw-event=PEER_REDUNDANCY_STATE_CHANGE(5), peer

ehsa-states: old=ACTIVE(1) new=STANDALONE(3)
*Apr  6 04:44:25.707: EHSA: Filter event raw-event=PEER_REDUNDANCY_STATE_CHANGE(5), peer

ehsa-states: old=STANDALONE(3) new=STANDBY(2)
*Apr  6 04:44:33.259: EHSA: Created Parser Mode History for tty 2
*Apr  6 04:44:33.259: EHSA: csb(0x201A24B4), line(interface Cable5/1/0), old_mode2(NULL),

parser_mode(0x645B7CE8), mode(configure)
*Apr  6 04:44:33.259: EHSA: PRC csb->line (interface Cable5/1/0), old_mode2 (configure),

csb->mode (interface)
*Apr  6 04:44:33.259: EHSA: Parser mode history for ttynum 0, pmode_idx 0
*Apr  6 04:44:33.259: EHSA: Parser mode history for ttynum 2, pmode_idx 1
*Apr  6 04:44:33.259: EHSA: Parser mode history table population 2 of 1512

...

*Apr  6 04:44:33.263: EHSA: csb(0x201A24B4), line(default cable upstream 3 shutdown),

old_mode2(configure), parser_mode(0x645B7D54), mode(interface)
*Apr  6 04:44:33.263: EHSA: PRC csb->line (default cable upstream 3 shutdown), old_mode2

(interface), csb->mode (interface)
*Apr  6 04:44:33.263: EHSA: Parser mode history for ttynum 0, pmode_idx 0
*Apr  6 04:44:33.263: EHSA: Parser mode history for ttynum 2, pmode_idx 1
*Apr  6 04:44:33.263: EHSA: Parser mode history table population 2 of 1512
*Apr  6 04:44:33.267: EHSA: csb(0x201A24B4), line(default cable upstream 3
modulation-profile
```

```
21), old_mode2(interface), parser_mode(0x645B7D54), mode(interface)

...

*Apr  6 04:48:41.355: EHSA: Sent keepalive
*Apr  6 04:48:43.007: EHSA: Process KEEPALIVE(30) bootstrap message.
...
```

Related Commands

Command	Description
redundancy force-switchover	Forces the standby RP to assume the role of the active RP.

debug hccp authentication

To display authentication debug messages for Hot Standby Connection-to-Connection (HCCP) groups, use the **debug hccp authentication** command in privileged EXEC mode. To disable HCCP authentication debug message output, use the **no** form of this command.

debug hccp authentication

no debug hccp authentication

Syntax Description

This command has no arguments or keywords.

Command Default

Debug is disabled and authentication messages for HCCP groups are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.1(3a)EC	This command was introduced.
12.2(4)XF1, 12.2(4)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR10012 router.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

You must use the **debug hccp events** command before the **debug hccp authentication** command will generate any debug message output.

Examples

The following example shows the additional N+1 redundancy authentication debug message output produced when the **debug hccp authentication** command has been activated:

```
Router# debug hccp events
Router# debug hccp authentication

Sep  7 09:51:50.151:HCCP 1 0->1:HELLO Learn tran 31708
Sep  7 09:51:50.151:auth md5 keyid 1 digest B77F65ED 1B38ED5C 87A7037B C006DAFB
```

Related Commands

Command	Description
debug hccp channel-switch	Displays debug messages related to an RF or channel switch that is being used for HCCP N+1 (1:n) redundancy.
debug hccp events	Displays debug messages for all HCCP group interaction.

Command	Description
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp channel-switch

To display debug messages related to a radio frequency (RF) or channel switch that is being used for Hot Standby Connection-to-Connection Protocol (HCCP) N+1 (1:n) redundancy, use the **debug hccp channel-switch** command in privileged EXEC mode. To disable the debug message output, use the **no** form of this command.

debug hccp channel-switch

no debug hccp channel-switch

Syntax Description

This command has no arguments or keywords.

Command Default

Debug is disabled and channel switch messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(4)XF1, 12.2(4)BC1	This command was introduced for the N+1 (1:n) RF Switch with the Cisco uBR10012 router.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

The **debug hccp channel-switch** command displays debugging messages related to the SNMP messages and other communication between the CMTS and an RF-switch or channel-switch during N+1 (1:n) HCCP operations.

Note You must use the **debug hccp events** command to activate HCCP event debugging before the **debug hccp channel-switch** command will generate any debug output.

Examples

The following example shows typical output for the **debug hccp channel-switch** command:

```
Router# debug hccp events
Router# debug hccp channel-switch

HCCP channel action debugging is on

Mar 7 10:11:55:HCCP: snmp response error from 1.8.26.100.
Mar 7 10:12:16:HCCP: snmp response error from 1.8.26.100.
Mar 7 10:13:23:HCCP_SNMP: unknown Wavecom snmp type
Mar 7 10:12:21:HCCP: failed to make community
Mar 7 10:12:21:HCCP: failed to create snmp message
Mar 7 10:12:21:HCCP: SPrintVarBind() failed
```

```
Mar 7 10:12:21:HCCP: snmp request failed
Mar 7 10:12:21:HCCP: snmp request cancelled
Mar 7 11:51:50:HCCP: group 1 member 1 Working is loading config
Mar 7 11:51:52:HCCP: group 1 member 1 Working finished loading config
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp events	Displays debug messages for all HCCP group interaction.
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp docsis-recovery

To display debug messages about the recovery of cable modems (CMs) during Hot Standby Connection-to-Connection Protocol (HCCP) N+1 (1:n) redundancy operations, use the **debug hccp docsis-recovery** command in privileged EXEC mode. To disable the debug message output, use the **no** form of this command.

debug hccp docsis-recovery

no debug hccp docsis-recovery

Syntax Description

This command has no arguments or keywords.

Command History

Debug is disabled and DOCSIS recovery messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(15)BC	This command was introduced.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

The **debug hccp docsis-recovery** command displays debugging messages about the recovery of CMs during N+1 (1:n) HCCP operations.

Note You must use the **debug hccp events** command to activate HCCP event debugging before the **debug hccp docsis-recovery** command will generate any debug output.

Examples

The following example shows typical output for the **debug hccp docsis-recovery** command:

```
Router# debug hccp events
Router# debug hccp docsis-recovery

HCCP DOCSIS Recovery ACTION/EVENT messages debugging is on

HCCP 1 1 Working: Built 4 prio 3 CMs [tot 23]
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp events	Displays debug messages for all HCCP group interaction.

Command	Description
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp events

To display debug messages for all Hot Standby Connection-to-Connection Protocl (HCCP) group interaction, use the **debug hccp events** command in privileged EXEC mode. To disable HCCP group debug message output, use the **no** form of this command.

debug hccp events

no debug hccp events

Syntax Description

This command has no arguments or keywords.

Command Default

Debug is disabled and no HCCP group messages are displayed.

Command Modes

Privileged Exec (#)

Command History

Release	Modification
12.1(3a)EC	This command was introduced.
12.2(4)XF1, 12.2(4)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR10012 router.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

You must use the **debug hccp events** command to activate HCCP debugging before the other **debug hccp** commands will display debugging output.

Examples

The following example shows HCCP group interaction debug message output produced when the **debug hccp events** command has been activated:

```
Router# debug hccp events

Sep  7 09:51:50.151:HCCP 1 0->1:HELLO Learn tran 31708
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp channel-switch	Displays debug messages related to an RF or channel switch that is being used for HCCP N+1 (1:n) redundancy.
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.

Command	Description
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp fast-failure-detection

To display debug messages for fast failure detection (FFD) operations during Hot Standby Connection-to-Connection (HCCP) N+1 (1:n) redundancy, use the **debug hccp fast-failure-detection** command in privileged EXEC mode. To disable the debug message output, use the **no** form of this command.

debug hccp fast-failure-detection

no debug hccp fast-failure-detection

Syntax Description

This command has no arguments or keywords.

Command Default

Debug is disabled and FFD messages are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(15)BC	This command was introduced.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

The **debug hccp fast-failure-detection** command displays debugging messages about the operation of sync pulse messages for FFD support during N+1 (1:n) HCCP operations. A Working interface sends these sync pulse messages upon a failure to proactively inform the Protect interface that a switchover is occurring.

Note You must use the **debug hccp events** command to activate HCCP event debugging before the **debug hccp fast-failure-detection** command will generate any debug output.

Examples

The following example shows typical FFD debugging messages that are displayed by the **debug hccp fast-failure-detection** command:

```
Router# debug hccp events
Router# debug hccp fast-failure-detection

HCCP Fast Failure Detection ACTION/EVENT messages debugging is on

 HCCP: FFD cutover packet address updated: src = 10.10.10.3, dst = 10.10.10.4
 HCCP: FFD cutover packet allocated!
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp events	Displays debug messages for all HCCP group interaction.
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp inter-db

To display debug messages for the inter-database events during Hot Standby Connection-to-Connection Protocol (HCCP) operations, use the **debug hccp inter-db** command in privileged EXEC mode. To disable HCCP group debug message output, use the **no** form of this command.

debug hccp inter-db {**error** | **events**}

no debug hccp inter-db {**error** | **events**}

Syntax Description

error	Display debugging messages related to database errors that occur.
events	Display debugging messages related to all events that occur during database operations.

Command Default

Debug is disabled and inter-database messages are not displayed.

Command History

Privileged Exec (#)

Command History

Release	Modification
12.1(3a)EC	This command was introduced.
12.2(4)XF1, 12.2(4)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR10012 router.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

You must use the **debug hccp events** command before the **debug hccp inter-db** command will generate any debug message output.

Examples

The following example shows typical debug output for the **debug hccp inter-db error** command:

```
Router# debug hccp events
Router# debug hccp inter-db error

HCCP inter database ERROR messages debugging is on

00:38:11: NULL itemplatep in cmts_build_us_srv_flow_loc.
00:38:11: NULL flowp in cmts_build_phs
00:38:11: NULL flowp in cmts_build_phs
00:38:11: Interface C5/0/0.3 does not exist. Resetting idb mapping for sid 3 to main
interface
00:38:11: cmts_map_sid_idb(): failed to find swidb for 5.0.0.10 on Cable5/0/0
00:38:12: cmts_map_sid_idb(): failed to find swidb for 5.0.0.24 on Cable5/0/0
00:38:12: NULL parsed_datp in cmts_build_phs
00:38:12: NULL cminstp in cmts_build_classifier
```

```
00:38:12: NULL iflowp in cmts_build_classifier
00:38:12: NULL icfrp in cmts_build_classifier
00:38:12: NULL parsed_datp in cmts_build_classifier
00:38:12: NULL parsed_datp in cmts_build_us_srv_flow_loc
00:38:12: NULL itemplatep in cmts_build_us_srv_flow_loc.
```

The following example shows typical debug output for the **debug hccp inter-db events** command:

```
Router# debug hccp events
Router# debug hccp inter-db events

HCCP inter database EVENTS messages debugging is on

Jun  4 10:44:38.282: SRV TEMPLATE packet sanity check... ok.
Jun  4 10:44:38.282: Processing inter_srv_template.
Jun  4 10:44:38.282: Adding a new inter_srv_template
Jun  4 10:44:38.282: updating an existing inter_srv_template
Jun  4 10:44:38.286: UCD packet sanity check... ok.
Jun  4 10:44:38.286: Processing inter_ucd_instance
Jun  4 10:44:38.286: UCD packet sanity check... ok.
Jun  4 10:44:38.286: Processing inter_ucd_instance
Jun  4 10:44:38.286: CM packet sanity check... ok.
Jun  4 10:44:38.286: Processing inter_cm_instance.
Jun  4 10:44:38.286: Failed to find CM with mac address 0000.0cbd.cedf
Jun  4 10:44:38.286: The old inter_cm_instance is stale
Jun  4 10:44:38.286: Deleting an old inter_cm_instance
Jun  4 10:44:38.286: Adding a new inter_cm_instance
Jun  4 10:44:38.286: Adding a new inter_sid_instance
Jun  4 10:44:38.286: Updating an existing inter_cm_instance
Jun  4 10:44:38.286: SRV_FLOW packet sanity check... ok.
Jun  4 10:44:38.338: SID packet sanity check... ok.
Jun  4 10:44:38.338: Processing inter_sid_instance
Jun  4 10:44:38.338: Updating an existing inter_sid_instance
Jun  4 10:44:38.338: CM packet sanity check... ok.
Jun  4 10:53:42.907: Ranged list packet sanity check... ok.
Jun  4 10:53:42.907: Processing ranged_list.
Jun  4 10:53:42.907: Ranged list packet sanity check... ok.
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp channel-switch	Displays debug messages related to an RF or channel switch that is being used for HCCP N+1 (1:n) redundancy.
debug hccp events	Displays debug messages for all HCCP group interaction.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp plane

To display debug messages for Hot Standby Connection-to-Connection Protocol (HCCP)-related messages sent between the router's control plane and data backplane, use the **debug hccp plane** command in privileged EXEC mode. To disable HCCP group debug message output, use the **no** form of this command.

debug hccp plane {**message** | **packet** | **syncpulse**}

no debug hccp plane {**message** | **packet** | **syncpulse**}

Syntax Description

message	Display debugging messages for the inter-plane messages that occur.
packet	Display a partial dump of packets that are sent between the control plane and data backplane.
syncpulse	Display debugging messages related to the "heartbeat" packets that are sent between the control plane and data backplane to synchronize events.

Command Default

Debug is disabled and HCCP control plane messages are not displayed.

Command History

Privileged Exec (#)

Command History

Release	Modification
12.2(4)BC1	This command was introduced for the Cisco uBR10012 router.
12.2(4)XF1, 12.2(4)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR10012 router.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router. The **syncpulse** option also was added.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

The processor card and line cards on the Cisco uBR10012 router chassis communicate with each other using the control plane (which is on the processor card) and the data backplane (the backplane into which the line cards are installed). The **debug hccp plane** command displays debugging messages related to the messages and packets that are transmitted between the control plane and data backplane.

Note You must use the **debug hccp events** command to activate HCCP event debugging before the **debug hccp plane** command will generate any debug output.

Examples

The following example shows typical output that is produced by the **debug hccp plane message** command:

```
Router# debug hccp events
```

```
Router# debug hccp plane message

HCCP inter-plane message debugging is on

Mar  7 08:47:26.513: HCCP 1 Working 4: Control plane sending message BECOME_ACTIVE_STRUCT
Mar  7 08:47:26.513: HCCP 1 Working Cable7/1/0: Control plane sending message FORWARDING
Mar  7 08:47:26.513: HCCP 1 Working 1: Control plane sending message STOP_SYNC
Mar  7 08:47:26.513: HCCP 1 Working 5: Control plane sending message STOP_SYNC
Mar  7 08:47:26.513: HCCP 2 Working 5: Control plane sending message BECOME_ACTIVE_STRUCT
Mar  7 08:47:26.513: HCCP 2 Working Cable7/0/1: Control plane sending message FORWARDING
Mar  7 08:47:26.513: HCCP 2 Working 4: Control plane sending message STOP_SYNC
Mar  7 08:47:26.513: HCCP 1 Protect 80: Data plane sending message FAILURE_SYNCPULSE
Mar  7 08:47:26.513: HCCP 4 Protect 81: Data plane sending message FAILURE_SYNCPULSE
Mar  7 08:47:26.513: HCCP 1 Protect 80: Data plane received message BECOME_ACTIVE_STRUCT
Mar  7 08:47:26.513: HCCP 2 Working 1: Control plane sending message STOP_SYNC
Mar  7 08:47:26.513: HCCP 1 Protect Cable8/1/0: Data plane received message FORWARDING
Mar  7 08:47:26.513: HCCP 1 Protect 1: Control plane sending message INIT_DS
Mar  7 08:47:26.513: HCCP 1 Protect 3: Control plane sending message INIT_DS
Mar  7 08:47:26.513: HCCP 2 Protect 5: Control plane sending message INIT_DS
Mar  7 08:47:26.513: HCCP 1 Protect Cable5/1/0: Control plane sending message READY
Mar  7 08:47:26.513: HCCP 1 Protect 1: Data plane received message START_SYNC
Mar  7 08:47:26.513: HCCP 1 Protect 1: Data plane received message DO_STATICSYNC
Mar  7 08:47:26.513: HCCP 1 Protect 1: Data plane sending message STATICDONE
Mar  7 08:47:26.513: HCCP 2 Protect 1: Data plane received message START_SYNC
Mar  7 08:47:26.513: HCCP 2 Protect 1: Data plane received message DO_STATICSYNC
Mar  7 08:47:26.513: HCCP 2 Protect 1: Data plane sending message STATICDONE
Mar  7 08:47:26.513: HCCP 2 Working 1: Control plane sending message BECOME_ACTIVE_STRUCT
Mar  7 08:47:26.513: HCCP 2 Working Cable5/0/1: Control plane sending message FORWARDING
Mar  7 08:47:26.513: HCCP 2 Protect 1: Control plane sending message STOP_SYNC
Mar  7 08:47:26.513: HCCP 2 Protect 1: Control plane sending message BECOME_STANDBY_STRUCT
Mar  7 08:47:26.513: HCCP 2 Protect Cable5/1/1: Control plane sending message BLOCKING
Mar  7 08:47:26.513: HCCP 2 Protect Cable5/1/1: Control plane sending message READY_NOT
Mar  7 08:47:26.513: HCCP 2 Protect 1: Control plane sending message DEACTIVATE_MEMBER
```

The following example shows a typical example of the message that is produced by the **debug hccp plane packet** command:

```
Router# debug hccp events
Router# debug hccp plane packet

HCCP inter-plane packet debugging is on

Aug  2 00:37:24.203: HCCP 1 70 Protect: Data plane receives DOCSIS sync packet
```

The following example shows a typical example of the message that is produced by the **debug hccp plane syncpulse** command:

```
Router# debug hccp events
Router# debug hccp plane syncpulse

HCCP inter-plane sync-pulse packet debugging is on

Aug  2 00:37:24.203: HCCP 1 70 Protect: Data plane receives DOCSIS sync packet
Aug  2 00:37:24.203:     CMTS HCCP docsis_ver 1, sw_ver 1
```

Tip Use both the **message** and **syncpulse** keyword options to debug heartbeat synchronization problems.

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp channel-switch	Displays debug messages related to an RF or channel switch that is being used for HCCP N+1 (1:n) redundancy.
debug hccp events	Displays debug messages for all HCCP group interaction.
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug hccp timing	Displays debug messages for the timing of HCCP events.

debug hccp rfswitch

To debug messages related to the Cisco NGRFSW-ADV, use the **debug hccp rfswitch** command in privileged EXEC mode. To stop debugging, use the **no** form of this command.

debug hccp rfswitch [**hello**]

no debug hccp rfswitch [**hello**]

Syntax Description

hello	Displays debug messages related to HELLO messages of the Cisco NGRFSW-ADV.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCG	This command was introduced.

Usage Guidelines

To debug messages related to HELLO messages of the Cisco NGRFSW-ADV, use the **debug hccp rfswitch hello** command. To debug messages related to a Cisco NGRFSW-ADV event, message, and failure, use the **debug hccp rfswitch** command.

Examples

The following example shows how to debug the Cisco NGRFSW-ADV event, message, and failure.

```
Router# debug hccp rfswitch
*Jun 26 22:43:56.434: RFSW: snd msg(size 8): 00 02 00 00 16 9B 6C 16
*Jun 26 22:43:57.058: RFSW: rcv msg (size 1291): 80 02 00 00 16 9B 41 EE 00 00 04 FF 00 D0
00 4B 01 01 00 03 02 03 00 0D 02 03 00 10 02 03 00 04 04 00 00 07 B8 05 0B 41
*Jun 26 22:43:59.374: RFSW: release watch boolean for type 0x02 when receive the reply
*Jun 26 22:44:05.946: RFSW: snd msg(size 8): 00 0A 00 00 16 9F 6D DE
*Jun 26 22:44:05.966: RFSW: rcv msg (size 64): 80 0A 00 00 16 9F 43 F3 00 00 00 34 00 00
19 02 00 02 61 73 76 33 2E 30 00 00 00 B8 8E 00 02 61 63 76 33 2E 30 00 00 00
*Jun 26 22:44:05.970: RFSW: release watch boolean for type 0x0A when receive the reply
*Jun 26 22:44:18.006: RFSW: snd msg(size 8): 00 03 00 00 16 A4 A0 36
*Jun 26 22:44:18.050: RFSW: rcv msg (size 92): 80 03 00 00 16 A4 2D 3A 00 00 00 50 40 6A
40 6A 40 6A 40 6A 40 6A 09 4F 09 4F 09 4F 09 4F 09 4F 09 38 09 38 09 38 09 38
*Jun 26 22:44:18.050: RFSW: release watch boolean for type 0x03 when receive the reply
```

The following example shows how to debug messages related to the HELLO messages of the Cisco NGRFSW-ADV.

```
Router# debug hccp rfswitch hello
*Jun 26 22:41:45.678: RFSW: snd msg(size 8): 00 01 00 00 16 6F 9E 64
*Jun 26 22:41:45.694: RFSW: rcv msg (size 54): 80 01 00 00 16 6F DE 38 00 00 00 2A 01 28
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
*Jun 26 22:41:45.694: RFSW: process HELLO message with seq 0x166F, Group: 1F
*Jun 26 22:41:48.678: RFSW: snd msg(size 8): 00 01 00 00 16 70 58 55
*Jun 26 22:41:48.694: RFSW: rcv msg (size 54): 80 01 00 00 16 70 A6 47 00 00 00 2A 01 28
80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
*Jun 26 22:41:48.694: RFSW: process HELLO message with seq 0x1670, Group: 1F
```

Associated Features

The **debug hccp rfswitch** command is used to debug the Cisco uBR Advanced RF Switch (NGRFSW-ADV). See Cisco uBR Advanced RF Switch Software Configuration Guide.

Related Commands

Command	Description
rf-switch auxport enable	Enables the AUX port on the Cisco NGRFSW-ADV.

debug hccp sync

To display debugging messages for Hot Standby Connection-to-Connection Protocol (HCCP) SYNC activity, use the **debug hccp sync** command in privileged EXEC mode. To disable the debug message output, use the **no** form of the command.

debug hccp sync

debug hccp sync cable [**bpi** | **classifier** | **clear-cm** | **cm** | **qos-profile** | **cpe-management** | **host** | **offered-band** | **phs** | **qosparam** | **ranged-list** | **service-flow** | **sid** | **tlvs** | **ucd**]

no debug hccp sync

no debug hccp sync cable [**bpi** | **classifier** | **clearn-cm** | **cm** | **qos-profile** | **cpe-management** | **host** **offered-band** | **phs** | **qosparam** | **ranged-list** | **service-flow** | **sid** | **tlvs** | **ucd**]

Syntax Description

cable	(Optional) Displays debugging for SYNC messages on the cable interface.
bpi	(Optional) Displays debugging for SYNC messages about BPI configuration and operation.
classifier	(Optional) Displays debugging for SYNC messages about classifier coordination.
clear-cm	(Optional) Displays debugging for CLEAR-CM messages that are sent between interfaces to sync the CM databases when the **clear cable modem** command is given.
cm	(Optional) Displays debugging for CM-related SYNC messages.
qos-profile	(Optional) Displays debugging for SYNC messages related to quality of service (QoS) profile messages generated by the CM.
cpe-management	(Optional) Displays debugging for SYNC messages that concern CPE-related parameters, such as MAX CPE, MAX CPE IP, and max learnable addresses.
host	(Optional) Displays debugging for host-related SYNC messages.
offered-band	(Optional) Displays debugging messages for CMTS-generated offered band messages.
phs	(Optional) Displays debugging for SYNC messages about PHS values.
qosparam	(Optional) Displays debugging for SYNC messages about QoS and service templates.
ranged-list	(Optional) Displays debugging for SYNC messages about ranged list messages.
service-flow	(Optional) Displays debugging for SYNC messages about service flows.
sid	(Optional) Displays debugging for SYNC messages that concern service IDs (SIDs).
tlvs	(Optional) Displays any type, length, or value errors in SYNC messages.
ucd	(Optional) Displays debugging for SYNC messages that concern DOCSIS upstream channel descriptor (UCD) messages.

Command Default

Debug is disabled and messages for HCCP sync activity are not displayed.

Command History

Release	Modification
12.1(3a)EC	This command was introduced.
12.2(4)XF1, 12.2(4)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR10012 router. Also, the **cable** message-specific options were added to allow the debugging of a particular set of messages without overloading the console.
12.2(8)BC1	The **offered-band** option was added to add in debugging problems with blind-hopping.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router.
12.2(15)BC2	The **cpe-management** option was added.
12.3BC	The **clear-cm** and **cm-qos-profile** options were added.

Usage Guidelines

You must activate the **debug hccp events** command before the **debug hccp sync** command will generate any debug message output.

Examples

The following example shows typical output for the **debug hccp sync** command:

```
Router# debug hccp events
Router# debug hccp sync

Sep  7 09:57:25.215:HCCP 1 0<-1:SYNC Teach tran 88 type DOCSIS10, tran_sync 82
Sep  7 09:57:25.215:HCCP 1 0->1:SYNC_ACK Learn tran 88
Sep  7 09:57:25.219:DOCSIS10_QOS:qos 1
```

The following example shows typical debugging messages for the **debug hccp sync cable offered-band** command:

```
Router# debug hccp events
Router# debug hccp sync cable offered-band

CMTS OFFERED BAND sync messages debugging is on

Apr 24 17:17:03.935: HCCP: send offered_band data for US 0
          freq 12400000 chnnl-width 1600000 ip-pwr-lvl 0,
          spec-grp 12, modulation 1, awacs NO,
          shared spectrum NO
Apr 24 17:17:03.935: assign from non-shared spectrum group
Apr 24 17:17:03.935: cmts_freqhop_upd(0x60D5FDB0, 1, 12400000, 0)
Apr 24 17:17:04.067: HCCP of offered band
          HCCP: get offered_band data for US 0
            freq 12400000 chnnl-width 1600000 ip-pwr-lvl 0,
            spec-grp 12, modulation 1, awacs NO,
            shared spectrum NO
Apr 24 17:17:16.467: HCCP: send offered_band data for US 0
          freq 12400000 chnnl-width 1600000 ip-pwr-lvl 0,
          spec-grp 12, modulation 1, awacs NO,
          shared spectrum NO
Apr 24 17:17:16.467: HCCP: send offered_band data for US 1
          freq 10800000 chnnl-width 1600000 ip-pwr-lvl 0,
          spec-grp 12, modulation 1, awacs NO,
          shared spectrum NO
Apr 24 17:17:16.467: HCCP: send offered_band data for US 2
          freq 11600000 chnnl-width 3200000 ip-pwr-lvl 0,
          spec-grp 12, modulation 1, awacs NO,
          shared spectrum NO
```

```
Apr 24 17:17:16.467: HCCP: send offered_band data for US 3
          freq 11600000 chnnl-width 3200000 ip-pwr-lvl 0,
          spec-grp 12, modulation 2, awacs NO,
          shared spectrum NO
Apr 24 17:17:16.467: HCCP of offered band
          HCCP: get offered_band data for US 0
            freq 12400000 chnnl-width 1600000 ip-pwr-lvl 0,
            spec-grp 12, modulation 1, awacs NO,
            shared spectrum NO
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp channel-switch	Displays debug messages related to an RF or channel switch that is being used for HCCP N+1 (1:n) redundancy.
debug hccp events	Displays debug messages for all HCCP group interaction.
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp timing	Displays debug messages for the timing of HCCP events.
debug packetcable hccp	Enables debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations.

debug hccp timing

To display debug messages for the timing of Hot Standby Connection-to-Connection Protocol (HCCP) events, use the **debug hccp timing** command in privileged EXEC mode. To disable the debug message output, use the **no** form of this command.

debug hccp timing [**if-config**]

no debug hccp timing [**if-config**]

Syntax Description

if-config	(Optional) Displays debugging messages showing the timing of the reconfiguration of cable interfaces during HCCP redundancy operations.

Command Default

Debug is disabled and messages for the timing of HCCP events are not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.1(3a)EC	This command was introduced.
12.2(4)XF1, 12.2(4)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR10012 router.
12.2(11)BC1	Support was added for the N+1 (1:n) RF Switch with the Cisco uBR7246VXR router.
12.2(15)BC1	The **if-config** option was added.
12.3(21)BC	This command is obsolete on the Cisco uBR7246VXR router.

Usage Guidelines

You must activate the **debug hccp events** command before the **debug hccp timing** command will generate any debug message output.

Examples

The following example shows typical output for the **debug hccp timing** command:

```
Router# debug hccp events
Router# debug hccp timing

HCCP timing measurement debugging is on

May 31 10:21:07.609 HCCP P is busy. Deactivating 1 6
May 31 10:21:07.609 HCCP P is busy. Deactivating 2 6
May 31 10:21:08.705 HCCP hwif_goingdown for Cable8/1/0. Deactivate 1 6
May 31 10:21:08.705 HCCP hwif_goingdown for Cable8/1/1. Deactivate 2 6
May 31 10:21:08.773 HCCP 2 6 Working: become standby - 68 msec
May 31 10:21:08.793 HCCP 1 6 Working: become standby - 20 msec
May 31 10:21:10.730 HCCP 1 1 Working: turn on "uc" - 8 msec
May 31 10:21:10.730 HCCP 1 1 Working: turn on "nru" - 0 msec
May 31 10:21:10.734 HCCP 1 1 Working: become active - 4 msec
```

```
May 31 10:21:10.774 HCCP 2 1 Working: turn on "uc" - 52 msec
May 31 10:21:10.774 HCCP 2 1 Working: turn on "nru" - 0 msec
May 31 10:21:10.774 HCCP 2 1 Working: become active - 0 msec
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/0. Deactivate 1 1
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/0. Deactivate 1 6
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/0. Deactivate 1 3
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/0. Deactivate 1 2
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/0. Deactivate 1 5
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/0. Deactivate 1 4
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/1. Deactivate 2 1
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/1. Deactivate 2 3
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/1. Deactivate 2 6
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/1. Deactivate 2 2
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/1. Deactivate 2 4
May 31 10:21:12.350 HCCP hwif_goingdown for Cable5/1/1. Deactivate 2 5
May 31 10:21:13.726 HCCP 1 1 Protect: turn off "uc" - 1972 msec
May 31 10:21:13.790 HCCP 2 1 Protect: turn off "uc" - 2036 msec
May 31 10:21:14.422 HCCP 1 1 Protect: turn off "nru" - 696 msec
May 31 10:21:14.422 HCCP 1 1 Protect: unload config (if) - 0 msec
May 31 10:21:14.438 HCCP 1 1 Protect: unload config (subif) - 16 msec
May 31 10:21:14.702 HCCP 1 1 Protect: unload config (ds) - 264 msec
May 31 10:21:14.702 HCCP 1 1 Protect: become standby - 0 msec
May 31 10:21:16.078 HCCP 2 1 Protect: turn off "nru" - 2288 msec
May 31 10:21:16.078 HCCP 2 1 Protect: unload config (if) - 0 msec
May 31 10:21:16.078 HCCP 2 1 Protect: unload config (subif) - 0 msec
May 31 10:21:16.599 HCCP 2 1 Protect: unload config (ds) - 520 msec
May 31 10:21:16.599 HCCP 2 1 Protect: become standby - 0 msec
May 31 10:21:17.014 HCCP: P missed hello ack in LEARN state and is locked. Deactivate 4 1
May 31 10:21:17.014 HCCP 4 1 Protect: turn off "rfswitch" - 52 msec
May 31 10:21:17.593 HCCP 3 1 Working: turn on "rfswitch" - 0 msec
May 31 10:21:17.593 HCCP 3 1 Working: become active - 0 msec
May 31 10:21:18.112 HCCP 1 1 Protect: load config (if) - 0 msec
May 31 10:21:18.112 HCCP 1 1 Protect: load config (subif) - 4 msec
May 31 10:21:18.331 HCCP 1 1 Protect: load config (ds) - 100 msec
May 31 10:21:18.331 HCCP 2 1 Working: turn off "rfswitch" - 0 msec
May 31 10:21:18.331 HCCP 2 Cable5/0/1 Protect: resolve conflict Learn->Teach
May 31 10:21:18.331 HCCP 2 1 Protect: load config (if) - 0 msec
May 31 10:21:18.331 HCCP 2 1 Protect: load config (subif) - 0 msec
May 31 10:21:19.691 HCCP 2 1 Protect: load config (ds) - 76 msec
May 31 10:21:20.112 HCCP 2 1 Protect: turn on "rfswitch" - 48 msec
May 31 10:21:20.112 HCCP 2 1 Protect: become active - 0 msec
May 31 10:21:20.112 HCCP 2 1 Protect: load config (ds) - 76 msec
May 31 10:21:20.112 HCCP 2 1 Protect: turn on "rfswitch" - 48 msec
May 31 10:21:20.112 HCCP 2 1 Protect: become active - 0 msec
```

The following example shows typical output for the **debug hccp timing if-config** command:

```
Router# debug hccp events
Router# debug hccp timing if-config

HCCP Timing measurements messages of (UN)LOAD IF config CLI is on

HCCP 1 1 Working: unload config (ds) - 112 msec
HCCP 1 1 Protect: load config (ds) - 123 msec
HCCP 1 1 Protect: load config (chnl set freq) - 35 msec
```

Related Commands

Command	Description
debug hccp authentication	Displays authentication debug messages for HCCP groups.
debug hccp channel-switch	Displays debug messages related to an RF or channel switch that is being used for HCCP N+1 (1:n) redundancy.
debug hccp events	Displays debug messages for all HCCP group interaction.
debug hccp inter-db	Displays debug messages for the inter-database events during HCCP operations.
debug hccp plane	Displays debug messages for HCCP-related messages sent between the router's control plane and data backplane.
debug hccp sync	Displays debug messages for HCCP synchronization messages.

debug hw-module all upgrade

To enable debug messages for field-programmable devices (FPDs), use the **debug hw-module all upgrade** command in privileged EXEC configuration mode. To disable debug messages, use the **no** form of the command.

debug hw-module all upgrade [**error** | **event**]

no debug hw-module all upgrade [**error** | **event**]

Syntax Description

all	Enable debug messaging for all supported modules in the system.
error	(Optional) Enables display of FPD upgrade error messages.
event	(Optional) Enables display of FPD upgrade event messages.

Defaults

No default behavior or values

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXE	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SCB	This command was integrated into Cisco IOS Release 12.2(33)SCB. The FPD image upgrade is supported only for the SPAs inserted in the Cisco SIP-600 on a Cisco uBR10012 router.

Usage Guidelines

The **debug hw-module all upgrade** command is intended for use by Cisco Systems technical support personnel.

If you attempt to use this command without a SPA installed, or with an incompatible SPA installed, the keyword options are not provided.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use **debug** commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco Systems technical support personnel. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased **debug** command processing overhead will affect system use.

For more information about FPD upgrades on SPA interface processors (SIPs) and shared port adapters (SPAs), refer to the *Cisco 7600 Series Router SIP, SSC, and SPA Software Configuration Guide*.

Examples

The following example enables FPD upgrade debug messages for all supported card types on the Cisco 7600 series router:

```
Router# debug hw-module all upgrade
```

debug hw-module bay

To enable debugging information for a SPA, use the **debug hw-module bay** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

Cisco IOS Releases 12.3(23)BC and 12.2(33)SCA

debug hw-module bay *slot*/*subslot*/*bay* {**commands** | **errors** | **events** | **interrupts** | **oir** | **periodic**}

no debug hw-module bay *slot*/*subslot*/*bay* {**commands** | **errors** | **events** | **interrupts** | **oir** | **periodic**}

Cisco IOS Release 12.2(33)SCB

debug hw-module bay *slot*/*bay*/*port* {**commands** | **errors** | **events** | **interrupts** | **oir** | **periodic**}

no debug hw-module bay *slot*/*bay*/*port* {**commands** | **errors** | **events** | **interrupts** | **oir** | **periodic**}

SyntaxDescription

slot	The slot where a SIP resides. On the Cisco uBR10012 router, slots 1 and 3 can be used for a SIP.
subslot	The subslot where the Wideband SIP resides. On the Cisco uBR10012 router, subslot 0 is always specified.
bay	The bay in the SIP where a SPA is located. Valid values are 0 (upper bay) and 1 (lower bay).
port	Specifies the interface number on the SPA.
commands	Enables debug messages for control plane configuration and commands on a SPA.
errors	Enables debug messages for error handling and race conditions on a SPA.
events	Enables debug messages for control plane event notifications on a SPA.
interrupts	Enables debug messages for interrupt handling on a SPA.
oir	Enables debug messages for online insertion and removal (OIR) processing on a SPA.
periodic	Enables debug messages for periodic processing on a SPA.

Command Default

No Wideband SPA debug messages are enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(21)BC	This command was introduced for the Cisco uBR10012 router.
12.2(33)SCA	This command was integrated into Cisco IOS Release 12.2(33)SCA.
12.2(33)SCB	This command was modified to change the addressing format for a SPA from *slot/subslot/bay* to *slot/bay/port.*

Usage Guidelines

The **debug hw-module bay** command is intended for use by Cisco technical support personnel.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use **debug** commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support personnel. Moreover, it is best to use **debug** commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased **debug** command processing overhead will affect system use.

If you attempt to use a **debug hw-module bay** command without a SPA installed, the keyword options are not provided.

Examples

The following example shows how to enable debug messages for error handling and race conditions:

```
Router# debug hw-module bay 1/0/0 errors
```

Related Commands

Command	Description
debug c10k-jacket	Enables debugging information for the Wideband SIP.
debug cable fn	Enables debugging information for cable fiber nodes.
debug cable wbcmts	Enables debugging information for the wideband CMTS.

debug nls

To debug the NLS request, use the **debug nls** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug nls verbose

no debug nls verbose

Syntax Description

verbose	(Optional) Displays detailed debugging information.

Command Default

Debug is disabled and NLS messages are not displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(21a)BC3	This command was introduced.

Examples

The following example shows enabling the **debug nls** command:

```
Router# debug nls
```

Related Commands

Command	Description
nls	Enables Network Layer signalling (NLS) functionality.

debug packetcable all

To display debugging messages for all PacketCable events, use the **debug packetcable all** command in privileged EXEC mode. To turn off Packetcable debugging, use the **no** form of this command.

debug packetcable all [**detail**]

no debug packetcable all [**detail**]

Syntax Description

detail	Displays additional debug messages for specific events, such as the content of PacketCable headers and messages. **Note** Release 12.2(15)BC1 removed this option and replaced it with the **verbose** option of the **debug packetcable subscriber** command.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(8)BC2	This command was introduced for the Cisco uBR7200 series universal broadband router.
12.2(15)BC1	Support was added for the Cisco uBR10012 universal broadband router. Also, the **detail** option was removed, and this command now requires that you enable PacketCable debugging for one or more IP addresses, using the **debug packetcable subscriber** command, before displaying any output. In addition, **packetcable gate coordination** support was removed for all platforms because that debugging option was removed from the release.

Usage Guidelines

This command enables the display of all PacketCable debugging messages, and is equivalent to giving the other **debug packetcable gate** commands individually. Use this command to get complete debugging about a particular event, such as the creation or termination of a call.

In Cisco IOS Release 12.2(15)BC1 and later releases, you must first use the **debug packetcable subscriber** command to enable PacketCable debugging for one or more IP addresses before the **debug packetcable all** command displays any output.

Note To display detailed debugging output in Cisco IOS Release 12.2(15)BC1 and later releases, use the **verbose** option when you enable PacketCable debugging output using the **debug packetcable subscriber** command.

Caution This command can generate a significant amount of debugging information, and the volume of this information could impact system performance on active PacketCable networks. Do not use this command except in lab or test networks, or where the need to troubleshoot problems makes the impact on system performance acceptable.

Examples

The following example shows typical output for the **debug packetcable all** command for the events starting with a call is placed up until the gates go into the committed state:

```
Router# debug packetcable all
PacketCable Client:
  Pktcbl gate control msgs debugging is on
  Pktcbl gate coord msgs debugging is on
  Pktcbl commit msgs debugging is on
  Pktcbl event msgs debugging is on
  Pktcbl rsvp-to-docsis msgs debugging is on
  Pktcbl gate database changes debugging is on

Router#

Jul 25 14:28:13.442 UTC: PktCbl(cops): Received a COPS DEC message, flags is 0x1
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Received GATE ALLOC message, tid=0x4
Jul 25 14:28:13.442 UTC: Pktcbl(gdb): Created new Subscriber IE for 3.3.1.3
Jul 25 14:28:13.442 UTC: Pktcbl(gdb): Updated subscriber IE [subs addr: 3.3.1.3, gate: 38]
Jul 25 14:28:13.442 UTC: Pktcbl(gdb): Created gate IE, gateid = 38, total gates: 1
Jul 25 14:28:13.442 UTC: Pktcbl(gdb): Started gate [id 38] timer t0 [30 sec]
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Building GCP message, added obj TRANSACTION ID      ; len:8 padding:0
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Building GCP message, added obj SUBSCRIBER ID (IPV4); len:8 padding:0
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Building GCP message, added obj GATE ID             ; len:8 padding:0
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Building GCP message, added obj ACTIVITY COUNT      ; len:8 padding:0
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Building GCP message, added obj GCOORD PORT         ; len:8 padding:0
Jul 25 14:28:13.442 UTC: Pktcbl(gcp): Built GCP message, GATE ALLOC ACK , length: 40, copsLen 68
Jul 25 14:28:13.658 UTC: PktCbl(cops): Received a COPS DEC message, flags is 0x1
Jul 25 14:28:13.658 UTC: Pktcbl(gcp): Received GATE SET message, tid=0x6
Jul 25 14:28:13.658 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 2
Jul 25 14:28:13.658 UTC: Pktcbl(gdb): GateID: 38, changed state to: 2
Jul 25 14:28:13.658 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 2
Jul 25 14:28:13.658 UTC: Pktcbl(gdb): Stopped gate [id 38] timer [type 1]
Jul 25 14:28:13.658 UTC: Pktcbl(gdb): Started gate [id 38] timer t1 [180000 msec]
Jul 25 14:28:13.658 UTC: Pktcbl(gcp): Building GCP message, added obj TRANSACTION ID      ; len:8 padding:0
Jul 25 14:28:13.658 UTC: Pktcbl(gcp): Building GCP message, added obj SUBSCRIBER ID (IPV4); len:8 padding:0
Jul 25 14:28:13.658 UTC: Pktcbl(gcp): Building GCP message, added obj GATE ID             ; len:8 padding:0
Jul 25 14:28:13.658 UTC: Pktcbl(gcp): Building GCP message, added obj ACTIVITY COUNT      ; len:8 padding:0
Jul 25 14:28:13.658 UTC: Pktcbl(gcp): Built GCP message, GATE SET ACK        , length: 32, copsLen 60
Jul 25 14:28:13.822 UTC: PktCbl(d2r): DSA-REQ received, gateid: 38
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): handle=38 dir=0 type=3 (get gate flowspec)
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): handle=38 dir=1 type=2 (get gate flowspec)
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): handle=38 dir=0 type=2 (get gate flowspec)
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): handle=38 dir=1 type=3 spec=61DA13A8 (set gate flowspec)
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): handle=38 dir=0 type=3 spec=61DA13C4 (set gate flowspec)
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 3
Jul 25 14:28:13.822 UTC: Pktcbl(gdb): GateID: 38, changed state to: 3
Jul 25 14:28:13.826 UTC: Pktcbl(em): Send msg: type=QOS_RESERVE dqos=0x628DE278
Jul 25 14:28:13.834 UTC: Pktcbl(gdb): Get gate spec info, handle=38 dir=0 spec=61DA1420
Jul 25 14:28:23.950 UTC: Pktcbl(g2g): received GATE_OPEN message, tid 2
Jul 25 14:28:23.950 UTC: Pktcbl(g2g): sending g2g GATE_OPEN_ACK message to 192.168.80.15:1812 from
192.168.80.1:50048
Jul 25 14:28:23.950 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 5
Jul 25 14:28:23.950 UTC: Pktcbl(gdb): GateID: 38, changed state to: 5
Jul 25 14:28:23.950 UTC: Pktcbl(gdb): Started gate [id 38] timert2 [2000 ms]
Jul 25 14:28:24.030 UTC: Pktcbl(gdb): Get gate spec info, handle=38 dir=1 spec=61DA1268
Jul 25 14:28:24.034 UTC: PktCbl(d2r): DSC-REQ received, gateid: 38
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): Get gate spec info, handle=38 dir=0 spec=61DA1308
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): handle=38 dir=0 type=6 (get gate flowspec)
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): handle=38 dir=1 type=3 (get gate flowspec)
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): handle=38 dir=0 type=3 (get gate flowspec)
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): handle=38 dir=1 type=6 spec=61DA13A8 (set gate flowspec)
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): handle=38 dir=0 type=6 spec=61DA13C4 (set gate flowspec)
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 4
```

```
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): GateID: 38, changed state to: 6
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 4
Jul 25 14:28:24.034 UTC: PktCbl(Commit): state: id=38
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 6
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): Cancelled gate [id 38] timer t2
Jul 25 14:28:24.034 UTC: Pktcbl(gdb): Cancelled gate [id 38] timer t1
Jul 25 14:28:24.034 UTC: Pktcbl(em): Send msg: type=QOS_COMMIT dqos=0x628DE278
Jul 25 14:28:24.466 UTC: Pktcbl(g2g): received GATE_OPEN message, tid 2
Jul 25 14:28:24.466 UTC: Pktcbl(g2g): Duplicate GATE-OPEN for gateID: 38, not processing it, sending
GATE-OPEN-ACK
Jul 25 14:28:24.466 UTC: Pktcbl(g2g): sending g2g GATE_OPEN_ACK message to 192.168.80.15:1812 from
192.168.80.1:50048
Jul 25 14:28:25.998 UTC: Pktcbl(g2g): received GATE_OPEN message, tid 2
Jul 25 14:28:25.998 UTC: Pktcbl(g2g): Duplicate GATE-OPEN for gateID: 38, not processing it, sending
GATE-OPEN-ACK
Jul 25 14:28:25.998 UTC: Pktcbl(g2g): sending g2g GATE_OPEN_ACK message to 192.168.80.15:1812 from
192.168.80.1:50048
Jul 25 14:28:55.446 UTC: PktCbl(d2r): DSD-REQ received, gateid: 38
Jul 25 14:28:55.446 UTC: Pktcbl(gdb): Gate delete requested, gateid = 38
Jul 25 14:28:55.446 UTC: Pktcbl(gdb): request to transition gate ID 38 to state 7
Jul 25 14:28:55.450 UTC: Pktcbl(gdb): GateID: 38, changed state to: 7
Jul 25 14:28:55.450 UTC: Pktcbl(em): Send msg: type=QOS_RELEASE dqos=0x628DE278
Jul 25 14:28:55.450 UTC: Pktcbl(gdb): Started gate [id 38] timer t5 [500 msec]
Jul 25 14:28:55.450 UTC: Pktcbl(g2g): sending g2g GATE_CLOSE message to 172.22.79.22:1812 from
172.22.79.44:50048
Jul 25 14:28:55.950 UTC: Pktcbl: Timer event
Jul 25 14:28:55.950 UTC: Pktcbl(g2g): Timer T5 expired, gateID: 38
Jul 25 14:28:55.950 UTC: Pktcbl(gdb): Started gate [id 38] timer t5 [500 msec]
Jul 25 14:28:55.950 UTC: Pktcbl(g2g): Retransmitting message: GATE_CLOSE, gateID: 38
Jul 25 14:28:56.450 UTC: Pktcbl: Timer event
Jul 25 14:28:56.450 UTC: Pktcbl(g2g): Timer T5 expired, gateID: 38
Jul 25 14:28:56.450 UTC: Pktcbl(gdb): Started gate [id 38] timer t5 [500 msec]
Jul 25 14:28:56.450 UTC: Pktcbl(g2g): Retransmitting message: GATE_CLOSE, gateID: 38
Jul 25 14:28:56.950 UTC: Pktcbl: Timer event
Jul 25 14:28:56.950 UTC: Pktcbl(g2g): Timer T5 expired, gateID: 38
Jul 25 14:28:56.950 UTC: Pktcbl(gdb): Started gate [id 38] timer t5 [500 msec]
Jul 25 14:28:56.950 UTC: Pktcbl(g2g): Retransmitting message: GATE_CLOSE, gateID: 38
Jul 25 14:28:57.450 UTC: Pktcbl: Timer event
Jul 25 14:28:57.450 UTC: Pktcbl(g2g): Timer T5 expired, gateID: 38
Jul 25 14:28:57.450 UTC: Pktcbl(gdb): Started gate [id 38] timer t5 [500 msec]
Jul 25 14:28:57.450 UTC: Pktcbl(g2g): Retransmitting message: GATE_CLOSE, gateID: 38
Jul 25 14:28:57.950 UTC: Pktcbl: Timer event
Jul 25 14:28:57.950 UTC: Pktcbl(g2g): Timer T5 expired, gateID: 38
Jul 25 14:28:57.950 UTC: Pktcbl(gdb): Deleting gate IE, gateid = 38, total gateids [1]
Jul 25 14:28:57.950 UTC: Pktcbl(gdb): Deleting subs IE, subs id = 3.3.1.3

Router#
```

Related Commands

Command	Description
debug cops	Displays debugging messages related to the COPS protocol.
debug packetcable cops	Enables debugging of Common Open Policy Service (COPS) messages and errors that are related to PacketCable operations on the Cisco CMTS router.
debug packetcable gate	Displays general debugging messages related to specific PacketCable gate events and messages.
debug packetcable hccp	Enables debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations.

Command	Description
debug packetcable ipc	Enables interprocess communications (IPC) messages related to PacketCable operations on the Cisco uBR10012 router.
debug packetcable subscriber	Enables PacketCable debugging for a specific subscriber IP address.

debug packetcable cops

To enable debugging of Common Open Policy Service (COPS) messages and errors that are related to PacketCable operations on the Cisco CMTS router, use the **debug packetcable cops** command in privileged EXEC mode. To turn off this debugging, use the **no** form of this command.

debug packetcable cops

no debug packetcable cops

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)BC1	This command was introduced for the Cisco uBR7246VXR and Cisco uBR10012 universal broadband routers.

Usage Guidelines

This command enables debugging for the COPS messages and errors that are related to PacketCable operations. It produces output that is similar to the **debug cops** command, except that the output for this command is limited to the use of the COPS protocol to support PacketCable connections.

Examples

The following example shows typical debugging output for the **debug packetcable cops** command:

```
Router# debug packetcable cops
Pktcbl COPS msgs debugging is on

Router#

Aug 21 12:41:23 UTC: Pktcbl(cops): Reallocating message buffer to 256 bytes
Aug 21 12:41:23 UTC: Pktcbl(cops): COPS header not valid
Aug 21 12:41:23 UTC: Pktcbl(cops): Not a valid COPS object [class: 31]
Aug 21 12:41:23 UTC: Pktcbl(cops): Last object [class: 31] not fully contained in COPS msg
Aug 21 12:41:23 UTC: Pktcbl(cops): No entry for the server! (client_accept)

Aug 21 12:43:41 UTC: PktCbl(cops): Received callback [code 7, handle: 0x64911528] from
COPS engine
Aug 21 12:43:41 UTC: PktCbl(cops): Remove GC (64911528)
Aug 21 12:43:42 UTC: PktCbl(cops): Received callback [code 8, handle: 0x64463AA4] from
COPS engine
Aug 21 12:43:42 UTC: PktCbl(cops): Incoming TCP connect from GC (1.10.90.1)
Aug 21 12:43:42 UTC: PktCbl(cops): Received callback [code 4, handle: 0x64463AA4] from
COPS engine
Aug 21 12:43:42 UTC: PktCbl(cops): Received COPS CLIENT_ACCEPT [tcp handle: 0x64463AA4]
```

Related Commands

Command	Description
debug cops	Displays debugging messages related to the COPS protocol.
debug packetcable all	Displays general debugging messages for all PacketCable events.
debug packetcable gate	Displays general debugging messages related to specific PacketCable gate events and messages.
debug packetcable hccp	Enables debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations.
debug packetcable ipc	Enables interprocess communications (IPC) messages related to PacketCable operations on the Cisco uBR10012 router.
debug packetcable subscriber	Enables PacketCable debugging for a specific subscriber IP address.

debug packetcable gate

To display general debugging messages for specific PacketCable gate events and messages, use the **debug packetcable gate** command in privileged EXEC mode. To turn off Packetcable debugging, use the **no** form of this command.

debug packetcable gate {**commit** | **control** | **coordination** | **database** | **docsis-mapping** | **events** [**process**] } [**detail**]

no debug packetcable gate {**commit** | **control** | **coordination** | **database** | **docsis-mapping** | **events** [**process**] } [**detail**]

Syntax Description

commit	Displays debugging messages for the commit events on each gate.
control	Displays debugging messages for gate control events (allocation, set, and delete).
coordination	Displays debugging messages for gate coordination events (gate close and open) (not supported in Release 12.2(15)BC1 and later releases).
database	Displays debugging messages for all updates to the gate database, including gate creation and deletion, state transitions, and structure updates.
docsis-mapping	Displays debugging messages for the mapping between PacketCable and Data-over-Cable System Interface Specification (DOCSIS) parameters.
events	Displays debugging messages for the event messages that are sent during a voice call.
process	(Optional) Displays non-subscriber-related process event messages (available only when using the **events** option).
detail	Displays additional debug messages for specific events, such as the contents of a PacketCable headers and messages. **Note** Release 12.2(15)BC1 removed this option and replaced it with the **verbose** option of the **debug packetcable subscriber** command.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(8)BC2	This command was introduced for the Cisco uBR7200 series universal broadband router.
12.2(15)BC1	The **coordination** and **detail** options were removed, and the **process** option was added. Support was also added for the Cisco uBR10012 router. In addition, most options no longer display any output until PacketCable debugging has been enabled for one or more IP addresses, using the **debug packetcable subscriber** command.
12.2(15)BC2	The **hccp** option was added.

Usage Guidelines

Use this command to enable debugging for the individual PacketCable functional systems on the Cisco CMTS. As a general rule, selectively use these commands to debug or troubleshoot problems instead of turning on all debugging with the **debug packetcable all** command.

In Cisco IOS Release 12.2(15)BC1 and later, you must also enable debugging for one or more IP addresses, using the **debug packetcable subscriber** command, before the **debug packetcable gate** command displays any output. The **debug packetcable gate events process** command is an exception to this, however, because it does not display subscriber-related information.

Note The **debug packetcable all** command is equivalent to repeatedly giving the **debug packetcable gate** command with each of its possible options.

Tip To display detailed debugging output in Cisco IOS Release 12.2(15)BC1 and later releases, use the **verbose** option when you enable PacketCable debugging output using the **debug packetcable subscriber** command.

Examples

This section shows typical examples for each type of debugging message.

Commit Event Debugging

The following example shows typical debugging messages for the commit events on a gate. Omit the **verbose** option if you do not want to display the header and message contents.

```
Router# debug packetcable subscriber 192.168.78.21 verbose
Router# debug packetcable commit
Pktcbl commit msgs (detail) debugging is on

Jul 25 16:13:34 UTC: Pktcbl(Commit): Commit msg from [addr/port 192.168.78.21/50024],
length 40
Jul 25 16:13:34 UTC: Pktcbl(Commit): (commit) - Calling DOCSIS QoS module Commit
Jul 25 16:13:34 UTC: Pktcbl(Commit): rsvp->docsis: Received a DOCSIS_COMMIT request
Jul 25 16:13:34 UTC: Pktcbl(Commit): Sending COMMIT response [241] [src
192.168.78.1/50024, dest 192.168.78.21/50024]
Jul 25 16:13:34 UTC: Commit Header
Jul 25 16:13:34 UTC:     ver (0x)    : 01
Jul 25 16:13:34 UTC:     msg type    : COMMIT_ACK [241]
Jul 25 16:13:34 UTC:     checksum    : 1D 4D
Jul 25 16:13:34 UTC:     sending ttl : 255
Jul 25 16:13:34 UTC:     msg length  : 40
Jul 25 16:13:34 UTC:  Destination 192.168.78.19, Protocol_Id 17, Don't Police , DstPort
20362
Jul 25 16:13:34 UTC:  Source 192.168.78.21, udp_source_port 20380
Jul 25 16:13:34 UTC:  GATE_ID              type 8 length 8 :
Jul 25 16:13:34 UTC:  Id = 2430632
Jul 25 16:13:34 UTC: Pktcbl(Commit): Sending COMMIT response [241] [src
192.168.78.1/50024, dest 192.168.78.21/50024]
Jul 25 16:13:34 UTC: Commit Header
Jul 25 16:13:34 UTC:     ver (0x)    : 01
Jul 25 16:13:34 UTC:     msg type    : COMMIT_ACK [241]
Jul 25 16:13:34 UTC:     checksum    : 1D 4D
Jul 25 16:13:34 UTC:     sending ttl : 255
Jul 25 16:13:34 UTC:     msg length  : 40
Jul 25 16:13:34 UTC:  Destination 192.168.78.19, Protocol_Id 17, Don't Police , DstPort
20362
Jul 25 16:13:34 UTC:  Source 192.168.78.21, udp_source_port 20380
Jul 25 16:13:34 UTC:  GATE_ID              type 8 length 8 :
```

```
Jul 25 16:13:34 UTC:  Id = 2430632
Jul 25 16:13:34 UTC: Pktcbl(Commit): Commit msg from [addr/port 192.168.78.19/50024],
length 40
Jul 25 16:13:34 UTC: Pktcbl(Commit): (commit) - Calling DOCSIS QoS module Commit
Jul 25 16:13:34 UTC: Pktcbl(Commit): Sending COMMIT response [241] [src
192.168.78.1/50024, dest 192.168.78.19/50024]
Jul 25 16:13:34 UTC: Commit Header
Jul 25 16:13:34 UTC:     ver (0x)    : 01
Jul 25 16:13:34 UTC:     msg type    : COMMIT_ACK [241]
Jul 25 16:13:34 UTC:     checksum    : 33 FFFFFF96
Jul 25 16:13:34 UTC:     sending ttl : 255
Jul 25 16:13:34 UTC:     msg length  : 40
Jul 25 16:13:34 UTC:  Destination 192.168.78.21, Protocol_Id 17, Don't Police , DstPort
20380
Jul 25 16:13:34 UTC:  Source 192.168.78.19, udp_source_port 20362
Jul 25 16:13:34 UTC:  GATE_ID              type 8 length 8 :
Jul 25 16:13:34 UTC:  Id = 2359392
```

Gate Control Event Debugging

The following example shows typical debugging messages for gate control events. Omit the **verbose** option if you do not want to display the header and message contents.

```
Router# debug packetcable subscriber 192.168.78.21 verbose
Router# debug packetcable control
Pktcbl gate control msgs (detail) debugging is on

Jul 25 16:23:15 UTC: Pktcbl(gcp): Received a COPS DEC message
Jul 25 16:23:15 UTC: --- Pktcbl(gcp): Received GCP message ------
Jul 25 16:23:15 UTC: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
Jul 25 16:23:15 UTC:     transaction id   : 0x5
Jul 25 16:23:15 UTC:     gcp cmd          : 1 (GATE ALLOC)
Jul 25 16:23:15 UTC: SUBSCRIBER ID (IPV4) : Object.[snum/stype/len 2/1/8]
Jul 25 16:23:15 UTC:     Addr             : 192.168.78.19
Jul 25 16:23:15 UTC: ACTIVITY COUNT       : Object.[snum/stype/len 4/1/8]
Jul 25 16:23:15 UTC:     Count            : 10
Jul 25 16:23:15 UTC: ---------------------------------------
Jul 25 16:23:15 UTC: Pktcbl(gcp): Building GCP message:added obj TRANSACTION ID      ;
len:8 padding:0
Jul 25 16:23:15 UTC: Pktcbl(gcp): Building GCP message:added obj SUBSCRIBER ID (IPV4);
len:8 padding:0
Jul 25 16:23:15 UTC: Pktcbl(gcp): Building GCP message:added obj GATE ID             ;
len:8 padding:0
Jul 25 16:23:15 UTC: Pktcbl(gcp): Building GCP message:added obj ACTIVITY COUNT      ;
len:8 padding:0
Jul 25 16:23:15 UTC: Pktcbl(gcp): Building GCP message:added obj GCOORD PORT         ;
len:8 padding:0
Jul 25 16:23:15 UTC: Pktcbl(gcp): Built GCP message, GATE ALLOC ACK      , length: 40,
copsLen 68
Jul 25 16:23:15 UTC:  --- Pktcbl(gcp): Sending GCP message ------
Jul 25 16:23:15 UTC: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
Jul 25 16:23:15 UTC:     transaction id   : 0x5
Jul 25 16:23:15 UTC:     gcp cmd          : 2 (GATE ALLOC ACK)
Jul 25 16:23:15 UTC: SUBSCRIBER ID (IPV4) : Object.[snum/stype/len 2/1/8]
Jul 25 16:23:15 UTC:     Addr             : 192.168.78.19
Jul 25 16:23:15 UTC: GATE ID              : Object.[snum/stype/len 3/1/8]
Jul 25 16:23:15 UTC:     GateID           : 2359392 (0x240060)
Jul 25 16:23:15 UTC: ACTIVITY COUNT       : Object.[snum/stype/len 4/1/8]
Jul 25 16:23:15 UTC:     Count            : 1
Jul 25 16:23:15 UTC: GCOORD PORT          : Object.[snum/stype/len 12/1/8]
Jul 25 16:23:15 UTC:     gcoord port      : [50048]
Jul 25 16:23:15 UTC:  ---------------------------------------
Jul 25 16:23:15 UTC: --- Pktcbl(gcp): Received GCP message ------
Jul 25 16:23:15 UTC: TRANSACTION ID       : Object.[snum/stype/len 1/1/8]
```

```
Jul 25 16:23:15 UTC:     transaction id   : 0x7
Jul 25 16:23:15 UTC:     gcp cmd          : 4 (GATE SET)
Jul 25 16:23:15 UTC: SUBSCRIBER ID (IPV4) : Object.[snum/stype/len 2/1/8]
Jul 25 16:23:15 UTC:     Addr             : 192.168.78.21
Jul 25 16:23:15 UTC: GATE ID              : Object.[snum/stype/len 3/1/8]
Jul 25 16:23:15 UTC:     GateID           : 2430632 (0x2516A8)
Jul 25 16:23:15 UTC: REMOTE GATE INFO     : Object.[snum/stype/len 6/1/33]
Jul 25 16:23:15 UTC:     gateid           : 2359392
Jul 25 16:23:15 UTC:     addr/port        : 192.168.2.236/0xC380
Jul 25 16:23:15 UTC:     gcoord flag      : 0x0
Jul 25 16:23:15 UTC:     algo             : 100
Jul 25 16:23:15 UTC:     security key     : [F6 B1 AF D2 26 35 76 11 11 73 EB 3D 07 6A 13
B7 ]
Jul 25 16:23:15 UTC: EVENT GNRTN INFO     : Object.[snum/stype/len 7/1/36]
Jul 25 16:23:15 UTC:     primary RKS      : [addr/port 192.168.2.240/26135]
Jul 25 16:23:15 UTC:     secondary RKS    : [addr/port 20.4.5.1/26391]
Jul 25 16:23:15 UTC:     flags            : 0x0
Jul 25 16:23:15 UTC:     billing corr ID  : [3A 12 35 B1 61 38 63 30 65 37 30 32 00 00 00
01 ]
Jul 25 16:23:15 UTC: MEDIA CNX EVENT INFO : Object.[snum/stype/len 8/1/84]
Jul 25 16:23:15 UTC:     called party #   : [39 37 38 32 34 34 31 31 33 33 00 00 00 00 00
00 00 00 00 00 00
Jul 25 16:23:15 UTC:     routing number   : [39 37 38 32 34 34 31 31 33 33 00 00 00 00 00
00 00 00 00 00 00
Jul 25 16:23:15 UTC:     charged number   : [39 37 38 32 34 34 31 31 33 36 00 00 00 00 00
00 00 00 00 00 00
Jul 25 16:23:15 UTC:     locn. routing #  : [39 37 38 32 34 34 31 31 33 36 00 00 00 00 00
00 00 00 00 00 00
Jul 25 16:23:15 UTC: GATE SPEC            : Object.[snum/stype/len 5/1/60]
Jul 25 16:23:15 UTC:     direction        : 1 (UPSTREAM)
Jul 25 16:23:15 UTC:     protocol id      : 17
Jul 25 16:23:15 UTC:     commit flag      : 0x0
Jul 25 16:23:15 UTC:     session class    : 1
Jul 25 16:23:15 UTC:     source           : 192.168.78.21
Jul 25 16:23:15 UTC:     dest             : 192.168.78.19
Jul 25 16:23:15 UTC:     src port         : 20380 (0x4F9C)
Jul 25 16:23:15 UTC:     dest port        : 20362 (0x4F8A)
Jul 25 16:23:15 UTC:     dscp             : 0x78
Jul 25 16:23:15 UTC:     timer t1         : 300000
Jul 25 16:23:15 UTC:     timer t2         : 2000000
Jul 25 16:23:15 UTC:     flowspec # 1     : [r/b/p/m/M
1176256512/1128792064/1176256512/0/200] [R/S: 1176256512/0]
Jul 25 16:23:15 UTC: GATE SPEC            : Object.[snum/stype/len 5/1/60]
Jul 25 16:23:15 UTC:     direction        : 0 (DOWNSTREAM)
Jul 25 16:23:15 UTC:     protocol id      : 17
Jul 25 16:23:15 UTC:     commit flag      : 0x0
Jul 25 16:23:15 UTC:     session class    : 1
Jul 25 16:23:15 UTC:     source           : 192.168.78.19
Jul 25 16:23:15 UTC:     dest             : 192.168.78.21
Jul 25 16:23:15 UTC:     src port         : 20362 (0x4F8A)
Jul 25 16:23:15 UTC:     dest port        : 20380 (0x4F9C)
Jul 25 16:23:15 UTC:     dscp             : 0x78
Jul 25 16:23:15 UTC:     timer t1         : 300000
Jul 25 16:23:15 UTC:     timer t2         : 2000000
Jul 25 16:23:15 UTC:     flowspec # 1     : [r/b/p/m/M
1176256512/1128792064/1176256512/0/200] [R/S: 1176256512/0]
Jul 25 16:23:15 UTC: SDP PARAMS           : Object.[snum/stype/len 11/1/348]
```

Gate Coordination Event Debugging

The following example shows typical debugging messages for gate coordination events. Omit the **verbose** option if you do not want to display the header and message contents.

Note The **coordination** option was removed in Cisco IOS Release 12.2(15)BC1.

```
Router# debug packetcable subscriber 192.168.78.21 verbose
Router# debug packetcable coordination
Pktcbl gate coord msgs (detail) debugging is on

Jul 25 14:35:17.661 UTC: Pktcbl(g2g): received GATE_OPEN message, tid 10
Jul 25 14:35:17.661 UTC: --- Pktcbl(g2g): Received G2G message ------
Jul 25 14:35:17.665 UTC: G2G message header
Jul 25 14:35:17.665 UTC:     Type          : GATE_OPEN
Jul 25 14:35:17.665 UTC:     TransactionID : 10
Jul 25 14:35:17.665 UTC:     Msg Length    : 28
Jul 25 14:35:17.665 UTC:     Auth (hex)    : 18 88 CB A2 55 9A 6B 19 3D DB 7B AD 83 FC FC
E9
Jul 25 14:35:17.665 UTC: GATE ID              : Object.[type/len 224/8]
Jul 25 14:35:17.665 UTC:     GateID          : 102 (0x66)
Jul 25 14:35:17.665 UTC: ---------------------------------------
Jul 25 14:35:17.665 UTC: Pktcbl(g2g): auth_request: 0x18 88 CB A2 55 9A 6B 19 3D DB 7B AD
83 FC FC E9
Jul 25 14:35:17.665 UTC: Pktcbl(g2g): sending g2g GATE_OPEN_ACK message to
192.168.80.15:1812 from 192.168.80.1:50048
Jul 25 14:35:17.665 UTC: --- Pktcbl(g2g): Sending G2G message ------
Jul 25 14:35:17.665 UTC: G2G message header
Jul 25 14:35:17.665 UTC:     Type          : GATE_OPEN_ACK
Jul 25 14:35:17.665 UTC:     TransactionID : 10
Jul 25 14:35:17.665 UTC:     Msg Length    : 28
Jul 25 14:35:17.665 UTC:     Auth (hex)    : 1A CA 3E 2D B4 D3 18 B9 4D DC 06 C8 98 71 E8
31
Jul 25 14:35:17.665 UTC: GATE ID              : Object.[type/len 224/8]
Jul 25 14:35:17.665 UTC:     GateID          : 102 (0x66)
Jul 25 14:35:17.665 UTC: ---------------------------------------
Jul 25 14:35:18.181 UTC: Pktcbl(g2g): received GATE_OPEN message, tid 10
Jul 25 14:35:18.181 UTC: --- Pktcbl(g2g): Received G2G message ------
Jul 25 14:35:18.181 UTC: G2G message header
Jul 25 14:35:18.181 UTC:     Type          : GATE_OPEN
Jul 25 14:35:18.181 UTC:     TransactionID : 10
Jul 25 14:35:18.181 UTC:     Msg Length    : 28
Jul 25 14:35:18.181 UTC:     Auth (hex)    : 18 88 CB A2 55 9A 6B 19 3D DB 7B AD 83 FC FC
E9
Jul 25 14:35:18.181 UTC: GATE ID              : Object.[type/len 224/8]
Jul 25 14:35:18.181 UTC:     GateID          : 102 (0x66)
Jul 25 14:35:18.181 UTC: ---------------------------------------
Jul 25 14:35:18.181 UTC: Pktcbl(g2g): Duplicate GATE-OPEN for gateID: 102, not processing
it, sending GATE-OPEN-ACK
Jul 25 14:35:18.181 UTC: Pktcbl(g2g): sending g2g GATE_OPEN_ACK message to
192.168.80.15:1812 from 192.168.80.1:50048
Jul 25 14:35:18.181 UTC: --- Pktcbl(g2g): Sending G2G message ------
Jul 25 14:35:18.181 UTC: G2G message header
Jul 25 14:35:18.181 UTC:     Type          : GATE_OPEN_ACK
Jul 25 14:35:18.181 UTC:     TransactionID : 10
Jul 25 14:35:18.181 UTC:     Msg Length    : 28
Jul 25 14:35:18.181 UTC:     Auth (hex)    : 1A CA 3E 2D B4 D3 18 B9 4D DC 06 C8 98 71 E8
31
Jul 25 14:35:18.181 UTC: GATE ID              : Object.[type/len 224/8]
Jul 25 14:35:18.181 UTC:     GateID          : 102 (0x66)
Jul 25 14:35:18.185 UTC: ---------------------------------------
Jul 25 14:35:19.705 UTC: Pktcbl(g2g): received GATE_OPEN message, tid 10
```

```
Jul 25 14:35:19.705 UTC: --- Pktcbl(g2g): Received G2G message ------
Jul 25 14:35:19.705 UTC: G2G message header
Jul 25 14:35:19.705 UTC:     Type          : GATE_OPEN
Jul 25 14:35:19.705 UTC:     TransactionID : 10
Jul 25 14:35:19.705 UTC:     Msg Length    : 28
Jul 25 14:35:19.705 UTC:     Auth (hex)    : 18 88 CB A2 55 9A 6B 19 3D DB 7B AD 83 FC FC
E9
Jul 25 14:35:19.705 UTC: GATE ID             : Object.[type/len 224/8]
Jul 25 14:35:19.709 UTC:     GateID          : 102 (0x66)
Jul 25 14:35:19.709 UTC: ------------------------------------
Jul 25 14:35:19.709 UTC: Pktcbl(g2g): Duplicate GATE-OPEN for gateID: 102, not processing
it, sending GATE-OPEN-ACK
Jul 25 14:35:19.709 UTC: Pktcbl(g2g): sending g2g GATE_OPEN_ACK message to
192.168.80.15:1812 from 192.168.80.1:50048
Jul 25 14:35:19.709 UTC: --- Pktcbl(g2g): Sending G2G message ------
Jul 25 14:35:19.709 UTC: G2G message header
Jul 25 14:35:19.709 UTC:     Type          : GATE_OPEN_ACK
Jul 25 14:35:19.709 UTC:     TransactionID : 10
Jul 25 14:35:19.709 UTC:     Msg Length    : 28
Jul 25 14:35:19.709 UTC:     Auth (hex)    : 1A CA 3E 2D B4 D3 18 B9 4D DC 06 C8 98 71 E8
31
Jul 25 14:35:19.709 UTC: GATE ID             : Object.[type/len 224/8]
Jul 25 14:35:19.709 UTC:     GateID          : 102 (0x66)
Jul 25 14:35:19.709 UTC: ------------------------------------
```

Note The GATE_OPEN and GATE_CLOSE messages show the gate ID of the remote gate, not the local gate.

Gate Database Debugging

The following example shows typical debugging messages for gate database events:

```
Router# debug packetcable subscriber 192.168.78.21
Router# debug packetcable database
Pktcbl gate database changes debugging is on

Jul 25 16:17:29 UTC: Pktcbl(gdb): Allocated gateid 2359392 0x240060 (bucket 0x24, 0x60)
Jul 25 16:17:29 UTC: Pktcbl(gdb): Created new Subscriber IE for 192.168.78.19
Jul 25 16:17:29 UTC: Pktcbl(gdb): Updated subscriber IE [subs addr: 192.168.78.19, gate:
2359392]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Created gate IE, gateid = 2359392, total gates: 1
Jul 25 16:17:29 UTC: Pktcbl(gdb): Started gate [id 2359392] timer t0 [30 sec]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Allocated gateid 2430632 0x2516A8 (bucket 0x25, 0x16A8)
Jul 25 16:17:29 UTC: Pktcbl(gdb): Created new Subscriber IE for 192.168.78.21
Jul 25 16:17:29 UTC: Pktcbl(gdb): Updated subscriber IE [subs addr: 192.168.78.21, gate:
2430632]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Created gate IE, gateid = 2430632, total gates: 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): Started gate [id 2430632] timer t0 [30 sec]
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2430632 direction
downstream to state 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2430632 direction
downstream to state 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2430632 direction
downstream to state 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): Stopped gate [id 2430632] timer [type 1]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Started gate [id 2430632] timer t1 [300 sec]
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2359392 direction
downstream to state 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2359392 direction
downstream to state 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2359392 direction
downstream to state 2
Jul 25 16:17:29 UTC: Pktcbl(gdb): Stopped gate [id 2359392] timer [type 1]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Started gate [id 2359392] timer t1 [300 sec]
```

```
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to set gate state to 3 in upstream direction for
gate handle 622D3650
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2430632 direction upstream
to state 3
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to set gate state to 3 in downstream direction
for gate handle 622D3650
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2430632 direction
downstream to state 3
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to set gate state to 3 in upstream direction for
gate handle 622C3EF8
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2359392 direction upstream
to state 3
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to set gate state to 3 in downstream direction
for gate handle 622C3EF8
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2359392 direction
downstream to state 3
Jul 25 16:17:29 UTC: Pktcbl(gdb): request to transition gate ID 2430632 direction upstream
to state 4
Jul 25 16:17:29 UTC: Pktcbl(gdb): Started gate [id 2430632] timert2 [2000000 ms]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Started gate [id 2430632] timer t5 [500 msec]
Jul 25 16:17:29 UTC: Pktcbl(gdb): Cancelled gate [id 2430632] timer t1
Jul 25 16:17:29 UTC: Pktcbl(gdb): Cancelled gate [id 2430632] timer t2
Jul 25 16:17:29 UTC: Pktcbl(gdb): Cancelled gate [id 2430632] timer t5
```

DOCSIS-to-RSVP Mapping Debugging

The following example shows typical debugging messages for the mapping between DOCSIS and RSVP messages:

```
Router# debug packetcable subscriber 192.168.78.21
Router# debug packetcable docsis-mapping
Pktcbl rsvp-to-docsis msgs debugging is on

Jul 25 16:17:33 UTC: Pktcbl(d2r): rsvp->docsis: Received a DOCSIS_RESERVE request
Jul 25 16:17:33 UTC: Pktcbl(d2r): rsvp->docsis: Received a DOCSIS_RESERVE request
Jul 25 16:17:33 UTC: Pktcbl(d2r): docsis->rsvp, DSA received, gateid: 2430632
Jul 25 16:17:33 UTC: Pktcbl(d2r): Notifying RSVP layer of CM initiated req
[api_msg.opcode: 4]
Jul 25 16:17:33 UTC: Pktcbl(d2r): docsis->rsvp, DSA received, gateid: 2359392
Jul 25 16:17:33 UTC: Pktcbl(d2r): Notifying RSVP layer of CM initiated req
[api_msg.opcode: 4]
Jul 25 16:17:33 UTC: Pktcbl(d2r): rsvp->docsis: Received a DOCSIS_COMMIT request
Jul 25 16:17:33 UTC: Pktcbl(d2r): rsvp->docsis: Received a DOCSIS_COMMIT request
Jul 25 16:17:33 UTC: Pktcbl(d2r): request to transition gate ID 2359392 direction upstream
to state 4
```

Event Message Debugging

The following example shows typical debugging messages for event messages:

```
Router# debug packetcable subscriber 192.168.78.21
Router# debug packetcable events
Pktcbl event msgs debugging is on

Jul 25 16:14:41 UTC: Pktcbl(em): RKS send call-answer for gate_id=2430632
Jul 25 16:14:41 UTC: Pktcbl(em): pktcbl_send_event() event_type=CALL_ANSWER port=0
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=1
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_HEADER)
Jul 25 16:14:41 UTC: Pktcbl(em): pktcbl_add_header(): seq_no=136
event_time=19170628064103.06 event_msg_type=15
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=5
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_PARTY_NUMBER)
Jul 25 16:14:41 UTC: Pktcbl(em): length=28, vendor_attr_length=22
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=16
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_CHARGE_NUMBER)
```

```
Jul 25 16:14:41 UTC: Pktcbl(em): length=28, vendor_attr_length=22
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=22
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_LOC_ROUTING_NUMBER)
Jul 25 16:14:41 UTC: Pktcbl(em): length=28, vendor_attr_length=22
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=25
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_ROUTING_NUMBER)
Jul 25 16:14:41 UTC: Pktcbl(em): length=28, vendor_attr_length=22
Jul 25 16:14:41 UTC: Pktcbl(em): RKS send qos-start for gate_id=2430632
Jul 25 16:14:41 UTC: Pktcbl(em): pktcbl_send_event() event_type=QOS_START port=0
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=1
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_HEADER)
Jul 25 16:14:41 UTC: Pktcbl(em): pktcbl_add_header(): seq_no=137
event_time=19170628064103.06 event_msg_type=7
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes, attr_id=26
Jul 25 16:14:41 UTC: Pktcbl(em): add_attributes(ATTR_MTA_UDP_PORTNUM)
Jul 25 16:14:41 UTC: Pktcbl(em): length=12, vendor_attr_length=6
```

Note Use the **debug packetcable events process** command to display timing-related information for PacketCable events.

Related Commands

Command	Description
debug cops	Displays debugging messages related to the COPS protocol.
debug packetcable all	Displays general debugging messages for all PacketCable events.
debug packetcable cops	Enables debugging of Common Open Policy Service (COPS) messages and errors that are related to PacketCable operations on the Cisco CMTS router.
debug packetcable hccp	Enables debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations.
debug packetcable ipc	Enables interprocess communications (IPC) messages related to PacketCable operations on the Cisco uBR10012 router.
debug packetcable subscriber	Enables PacketCable debugging for a specific subscriber IP address.

debug packetcable hccp

To display debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations, use the **debug packetcable hccp** command in privileged EXEC mode. To turn off the debugging, use the **no** form of this command.

debug packetcable hccp

no debug packetcable hccp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)BC2	This command was introduced for the Cisco uBR7246VXR and Cisco uBR10012 universal broadband routers.

Usage Guidelines

Use this command to enable debugging for events that are related to HCCP redundancy operations that are related to PacketCable operations on the Cisco CMTS. These debugging messages show errors that occurred when the Cisco CMTS was synchronizing the PacketCable databases and state information between the Working and Protect interfaces.

Examples

The following example shows a typical debugging message for the **debug packetcable hccp** command.

```
Router# debug packetcable hccp

Pktcbl HCCP msgs debugging is on

Pktcbl(gdb): gdb pointer is null for if_index = 13, gid = 3
```

Tip The above debug message indicates that the Cisco CMTS was synchronizing the PacketCable gate database on the Protect interface at the same time a PacketCable gate was in the process of being created. Because the gate had not yet been fully initializated, the synchronization of the gate database had incomplete information about the gate. This is typically a temporary condition that is resolved at the next synchronization of the interfaces.

Related Commands

Command	Description
debug cops	Displays debugging messages related to the COPS protocol.
debug hccp sync	Displays debug messages for HCCP synchronization messages.
debug packetcable all	Displays general debugging messages for all PacketCable events.

Command	Description
debug packetcable cops	Enables debugging of Common Open Policy Service (COPS) messages and errors that are related to PacketCable operations on the Cisco CMTS router.
debug packetcable gate	Displays general debugging messages related to specific PacketCable gate events and messages.
debug packetcable ipc	Enables interprocess communications (IPC) messages related to PacketCable operations on the Cisco uBR10012 router.

debug packetcable ipc

To enable interprocess communications (IPC) messages related to PacketCable operations on the Cisco uBR10012 router, use the **debug packetcable ipc** command in privileged EXEC mode. To turn off Packetcable IPC debugging, use the **no** form of this command.

debug packetcable ipc

no debug packetcable ipc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)BC1	This command was introduced for the Cisco uBR10012 universal broadband router.

Usage Guidelines

This command enables debugging for the IPC messages that are sent between the PRE1 module and cable interface line cards on the Cisco uBR10012 router during PacketCable operations. This command is not supported on the Cisco uBR7200 series router because this router does not use IPC messaging for PacketCable operations.

Tip On the Cisco uBR10012 router, the **debug packetcable ipc** command displays some of the debugging messages that are displayed on other Cisco CMTS platforms by the **control** option for the **debug packetcable gate** command.

Note The **debug packetcable ipc** command does not display any output until you have also enabled PacketCable debugging for one or more IP addresses using the**debug packetcable subscriber** command.

Examples

The following example shows typical debugging output for the **debug packetcable ipc** command:

```
Router# debug packetcable subscriber 10.10.10.131
Router# debug packetcable ipc
Debug pktcbl IPC messages debugging is on
Router#

Jul 26 08:51:34 UTC: Pktcbl(ipc): Delete gate [5123] notification sent from Cable 6/1/1 to
RP
Jul 26 08:51:34 UTC: Pktcbl(ipc): Delete gate [5123] IPC received on LC
Jul 26 08:51:34 UTC: Pktcbl(ipc): Gate-set [5123] notification sent from Cable 6/1/1 to RP
Jul 26 08:51:34 UTC: Pktcbl(ipc): Create gate [5126] IPC sent to Cable 6/1/1
Jul 26 08:51:34 UTC: Pktcbl(ipc): Create gate [5126] IPC received on LC
```

```
Jul 26 08:51:34 UTC: Pktcbl(ipc): Gate-set response [5126] IPC message received on RP
Jul 26 08:54:17 UTC: Pktcbl(ipc): Create gate [7173] IPC sent to Cable6/1/1
Jul 26 08:54:17 UTC: Pktcbl(ipc): Gate-set [7173] ipc sent to Cable6/1/1
Jul 26 08:54:17 UTC: Pktcbl(ipc): Gate-set response [7173] IPC message received on RP
```

Related Commands

Command	Description
debug cops	Displays debugging messages related to the COPS protocol.
debug packetcable all	Displays general debugging messages for all PacketCable events.
debug packetcable cops	Enables debugging of Common Open Policy Service (COPS) messages and errors that are related to PacketCable operations on the Cisco CMTS router.
debug packetcable gate	Displays general debugging messages related to specific PacketCable gate events and messages.
debug packetcable hccp	Enables debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations.
debug packetcable subscriber	Enables PacketCable debugging for a specific subscriber IP address.

debug packetcable subscriber

To enable PacketCable debugging for a specific subscriber IP address or subnet, use the **debug packetcable subscriber** command in privileged EXEC mode. To turn off Packetcable debugging for a particular IP address or subnet, use the **no** form of this command.

debug packetcable subscriber *ip-address* [*mask*] [**verbose**]

no debug packetcable subscriber *ip-address* [*mask*] [**verbose**]

Syntax Description

ip-address	IP address for which PacketCable debugging should be enabled.
mask	Subnet mask to use for this IP address. The router uses this mask to determine the scope of the IP addresses to be debugged. The default is 255.255.255.255 to indicate the IP address is a specific, single IP address.
verbose	Displays detailed information as part of the debug messages, such as the contents of a PacketCable headers and messages. **Note** The **verbose** option replaces the **detail** option that appeared in previous versions of the other **debug packetcable** command.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)BC1	This command was introduced for the Cisco uBR7200 series universal broadband router.

Usage Guidelines

This command enables debugging for one or more specific PacketCable subscribers. This command by itself does not generate any debugging output, but enables debugging output for the other **debug packetcable** commands.

In Cisco IOS Release 12.2(15)BC1 and later releases, you must use the debug packet subscriber command to enable debugging for at least one IP address before the following **debug packetcable** commands will generate output:

- **debug packetcable gate commit**
- **debug packetcable gate control**
- **debug packetcable gate database**
- **debug packetcable gate docsis-mapping**
- **debug packetcable gate events**
- **debug packetcable ipc**

Note You do not need to use the debug packetcable subscribe command before using the **debug packetcable cops** and **debug packetcable gate events process** commands because these display information that is not subscriber-specific.

Examples

The following example shows how to enable PacketCable gate control debugging for one specific PacketCable subscriber at the IP address of 10.10.10.131:

```
Router# debug packetcable subscriber 10.10.10.131
Router# debug packetcable gate control
Router#
```

The following example shows how to enable PacketCable events debugging for all PacketCable subscribers within the 6.6.1.0 subnet:

```
Router# debug packetcable subscriber 6.6.1.0 255.255.255.0
Router# debug packetcable gate events
Router#
```

Related Commands

Command	Description
debug cops	Displays debugging messages related to the COPS protocol.
debug packetcable all	Displays general debugging messages for all PacketCable events.
debug packetcable cops	Enables debugging of Common Open Policy Service (COPS) messages and errors that are related to PacketCable operations on the Cisco CMTS router.
debug packetcable gate	Displays general debugging messages related to specific PacketCable gate events and messages.
debug packetcable hccp	Enables debugging messages for events that are related to Hot Standby Connection-to-Connection Protocol (HCCP) N+1 redundancy synchronization for PacketCable operations.
debug packetcable ipc	Enables interprocess communications (IPC) messages related to PacketCable operations on the Cisco uBR10012 router.

debug pxf

To enable debugging of the Parallel eXpress Forwarding (PXF) subsystems on the active Performance Routing Engine (PRE1) module on the Cisco uBR10012 router, use the **debug pxf** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

debug pxf {**dma** | **microcode** | **stats** | **subblocks**}

no debug pxf {**dma** | **microcode** | **stats** | **subblocks**}

Syntax Description

dma	Displays debugging information for direct memory access (DMA) buffers, error counters, and registers on the PXF processor.
microcode	Disables the automatic reloading of the microcode on the PXF processors upon a crash or fault condition. (If a fault or crash occurs, you must manually load the microcode on the PXF processors using the **microcode reload** command.)
stats	Displays debugging information for the statistics collector events on the PXF processor.
subblocks	Displays debugging information for changes in the subblocks onboard the PXF processor.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)BC1	This command was introduced for the Cisco uBR10012 universal broadband router.

Usage Guidelines

The **debug pxf** command enables debugging of the different subsystems that are active on the PXF processors that are on the PRE1 modules in the Cisco uBR10012 router.

Also, by default the PRE1 modules automatically reload the microcode on the PXF processors in case of a crash or fault. The **debug pxf microcode** command disables this automatic reloading of the microcode, to allow further debugging. When you have finished debugging the PXF processor, you then must manually reload the microcode on the PXF processor using the **microcode reload** command.

Examples

debug pxf dma

The following example shows how to enable debugging of the DMA engine on the PXF processor, as well as typically debugging messages that appear for the packets sent and received by the **ping** command:

```
Router# debug pxf dma

PXF DMA ASIC debugging is on

Router# ping 11.1.1.1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 11.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/8/8 ms

Router#

01:08:28: Entering c10k_cobalt_send.
01:08:28:       Packet decode: datagramstart 0x067D14C8 length 116
01:08:28:       Header decode: Chan 0, VCCI 2
01:08:28:       Header decode: flags 0x0000
01:08:28:       c10k_cobalt_send: Checked the idb state.
01:08:28:       c10k_cobalt_send: Checked the FromRP Q count.
01:08:28:       c10k_cobalt_send: Particle count 0.
01:08:28: Entering cobalt_pak_to_fromrp_ring_inline, scattered FALSE
01:08:28:   cobalt_pak_to_fromrp_ring_inline from_rp_tail 13
01:08:28: Entering c10k_cobalt_type0_interrupt, status reg: 0x00000018.
01:08:28:   ToRP Dispatched.
01:08:28: Entering cobalt_to_rp_interrupt.
01:08:28:       cobalt_to_rp_interrupt: to_rp_head 28
01:08:28:       cobalt_to_rp_interrupt: descriptor address 0x0672F900
01:08:28:               descriptor buf pointer 0x0672AC00
01:08:28:               descriptor buf length 114
01:08:28:               descriptor buf flags 0x000C
01:08:28:       cobalt_to_rp_interrupt, PAK DATA:
01:08:28:               0x00720012 0x180100FF 0x00000F00 0x08004500
01:08:28:               0x0064D02E 0x0000FF01 0xE9680100 0x00010100
01:08:28:               0x00010800 0xC68A80FF 0x26420000 0x0000003E
01:08:28:               0x1040ABCD 0xABCDABCD 0xABCDABCD 0xABCDABCD
01:08:28:
01:08:28: Entering cobalt_process_to_rp_packet.
01:08:28:       Packet decode: datagramstart 0x0672AC00
01:08:28:       Header decode: length 114, cause 0x0012
01:08:28:       Header decode: rsrc_num 3, column 0
01:08:28:       Header decode: flags 1
01:08:28:       Cobalt packet passed to RP driver!
01:08:28:       cobalt_to_rp_interrupt: to_rp_head 29
01:08:28:       cobalt_to_rp_interrupt: descriptor address 0x0672F910
01:08:28:               descriptor buf pointer 0x0672A980
01:08:28:               descriptor buf length 114
01:08:28:               descriptor buf flags 0x000C
01:08:28:       cobalt_to_rp_interrupt, PAK DATA:
01:08:28:               0x00720012 0x280100FF 0x00000F00 0x08004500
01:08:28:               0x0064D02F 0x0000FF01 0xD3650B01 0x01010B01
01:08:28:               0x01010800 0x9AEB1885 0x1A1F0000 0x0000003E
01:08:28:               0xB07CABCD 0xABCDABCD 0xABCDABCD 0xABCDABCD
01:08:28:
01:08:28: Entering cobalt_process_to_rp_packet.
01:08:28:       Packet decode: datagramstart 0x0672A980
01:08:28:       Header decode: length 114, cause 0x0012
01:08:28:       Header decode: rsrc_num 5, column 0
01:08:28:       Header decode: flags 1
01:08:28:       Cobalt packet passed to RP driver!
01:08:28: Exiting cobalt_to_rp_interrupt, to_rp_head 30, Packets processed 2
01:08:28:   FromRP Dispatched.
01:08:28: Entering cobalt_from_rp_interrupt.
01:08:28:   Cleaned descriptor 0x0672FB10.
01:08:28:       desc buf ptr 0x00000000.
01:08:28:       desc buf len 0x00000000.
01:08:28:       desc buf flags 0x00000000.
01:08:28: ICMP: echo reply sent, src 11.1.1.1, dst 11.1.1.1
01:08:28: Entering c10k_cobalt_send.
01:08:28:       Packet decode: datagramstart 0x067D1608 length 116
01:08:28:       Header decode: Chan 0, VCCI 2
```

```
01:08:28:       Header decode: flags 0x0000
01:08:28:       c10k_cobalt_send: Checked the idb state.
01:08:28:       c10k_cobalt_send: Checked the FromRP Q count.
01:08:28:       c10k_cobalt_send: Particle count 0.
01:08:28: Entering cobalt_pak_to_fromrp_ring_inline, scattered FALSE
01:08:28:   cobalt_pak_to_fromrp_ring_inline from_rp_tail 14
01:08:28: Entering c10k_cobalt_type0_interrupt, status reg: 0x00000018.
01:08:28:   ToRP Dispatched.
01:08:28: Entering cobalt_to_rp_interrupt.
01:08:28:       cobalt_to_rp_interrupt: to_rp_head 30
01:08:28:       cobalt_to_rp_interrupt: descriptor address 0x0672F920
01:08:28:               descriptor buf pointer 0x0672A700
01:08:28:               descriptor buf length 114
01:08:28:               descriptor buf flags 0x000C
01:08:28:       cobalt_to_rp_interrupt, PAK DATA:
01:08:28:               0x00720012 0x280100FF 0x00000F00 0x08004500
01:08:28:               0x0064D02F 0x0000FF01 0xD3650B01 0x01010B01
01:08:28:               0x01010000 0xA2EB1885 0x1A1F0000 0x0000003E
01:08:28:               0xB07CABCD 0xABCDABCD 0xABCDABCD 0xABCDABCD
01:08:28:
01:08:28: Entering cobalt_process_to_rp_packet.
01:08:28:       Packet decode: datagramstart 0x0672A700
01:08:28:       Header decode: length 114, cause 0x0012
01:08:28:       Header decode: rsrc_num 5, column 0
01:08:28:       Header decode: flags 1
01:08:28:       Cobalt packet passed to RP driver!
01:08:28: Exiting cobalt_to_rp_interrupt, to_rp_head 31, Packets processed 1
01:08:28:   FromRP Dispatched.
01:08:28: Entering cobalt_from_rp_interrupt.
01:08:28:   Cleaned descriptor 0x0672FB20.
01:08:28:       desc buf ptr 0x00000000.
01:08:28:       desc buf len 0x00000000.
01:08:28:       desc buf flags 0x00000000.
01:08:28: ICMP: echo reply rcvd, src 11.1.1.1, dst 11.1.1.1
01:08:28: ICMP: echo reply sent, src 1.0.0.1, dst 1.0.0.1
01:08:28: Entering c10k_cobalt_send.
01:08:28:       Packet decode: datagramstart 0x067D1388 length 116
01:08:28:       Header decode: Chan 0, VCCI 5
01:08:28:       Header decode: flags 0x0000
01:08:28:       c10k_cobalt_send: Checked the idb state.
01:08:28:       c10k_cobalt_send: Checked the FromRP Q count.
01:08:28:       c10k_cobalt_send: Particle count 0.
01:08:28: Entering cobalt_pak_to_fromrp_ring_inline, scattered FALSE
01:08:28:   cobalt_pak_to_fromrp_ring_inline from_rp_tail 15
01:08:28: Entering c10k_cobalt_type0_interrupt, status reg: 0x00000018.
01:08:28:   ToRP Dispatched.
01:08:28: Entering cobalt_to_rp_interrupt.
01:08:28: Exiting cobalt_to_rp_interrupt, to_rp_head 31, Packets processed 0
01:08:28:   FromRP Dispatched.
01:08:28: Entering cobalt_from_rp_interrupt.
01:08:28:   Cleaned descriptor 0x0672FB30.
01:08:28:       desc buf ptr 0x00000000.
01:08:28:       desc buf len 0x00000000.
01:08:28:       desc buf flags 0x00000001.
01:08:28: Entering c10k_cobalt_type0_interrupt, status reg: 0x00000010.
01:08:28:   ToRP Dispatched.
01:08:28: Entering cobalt_to_rp_interrupt.
01:08:28:       cobalt_to_rp_interrupt: to_rp_head 31
01:08:28:       cobalt_to_rp_interrupt: descriptor address 0x0672F930
01:08:28:               descriptor buf pointer 0x0672A480
01:08:28:               descriptor buf length 114
01:08:28:               descriptor buf flags 0x000D
01:08:28:       cobalt_to_rp_interrupt, PAK DATA:
01:08:28:               0x00720012 0x180100FF 0x00000F00 0x08004500
```

```
01:08:28:               0x0064D02E 0x0000FF01 0xE9680100 0x00010100
01:08:28:               0x00010000 0xCE8A80FF 0x26420000 0x0000003E
01:08:28:               0x1040ABCD 0xABCDABCD 0xABCDABCD 0xABCDABCD
01:08:28:
01:08:28: Entering cobalt_process_to_rp_packet.
01:08:28:       Packet decode: datagramstart 0x0672A480
01:08:28:       Header decode: length 114, cause 0x0012
01:08:28:       Header decode: rsrc_num 3, column 0
01:08:28:       Header decode: flags 1
01:08:28:       Cobalt packet passed to RP driver!
01:08:28: Exiting cobalt_to_rp_interrupt, to_rp_head 0, Packets processed 1
01:08:28: ICMP: echo reply rcvd, src 1.0.0.1, dst 1.0.0.1
Router#
```

debug pxf microcode

The following example shows how to disable the automatic reloading of the microcode on the PXF processors if a fault or crash occurs:

```
Router# debug pxf microcode

PXF microcode debugging is on

Router#
```

If a fault or crash then occurs on the PXF processors, the following messages are displayed on the console to indicate that the automatic reload of the microcode has been disabled and that the microcode must be manually reloaded using the **microcode reload** command.

```
Mar  1 16:29:58.796: %SLICE_TOASTER-4-PXF_CRASHINFO: Writing Slice (PXF) debug information
to slot0:pxf_crashinfo_20020301-212958.
Mar  1 16:30:00.252: NOTE: 'debug pxf microcode' is set. Use 'micro reload pxf' to restart
the PXF
```

debug pxf stats

The following example shows how to enable debugging of the statistics collector on the PXF processors, along with some typical debugging messages:

```
Router# debug pxf stats

PXF hardware statistics debugging is on

Router#

00:05:28: read vpn session 1 stat from toaster
00:05:28: read vpn session 2 stat from toaster
00:05:29: Statcoll: read simple stats failed
00:05:29: Statcoll read simple_octet...: NULL params (hwsb=0xFF, tt_info=0x%C0)
```

debug pxf subblocks

The following example shows how to enable debugging of the subblocks on the PXF processors:

```
Router# debug pxf subblocks

PXF Subblocks debugging is on

Router#
```

Related Commands

Command	Description
clear pxf	Clears the DMA and error checking and correcting (ECC) error counters on the PXF processor.
show hardware pxf cable	Displays information about the multicast echo and packet intercept features for one or all cable interfaces.
show hardware pxf cpu	Displays the display different statistics about the operation of the CPU processor during PXF processing.
show hardware pxf microcode	Displays identifying information for the microcode being used on the processor.
show hardware pxf xcm	Displays the current state of ECC for the External Column Memory (XCM) on the PXF processor.

debug pxf atom

To divert upstream and downstream Layer 2 Virtual Private Network (L2VPN) packets from the hardware switching path to the software switching path on the CMTS routers, use the **debug pxf atom** command in privileged EXEC mode. To cancel this operation, use the **no** form of this command.

debug pxf atom {**ac** | **mpls** | **punt** {**upstream cable** *slot*/*subslot*/*port sid-value* | **downstream** *label-number*}}

no debug pxf atom {**ac** | **mpls** | **punt** {**upstream cable** *slot*/*subslot*/*port sid-value* | **downstream** *label-number*}}

Syntax Description

ac	Specifies the attachment circuit.
mpls	Specifies the MPLS forwarding information (MFI).
punt	Diverts packet from the hardware switching path to the software switching path.
upstream	Specifies the upstream packets.
cable *slot*/*subslot*/*port*	Specifies the cable interface for the L2VPN-compliant cable modem, where: • *slot*—Chassis slot number of the cable interface line card. Valid values are from 5 to 8. • *subslot*—Secondary slot number of the cable interface line card. Valid subslots are 0 or 1. • *port*—Port number. Valid values are from 0 to 4 (depending on the cable interface).
sid-value	Service ID (SID) of the cable modem.
downstream *label-number*	Specifies the local Multiprotocol Label Switching (MPLS) label of the pseudowire on the CMTS for downstream packets.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SCC	This command was introduced in Cisco IOS Release 12.2(33)SCC.

Usage Guidelines

This debug command enables the Parallel eXpress Forwarding (PXF) function to divert the upstream L2VPN packets from the hardware switching path to the software switching path, if the upstream packets match with the specified cable interface and SID. This debug command also enables the PXF function to divert the downstream L2VPN packets from the hardware switching path to the software switching path, if the downstream packets match with the specified local MPLS pseudowire label.

The **debug pxf atom** command must be used in conjunction with the **debug cable mac-address** and **debug cable l2-vpn** commands.

Related Commands

Command	Description
debug cable mac-address	Enables debugging output for a specific cable modem.
debug cable l2-vpn	Enables debugging output for the Layer 2 mapping of cable modems to a permanent virtual connection (PVC) or a virtual local area network (VLAN).

debug redundancy

To enable debugging of the Route Processor Redundancy (RPR) feature and its background procedures, use the **debug redundancy** command in Privileged EXEC mode.

debug redundancy {**alarms** | **all** | **configsync** | **fsm** | **keepalive** | **peer-monitor** | **services** | **timesync**}

Syntax Description

alarms	Enables debugging messages for alarms sent because of redundancy procedures.
all	Enables all redundancy debugging messages.
configsync	Enables debugging messages for the synchronization of the configuration files.
fsm	Enables debugging for changes in the redundancy finite state machine (FSM).
keepalive	Enables debugging messages for the keepalive messages sent between Performance Routing Engine (PRE1) modules.
peer-monitor	Enables debugging messages for the standby PRE1 module's monitoring of the active PRE1 module.
rf-fsm	Enables debugging for changes in the redundancy finite state machine (FSM).
services	Enables debugging for the services requested during redundancy processing.
timesync	Enables debugging messages for time synchronization procedures.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)XF1	This command was introduced for the Cisco uBR10012 router.
12.2(11)BC3	The **fsm** option was renamed to **rf-fsm**.

Examples

The following example shows all redundancy debugging messages being enabled:

```
Router# debug redundancy all

Redundancy All debugging is on

Router#
```

The following example shows typical messages that the **debug redundancy alarms** command displays:

```
Router# debug redundancy alarms

Redundancy Alarms debugging is on
Router#
```

```
01:28:48: %REDUNDANCY-5-PEER_MONITOR_EVENT: Primary detected a secondary crash
(raw-event=KEEPALIVE_FAILURE(7))
slave_down: generating Secondary-Down alarm
Asserting alarm : SEC_FAILURE

01:28:48: %REDUNDANCY-5-PEER_MONITOR_EVENT: Primary detected a secondary crash
(raw-event=PEER_REDUNDANCY_STATE_CHANGE(5))
```

The following example shows the typical state changes that the **debug redundancy fsm** command displays when the standby PRE1 module is reset:

```
Router# debug redundancy fsm

Redundancy FSM debugging is on
Router#

01:15:30: %REDUNDANCY-5-PEER_MONITOR_EVENT: Primary detected a secondary crash
(raw-event=KEEPALIVE_FAILURE(7))
Flushing IPC entries in FSM queue

01:15:30: ehsa_fsm: state change, events: major=2 minor=1
REDUNDANCY_PEERSECONDARY_INITED(9) => REDUNDANCY_PEERSECONDARY_NONOPERATIONAL(6)

01:15:31: %REDUNDANCY-5-PEER_MONITOR_EVENT: Primary detected a secondary crash
(raw-event=PEER_REDUNDANCY_STATE_CHANGE(5))

01:15:31: %REDUNDANCY-5-PEER_MONITOR_EVENT: Primary detected a secondary crash
(raw-event=KEEPALIVE_FAILURE(7))
Flushing IPC entries in FSM queue

01:15:31: ehsa_fsm: state change, events: major=2 minor=1
REDUNDANCY_PEERSECONDARY_INITED(9) => REDUNDANCY_PEERSECONDARY_NONOPERATIONAL(6)

01:15:31: %REDUNDANCY-5-PEER_MONITOR_EVENT: Primary detected a secondary crash
(raw-event=PEER_REDUNDANCY_STATE_CHANGE(5))
```

The following example shows the messages that are displayed by the **debug redundancy keepalive** command:

```
Router# debug red keepalive

Redundancy Keepalive debugging is on
Router#

Sent keepalive
Received keepalive
Sent keepalive
Received keepalive
Sent keepalive
Received keepalive
Sent keepalive
```

Related Commands

Command	Description
associate	Associates two line cards for APS redundancy protection.
auto-sync	Configures what system files the primary and standby PRE1 modules automatically synchronize.
main-cpu	Enters main CPU redundancy configuration mode.

Command	Description
redundancy	Enters redundancy configuration mode.
redundancy force-failover main-cpu	Forces a switchover so that the standby PRE1 module becomes the active PRE1 module.

debug usb

To display the packets that are transmitted and received over the USB interface, use the **debug usb** command in privileged EXEC mode. To turn off debugging for the USB interface, use the **no** form of this command.

Cisco uBR925 cable access router, Cisco CVA122 Cable Voice Adapter

debug usb {**rx** | **tx**}

no debug usb {**rx** | **tx**}

Syntax Description

rx	Displays packets received on the USB interface.
tx	Displays packets transmitted on the USB interface.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(5)XU	This command was introduced for the Cisco CVA122 Cable Voice Adapter.
12.2(2)XA	Support was added for the Cisco uBR925 cable access router.

Usage Guidelines

This command displays a hexadecimal dump of each packet transmitted or received on the USB interface and should be used only while debugging CM operation. Displaying debugging messages, especially for each packet transmitted or received, consumes system resources. Turning on too many messages could negatively affect system performance.

Examples

The following shows typical output from the **debug usb** command.

```
Router# debug usb rx
USB Rx debugging is on
Router# debug usb tx
USB Tx debugging is on
 Nov 22 11:01:36.909: 0x040: 72 6C 19 E7 0C 12
 Nov 22 11:12:18.447: USB transmit packet data (size=114):
 Nov 22 11:12:18.447: 0x000: 00 01 64 FF E1 13 00 01 64 FF E1 12 08 00 45 00
 Nov 22 11:12:18.447: 0x010: 00 64 16 32 00 00 FF 01 62 FF 20 01 01 01 20 01
 Nov 22 11:12:18.451: 0x020: 01 65 08 00 40 5D 1E 3D 07 2D 00 00 00 00 04 19
 Nov 22 11:12:18.451: 0x052: AB CD AB CD AB CD AB CD AB CD AB CD
 Nov 22 11:12:18.455: 0x060: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
 Nov 22 11:12:18.455: 0x070: AB CD
 Nov 22 11:12:18.459: USB receive packet data (size=60):
 Nov 22 11:12:18.459: 0x000: FF FF FF FF FF FF 00 01 64 FF E1 13 08 06 00 01
 Nov 22 11:12:18.463: 0x010: 08 00 06 04 00 01 00 01 64 FF E1 13
 Nov 22 11:12:18.463: 0x01C: 20 01 01 65
 Nov 22 11:12:18.463: 0x020: 00 00 00 00 00 00 20 01 01 01 00 00 00 00 00 00
 Nov 22 11:12:18.467: 0x030: 00 00 00 00 00 00 00 00 00 00 00 00
 Nov 22 11:12:18.467: USB transmit packet data (size=60):
```

```
Nov 22 11:12:18.471: 0x000: 00 01 64 FF E1 13 00 01 64 FF E1 12 08 06 00 01
Nov 22 11:12:18.471: 0x010: 08 00 06 04 00 02 00 01 64 FF E1 12
Nov 22 11:12:18.471: 0x01C: 20 01 01 01
Nov 22 11:12:18.471: 0x020: 00 01 64 FF E1 13 20 01 01 65 00 00 00 00 00 00
Nov 22 11:12:18.475: 0x030: 00 00 00 00 00 00 00 00 00 00 00 00
Nov 22 11:12:18.479: USB receive packet data (size=114):
Nov 22 11:12:18.479: 0x000: 00 01 64 FF E1 12 00 01 64 FF E1 13 08 00 45 00
Nov 22 11:12:18.483: 0x010: 00 64 20 25 00 00 80 01 D8 0C 20 01 01 65 20 01
Nov 22 11:12:18.483: 0x020: 01 01 00 00 48 5D 1E 3D 07 2D 00 00 00 00 04 19
Nov 22 11:12:18.487: 0x052: AB CD AB CD AB CD AB CD AB CD AB CD AB CD
Nov 22 11:12:18.487: 0x060: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
Nov 22 11:12:18.487: 0x070: AB CD
Nov 22 11:12:18.491: USB transmit packet data (size=114):
Nov 22 11:12:18.491: 0x000: 00 01 64 FF E1 13 00 01 64 FF E1 12 08 00 45 00
Nov 22 11:12:18.495: 0x010: 00 64 16 33 00 00 FF 01 62 FE 20 01 01 01 20 01
Nov 22 11:12:18.495: 0x020: 01 65 08 00 40 30 1E 3E 07 2D 00 00 00 00 04 19
Nov 22 11:12:18.499: 0x052: AB CD AB CD AB CD AB CD AB CD AB CD AB CD
Nov 22 11:12:18.499: 0x060: AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
Nov 22 11:12:18.499: 0x070: AB CD
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays information about Baseline Privacy Interface (BPI) key management.
debug cable-modem bridge	Displays bridge filter processing information.
debug cable-modem error	Displays debugging messages for the cable interface driver.
debug cable-modem interrupts	Displays information about cable interface interrupts.
debug cable-modem mac log	Displays comprehensive debugging messages for the cable interface MAC layer.