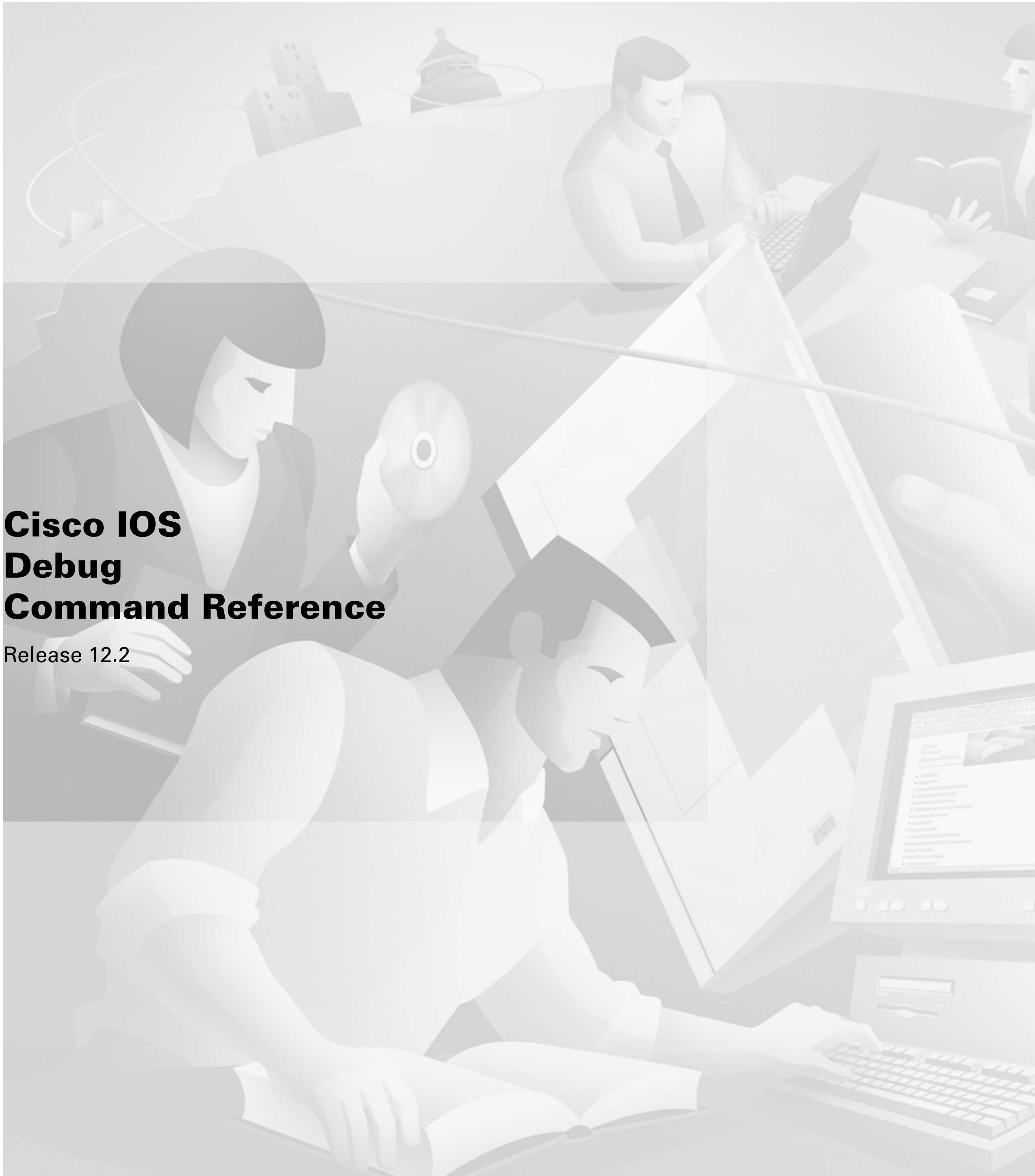
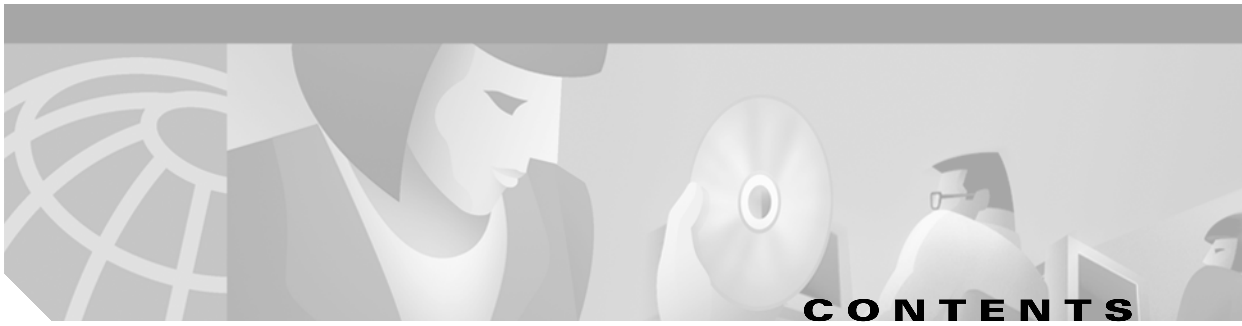


Cisco IOS Debug Command Reference

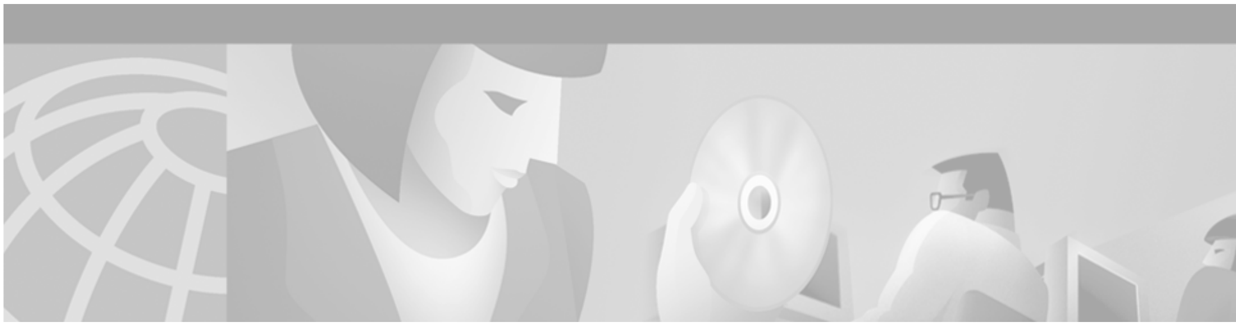
Release 12.2

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Customer Order Number: DOC-7812254=
Text Part Number: 78-12254-02



CONTENTS



About Cisco IOS Software Documentation

This chapter discusses the objectives, audience, organization, and conventions of Cisco IOS software documentation. It also provides sources for obtaining documentation from Cisco Systems.

Documentation Objectives

Cisco IOS software documentation describes the tasks and commands necessary to configure and maintain Cisco networking devices.

Audience

The Cisco IOS software documentation set is intended primarily for users who configure and maintain Cisco networking devices (such as routers and switches) but who may not be familiar with the tasks, the relationship between tasks, or the Cisco IOS software commands necessary to perform particular tasks. The Cisco IOS software documentation set is also intended for those users experienced with Cisco IOS software who need to know about new features, new configuration options, and new software characteristics in the current Cisco IOS software release.

Documentation Organization

The Cisco IOS software documentation set consists of documentation modules and master indexes. In addition to the main documentation set, there are supporting documents and resources.

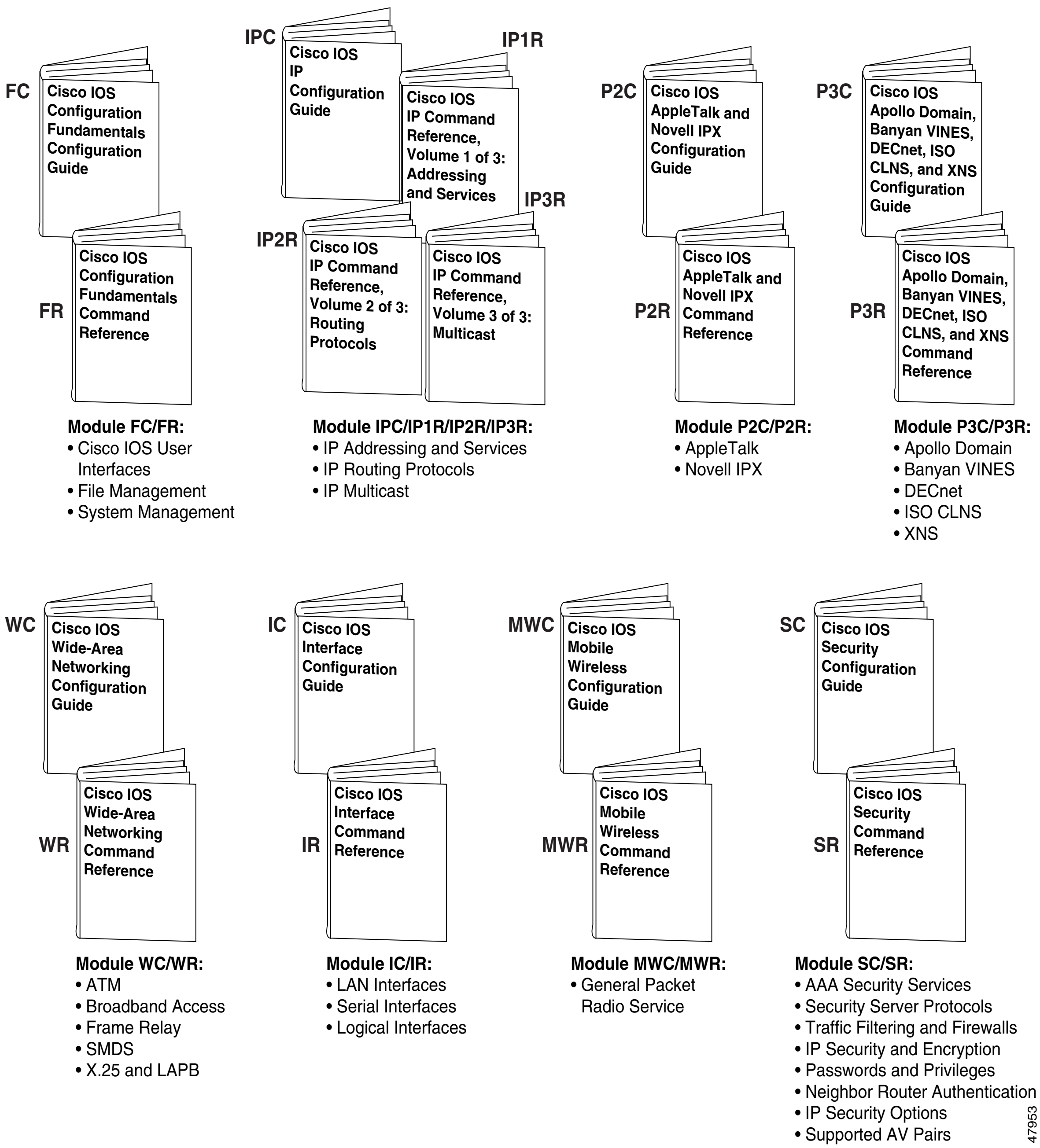
Documentation Modules

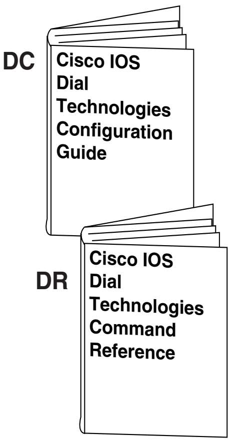
The Cisco IOS documentation modules consist of configuration guides and corresponding command reference publications. Chapters in a configuration guide describe protocols, configuration tasks, and Cisco IOS software functionality and contain comprehensive configuration examples. Chapters in a command reference publication provide complete Cisco IOS command syntax information. Use each configuration guide in conjunction with its corresponding command reference publication.

Figure 1 shows the Cisco IOS software documentation modules.

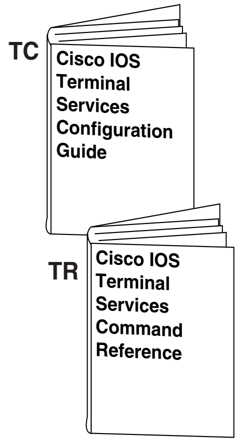
Note The abbreviations (for example, FC and FR) next to the book icons are page designators, which are defined in a key in the index of each document to help you with navigation. The bullets under each module list the major technology areas discussed in the corresponding books.

Figure 1 Cisco IOS Software Documentation Modules

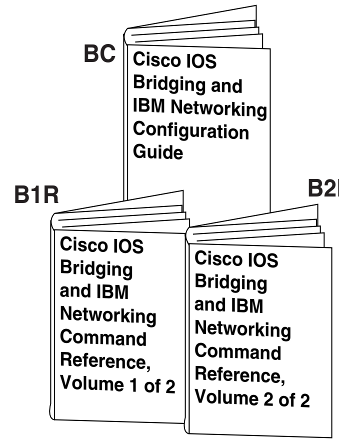




Module DC/DR:
- Preparing for Dial Access
- Modem and Dial Shelf Configuration and Management
- ISDN Configuration
- Signalling Configuration
- Dial-on-Demand Routing Configuration
- Dial-Backup Configuration
- Dial-Related Addressing Services
- Virtual Templates, Profiles, and Networks
- PPP Configuration
- Callback and Bandwidth Allocation Configuration
- Dial Access Specialized Features
- Dial Access Scenarios

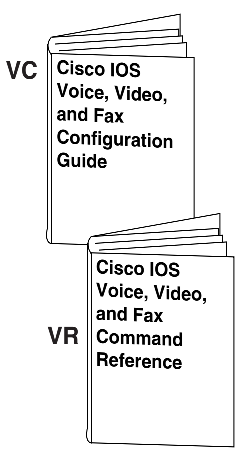


Module TC/TR:
- ARA
- LAT
- NASI
- Telnet
- TN3270
- XRemote
- X.28 PAD
- Protocol Translation

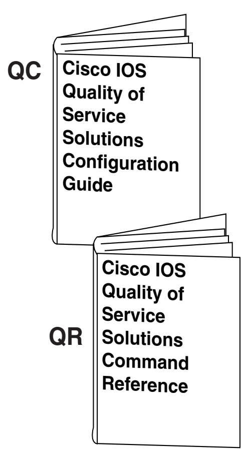


Module BC/B1R:
- Transparent Bridging
- SRB
- Token Ring Inter-Switch Link
- Token Ring Route Switch Module
- RSRB
- DLSw+
- Serial Tunnel and Block Serial Tunnel
- LLC2 and SDLC
- IBM Network Media Translation
- SNA Frame Relay Access
- NCIA Client/Server
- Airline Product Set

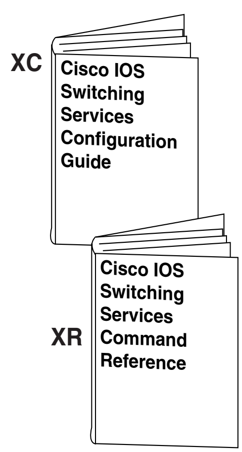
Module BC/B2R:
- DSPU and SNA Service Point
- SNA Switching Services
- Cisco Transaction Connection
- Cisco Mainframe Channel Connection
- CLAW and TCP/IP Offload
- CSNA, CMPC, and CMPC+
- TN3270 Server



Module VC/VR:
- Voice over IP
- Call Control Signalling
- Voice over Frame Relay
- Voice over ATM
- Telephony Applications
- Trunk Management
- Fax, Video, and Modem Support



Module QC/QR:
- Packet Classification
- Congestion Management
- Congestion Avoidance
- Policing and Shaping
- Signalling
- Link Efficiency Mechanisms



Module XC/XR:
- Cisco IOS Switching Paths
- NetFlow Switching
- Multiprotocol Label Switching
- Multilayer Switching
- Multicast Distributed Switching
- Virtual LANs
- LAN Emulation

47954

Master Indexes

Two master indexes provide indexing information for the Cisco IOS software documentation set: an index for the configuration guides and an index for the command references. Individual books also contain a book-specific index.

The master indexes provide a quick way for you to find a command when you know the command name but not which module contains the command. When you use the online master indexes, you can click the page number for an index entry and go to that page in the online document.

Supporting Documents and Resources

The following documents and resources support the Cisco IOS software documentation set:

- *Cisco IOS Command Summary* (two volumes)—This publication explains the function and syntax of the Cisco IOS software commands. For more information about defaults and usage guidelines, refer to the Cisco IOS command reference publications.
- *Cisco IOS System Error Messages*—This publication lists and describes Cisco IOS system error messages. Not all system error messages indicate problems with your system. Some are purely informational, and others may help diagnose problems with communications lines, internal hardware, or the system software.
- *Cisco IOS Debug Command Reference*—This publication contains an alphabetical listing of the **debug** commands and their descriptions. Documentation for each command includes a brief description of its use, command syntax, usage guidelines, and sample output.
- *Dictionary of Internetworking Terms and Acronyms*—This Cisco publication compiles and defines the terms and acronyms used in the internetworking industry.
- New feature documentation—The Cisco IOS software documentation set documents the mainline release of Cisco IOS software (for example, Cisco IOS Release 12.2). New software features are introduced in early deployment releases (for example, the Cisco IOS "T" release train for 12.2, 12.2(x)T). Documentation for these new features can be found in standalone documents called "feature modules." Feature module documentation describes new Cisco IOS software and hardware networking functionality and is available on Cisco.com and the Documentation CD-ROM.
- Release notes—This documentation describes system requirements, provides information about new and changed features, and includes other useful information about specific software releases. See the section "Using Software Release Notes" in the chapter "Using Cisco IOS Software" for more information.
- Caveats documentation—This documentation provides information about Cisco IOS software defects in specific software releases.
- RFCs—RFCs are standards documents maintained by the Internet Engineering Task Force (IETF). Cisco IOS software documentation references supported RFCs when applicable. The full text of referenced RFCs may be obtained on the World Wide Web at http://www.rfc-editor.org/.
- MIBs—MIBs are used for network monitoring. For lists of supported MIBs by platform and release, and to download MIB files, see the Cisco MIB website on Cisco.com at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

Document Conventions

Within Cisco IOS software documentation, the term *router* is generally used to refer to a variety of Cisco products (for example, routers, access servers, and switches). Routers, access servers, and other networking devices that support Cisco IOS software are shown interchangeably within examples. These products are used only for illustrative purposes; that is, an example that shows one product does not necessarily indicate that other products are not supported.

The Cisco IOS documentation set uses the following conventions:

Convention	Description
^ or Ctrl	The ^ and Ctrl symbols represent the Control key. For example, the key combination ^D or Ctrl-D means hold down the Control key while you press the D key. Keys are indicated in capital letters but are not case sensitive.
string	A string is a nonquoted set of characters shown in italics. For example, when setting an SNMP community string to public, do not use quotation marks around the string or the string will include the quotation marks.

Command syntax descriptions use the following conventions:

<table>
<tr><th>Convention</th><th>Description</th></tr>
<tr><td>boldface</td><td>Boldface text indicates commands and keywords that you enter literally as shown.</td></tr>
<tr><td>italics</td><td>Italic text indicates arguments for which you supply values.</td></tr>
<tr><td>[x]</td><td>Square brackets enclose an optional element (keyword or argument).</td></tr>
<tr><td>|</td><td>A vertical line indicates a choice within an optional or required set of keywords or arguments.</td></tr>
<tr><td>[x | y]</td><td>Square brackets enclosing keywords or arguments separated by a vertical line indicate an optional choice.</td></tr>
<tr><td>{x | y}</td><td>Braces enclosing keywords or arguments separated by a vertical line indicate a required choice.</td></tr>
</table>

Nested sets of square brackets or braces indicate optional or required choices within optional or required elements. For example:

<table>
<tr><th>Convention</th><th>Description</th></tr>
<tr><td>[x {y | z}]</td><td>Braces and a vertical line within square brackets indicate a required choice within an optional element.</td></tr>
</table>

Examples use the following conventions:

Convention	Description
`screen`	Examples of information displayed on the screen are set in Courier font.
`boldface screen`	Examples of text that you must enter are set in Courier bold font.
< >	Angle brackets enclose text that is not printed to the screen, such as passwords.

Convention	Description
!	An exclamation point at the beginning of a line indicates a comment line. (Exclamation points are also displayed by the Cisco IOS software for certain processes.)
[ ]	Square brackets enclose default responses to system prompts.

The following conventions are used to attract the attention of the reader:

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in this manual.

Timesaver Means the *described action saves time*. You can save time by performing the action described in the paragraph.

Obtaining Documentation

The following sections provide sources for obtaining documentation from Cisco Systems.

World Wide Web

The most current Cisco documentation is available on the World Wide Web at the following website:

http://www.cisco.com

Translated documentation is available at the following website:

http://www.cisco.com/public/countries_languages.html

Documentation CD-ROM

Cisco documentation and additional literature are available in a CD-ROM package, which ships with your product. The Documentation CD-ROM is updated monthly and may be more current than printed documentation. The CD-ROM package is available as a single unit or through an annual subscription.

Ordering Documentation

Cisco documentation can be ordered in the following ways:

- Registered Cisco Direct Customers can order Cisco product documentation from the Networking Products MarketPlace:

http://www.cisco.com/cgi-bin/order/order_root.pl

- Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:

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

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco corporate headquarters (California, USA) at 408 526-7208 or, in North America, by calling 800 553-NETS(6387).

Documentation Feedback

If you are reading Cisco product documentation on the World Wide Web, you can submit technical comments electronically. Click **Feedback** in the toolbar and select **Documentation**. After you complete the form, click **Submit** to send it to Cisco.

You can e-mail your comments to bug-doc@cisco.com.

To submit your comments by mail, use the response card behind the front cover of your document, or write to the following address:

Cisco Systems, Inc.
Document Resource Connection
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain documentation, troubleshooting tips, and sample configurations from online tools. For Cisco.com registered users, additional troubleshooting tools are available from the TAC website.

Cisco.com

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information and resources at anytime, from anywhere in the world. This highly integrated Internet application is a powerful, easy-to-use tool for doing business with Cisco.

Cisco.com provides a broad range of features and services to help customers and partners streamline business processes and improve productivity. Through Cisco.com, you can find information about Cisco and our networking solutions, services, and programs. In addition, you can resolve technical issues with online technical support, download and test software packages, and order Cisco learning materials and merchandise. Valuable online skill assessment, training, and certification programs are also available.

Customers and partners can self-register on Cisco.com to obtain additional personalized information and services. Registered users can order products, check on the status of an order, access technical support, and view benefits specific to their relationships with Cisco.

To access Cisco.com, go to the following website:

http://www.cisco.com

Technical Assistance Center

The Cisco TAC website is available to all customers who need technical assistance with a Cisco product or technology that is under warranty or covered by a maintenance contract.

Contacting TAC by Using the Cisco TAC Website

If you have a priority level 3 (P3) or priority level 4 (P4) problem, contact TAC by going to the TAC website:

http://www.cisco.com/tac

P3 and P4 level problems are defined as follows:

- P3—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.
- P4—You need information or assistance on Cisco product capabilities, product installation, or basic product configuration.

In each of the above cases, use the Cisco TAC website to quickly find answers to your questions.

To register for Cisco.com, go to the following website:

http://www.cisco.com/register/

If you cannot resolve your technical issue by using the TAC online resources, Cisco.com registered users can open a case online by using the TAC Case Open tool at the following website:

http://www.cisco.com/tac/caseopen

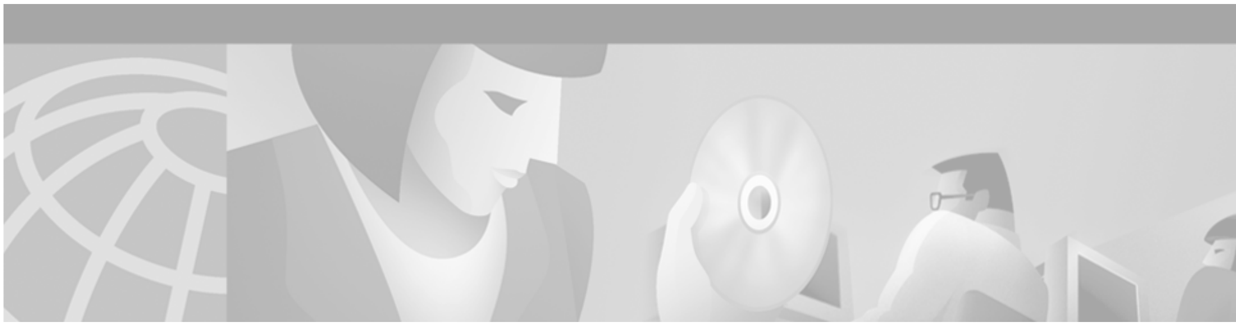
Contacting TAC by Telephone

If you have a priority level 1 (P1) or priority level 2 (P2) problem, contact TAC by telephone and immediately open a case. To obtain a directory of toll-free numbers for your country, go to the following website:

http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml

P1 and P2 level problems are defined as follows:

- P1—Your production network is down, causing a critical impact to business operations if service is not restored quickly. No workaround is available.
- P2—Your production network is severely degraded, affecting significant aspects of your business operations. No workaround is available.



Using Cisco IOS Software

This chapter provides helpful tips for understanding and configuring Cisco IOS software using the command-line interface (CLI). It contains the following sections:

- Understanding Command Modes
- Getting Help
- Using the no and default Forms of Commands
- Saving Configuration Changes
- Filtering Output from the show and more Commands
- Identifying Supported Platforms

For an overview of Cisco IOS software configuration, refer to the *Cisco IOS Configuration Fundamentals Configuration Guide*.

For information on the conventions used in the Cisco IOS software documentation set, see the chapter "About Cisco IOS Software Documentation" located at the beginning of this book.

Understanding Command Modes

You use the CLI to access Cisco IOS software. Because the CLI is divided into many different modes, the commands available to you at any given time depend on the mode you are currently in. Entering a question mark (**?**) at the CLI prompt allows you to obtain a list of commands available for each command mode.

When you log in to the CLI, you are in user EXEC mode. User EXEC mode contains only a limited subset of commands. To have access to all commands, you must enter privileged EXEC mode, normally by using a password. From privileged EXEC mode you can issue any EXEC command—user or privileged mode—or you can enter global configuration mode. Most EXEC commands are one-time commands. For example, **show** commands show important status information, and **clear** commands clear counters or interfaces. The EXEC commands are not saved when the software reboots.

Configuration modes allow you to make changes to the running configuration. If you later save the running configuration to the startup configuration, these changed commands are stored when the software is rebooted. To enter specific configuration modes, you must start at global configuration mode. From global configuration mode, you can enter interface configuration mode and a variety of other modes, such as protocol-specific modes.

ROM monitor mode is a separate mode used when the Cisco IOS software cannot load properly. If a valid software image is not found when the software boots or if the configuration file is corrupted at startup, the software might enter ROM monitor mode.

Table 1 describes how to access and exit various common command modes of the Cisco IOS software. It also shows examples of the prompts displayed for each mode.

Table 1 Accessing and Exiting Command Modes

Command Mode	Access Method	Prompt	Exit Method
User EXEC	Log in.	`Router>`	Use the **logout** command.
Privileged EXEC	From user EXEC mode, use the **enable** EXEC command.	`Router#`	To return to user EXEC mode, use the **disable** command.
Global configuration	From privileged EXEC mode, use the **configure terminal** privileged EXEC command.	`Router(config)#`	To return to privileged EXEC mode from global configuration mode, use the **exit** or **end** command, or press **Ctrl-Z**.
Interface configuration	From global configuration mode, specify an interface using an **interface** command.	`Router(config-if)#`	To return to global configuration mode, use the **exit** command. To return to privileged EXEC mode, use the **end** command, or press **Ctrl-Z**.
ROM monitor	From privileged EXEC mode, use the **reload** EXEC command. Press the **Break** key during the first 60 seconds while the system is booting.	`>`	To exit ROM monitor mode, use the **continue** command.

For more information on command modes, refer to the "Using the Command-Line Interface" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

Getting Help

Entering a question mark (**?**) at the CLI prompt displays a list of commands available for each command mode. You can also get a list of keywords and arguments associated with any command by using the context-sensitive help feature.

To get help specific to a command mode, a command, a keyword, or an argument, use one of the following commands:

Command	Purpose
help	Provides a brief description of the help system in any command mode.
*abbreviated-command-entry***?**	Provides a list of commands that begin with a particular character string. (No space between command and question mark.)
abbreviated-command-entry<**Tab**>	Completes a partial command name.
?	Lists all commands available for a particular command mode.
command **?**	Lists the keywords or arguments that you must enter next on the command line. (Space between command and question mark.)

Example: How to Find Command Options

This section provides an example of how to display syntax for a command. The syntax can consist of optional or required keywords and arguments. To display keywords and arguments for a command, enter a question mark (**?**) at the configuration prompt or after entering part of a command followed by a space. The Cisco IOS software displays a list and brief description of available keywords and arguments. For example, if you were in global configuration mode and wanted to see all the keywords or arguments for the **arap** command, you would type **arap ?**.

The <cr> symbol in command help output stands for “carriage return.” On older keyboards, the carriage return key is the Return key. On most modern keyboards, the carriage return key is the Enter key. The <cr> symbol at the end of command help output indicates that you have the option to press **Enter** to complete the command and that the arguments and keywords in the list preceding the <cr> symbol are optional. The <cr> symbol by itself indicates that no more arguments or keywords are available and that you must press **Enter** to complete the command.

Table 2 shows examples of how you can use the question mark (**?**) to assist you in entering commands. The table steps you through configuring an IP address on a serial interface on a Cisco 7206 router that is running Cisco IOS Release 12.0(3).

Table 2 How to Find Command Options

Command	Comment
Router> **enable** Password: *<password>* Router#	Enter the **enable** command and password to access privileged EXEC commands. You are in privileged EXEC mode when the prompt changes to Router#.
Router# **configure terminal** Enter configuration commands, one per line. End with CNTL/Z. Router(config)#	Enter the **configure terminal** privileged EXEC command to enter global configuration mode. You are in global configuration mode when the prompt changes to Router(config)#.
Router(config)# **interface serial ?** <0-6> Serial interface number Router(config)# **interface serial 4 ?** / Router(config)# **interface serial 4/ ?** <0-3> Serial interface number Router(config)# **interface serial 4/0** Router(config-if)#	Enter interface configuration mode by specifying the serial interface that you want to configure using the **interface serial** global configuration command. Enter **?** to display what you must enter next on the command line. In this example, you must enter the serial interface slot number and port number, separated by a forward slash. You are in interface configuration mode when the prompt changes to Router(config-if)#.

Table 2 How to Find Command Options (continued)

Command	Comment

`Router(config-if)# ?`
`Interface configuration commands:`
`  .`
`  .`
`  .`
`  ip                  Interface Internet Protocol config commands`
`  keepalive           Enable keepalive`
`  lan-name            LAN Name command`
`  llc2                LLC2 Interface Subcommands`
`  load-interval       Specify interval for load calculation for an`
`                      interface`
`  locaddr-priority    Assign a priority group`
`  logging             Configure logging for interface`
`  loopback            Configure internal loopback on an interface`
`  mac-address         Manually set interface MAC address`
`  mls                 mls router sub/interface commands`
`  mpoa                MPOA interface configuration commands`
`  mtu                 Set the interface Maximum Transmission Unit (MTU)`
`  netbios             Use a defined NETBIOS access list or enable`
`                      name-caching`
`  no                  Negate a command or set its defaults`
`  nrzi-encoding       Enable use of NRZI encoding`
`  ntp                 Configure NTP`
`  .`
`  .`
`  .`
`Router(config-if)#`
	Enter **?** to display a list of all the interface configuration commands available for the serial interface. This example shows only some of the available interface configuration commands.

`Router(config-if)#` **`ip ?`**
`Interface IP configuration subcommands:`
`  access-group        Specify access control for packets`
`  accounting          Enable IP accounting on this interface`
`  address             Set the IP address of an interface`
`  authentication      authentication subcommands`
`  bandwidth-percent   Set EIGRP bandwidth limit`
`  broadcast-address   Set the broadcast address of an interface`
`  cgmp                Enable/disable CGMP`
`  directed-broadcast  Enable forwarding of directed broadcasts`
`  dvmrp               DVMRP interface commands`
`  hello-interval      Configures IP-EIGRP hello interval`
`  helper-address      Specify a destination address for UDP broadcasts`
`  hold-time           Configures IP-EIGRP hold time`
`  .`
`  .`
`  .`
`Router(config-if)#` **`ip`**
	Enter the command that you want to configure for the interface. This example uses the **ip** command.
	
	Enter **?** to display what you must enter next on the command line. This example shows only some of the available interface IP configuration commands.

Table 2 How to Find Command Options (continued)

Command	Comment

`Router(config-if)# ip address ?`
`  A.B.C.D            IP address`
`  negotiated         IP Address negotiated over PPP`
`Router(config-if)# ip address`
	Enter the command that you want to configure for the interface. This example uses the **ip address** command.
	
	Enter **?** to display what you must enter next on the command line. In this example, you must enter an IP address or the **negotiated** keyword.
	
	A carriage return (<cr>) is not displayed; therefore, you must enter additional keywords or arguments to complete the command.

`Router(config-if)# ip address 172.16.0.1 ?`
`  A.B.C.D            IP subnet mask`
`Router(config-if)# ip address 172.16.0.1`
	Enter the keyword or argument you want to use. This example uses the 172.16.0.1 IP address.
	
	Enter **?** to display what you must enter next on the command line. In this example, you must enter an IP subnet mask.
	
	A <cr> is not displayed; therefore, you must enter additional keywords or arguments to complete the command.

`Router(config-if)# ip address 172.16.0.1 255.255.255.0 ?`
`  secondary          Make this IP address a secondary address`
`  <cr>`
`Router(config-if)# ip address 172.16.0.1 255.255.255.0`
	Enter the IP subnet mask. This example uses the 255.255.255.0 IP subnet mask.
	
	Enter **?** to display what you must enter next on the command line. In this example, you can enter the **secondary** keyword, or you can press **Enter**.
	
	A <cr> is displayed; you can press **Enter** to complete the command, or you can enter another keyword.

`Router(config-if)# ip address 172.16.0.1 255.255.255.0` `Router(config-if)#`	In this example, Enter is pressed to complete the command.

Using the no and default Forms of Commands

Almost every configuration command has a **no** form. In general, use the **no** form to disable a function. Use the command without the **no** keyword to reenable a disabled function or to enable a function that is disabled by default. For example, IP routing is enabled by default. To disable IP routing, use the **no ip routing** command; to reenable IP routing, use the **ip routing** command. The Cisco IOS software command reference publications provide the complete syntax for the configuration commands and describe what the **no** form of a command does.

Configuration commands also can have a **default** form, which returns the command settings to the default values. Most commands are disabled by default, so in such cases using the **default** form has the same result as using the **no** form of the command. However, some commands are enabled by default and have variables set to certain default values. In these cases, the **default** form of the command enables the command and sets the variables to their default values. The Cisco IOS software command reference publications describe the effect of the **default** form of a command if the command functions differently than the **no** form.

Saving Configuration Changes

Use the **copy system:running-config nvram:startup-config** command to save your configuration changes to the startup configuration so that the changes will not be lost if the software reloads or a power outage occurs. For example:

```
Router# copy system:running-config nvram:startup-config
Building configuration...
```

It might take a minute or two to save the configuration. After the configuration has been saved, the following output appears:

```
[OK]
Router#
```

On most platforms, this task saves the configuration to NVRAM. On the Class A Flash file system platforms, this task saves the configuration to the location specified by the CONFIG_FILE environment variable. The CONFIG_FILE variable defaults to NVRAM.

Filtering Output from the show and more Commands

In Cisco IOS Release 12.0(1)T and later releases, you can search and filter the output of **show** and **more** commands. This functionality is useful if you need to sort through large amounts of output or if you want to exclude output that you need not see.

To use this functionality, enter a **show** or **more** command followed by the "pipe" character (|); one of the keywords **begin**, **include**, or **exclude**; and a regular expression on which you want to search or filter (the expression is case-sensitive):

command | {**begin** | **include** | **exclude**} *regular-expression*

The output matches certain lines of information in the configuration file. The following example illustrates how to use output modifiers with the **show interface** command when you want the output to include only lines in which the expression "protocol" appears:

```
Router# show interface | include protocol

FastEthernet0/0 is up, line protocol is up
Serial4/0 is up, line protocol is up
Serial4/1 is up, line protocol is up
Serial4/2 is administratively down, line protocol is down
Serial4/3 is administratively down, line protocol is down
```

For more information on the search and filter functionality, refer to the "Using the Command-Line Interface" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

Identifying Supported Platforms

Cisco IOS software is packaged in feature sets consisting of software images that support specific platforms. The feature sets available for a specific platform depend on which Cisco IOS software images are included in a release. To identify the set of software images available in a specific release or to find out if a feature is available in a given Cisco IOS software image, see the following sections:

- Using Feature Navigator
- Using Software Release Notes

Using Feature Navigator

Feature Navigator is a web-based tool that enables you to quickly determine which Cisco IOS software images support a particular set of features and which features are supported in a particular Cisco IOS image.

Feature Navigator is available 24 hours a day, 7 days a week. To access Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, e-mail the Contact Database Administration group at cdbadmin@cisco.com. If you do not have an account on Cisco.com, go to http://www.cisco.com/register and follow the directions to establish an account.

To use Feature Navigator, you must have a JavaScript-enabled web browser such as Netscape 3.0 or later, or Internet Explorer 4.0 or later. Internet Explorer 4.0 always has JavaScript enabled. To enable JavaScript for Netscape 3.x or Netscape 4.x, follow the instructions provided with the web browser. For JavaScript support and enabling instructions for other browsers, check with the browser vendor.

Feature Navigator is updated when major Cisco IOS software releases and technology releases occur. You can access Feature Navigator at the following URL:

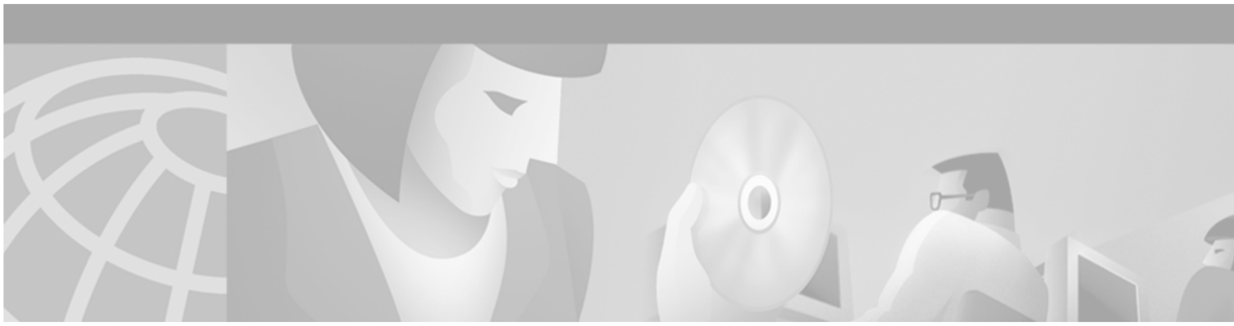
http://www.cisco.com/go/fn

Using Software Release Notes

Cisco IOS software releases include release notes that provide the following information:

- Platform support information
- Memory recommendations
- Microcode support information
- Feature set tables
- Feature descriptions
- Open and resolved severity 1 and 2 caveats for all platforms

Release notes are intended to be release-specific for the most current release, and the information provided in these documents may not be cumulative in providing information about features that first appeared in previous releases.



Using Debug Commands

This chapter explains how you use **debug** commands to diagnose and resolve internetworking problems. Specifically, it covers the following topics:

- Entering **debug** commands
- Using the **debug ?** command
- Using the **debug all** command
- Generating **debug** command output
- Redirecting **debug** and error message output

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 staff. 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.

Entering debug Commands

All **debug** commands are entered in privileged EXEC mode, and most **debug** commands take no arguments. For example, to enable the **debug isdn q931** command, enter the following the command line in privileged EXEC mode at :

debug isdn q931

To turn off the **debug isdn q931** command, enter the **no** form of the command at the command line in privileged EXEC mode:

no debug isdn q931

Alternately, you can enter the **undebug** form of the command in privileged EXEC mode:

undebug isdn q931

To display the state of each debugging option, enter the following at the command line in privileged EXEC mode:

show debugging

Using the debug ? Command

To list and see a brief description of all the debugging command options, enter the following command in privileged EXEC mode at the command line:

debug ?

Not all debugging commands listed in the **debug ?** output are described in this document. Commands are included here based on the their usefulness in assisting you to diagnose network problems. Commands not included are typically used internally by Cisco engineers during the development process and are not intended for use outside the Cisco environment.

Using the debug all Command

To enable all system diagnostics, enter the following command at the command line in privileged EXEC mode:

debug all

The **no debug all** command turns off all diagnostic output. Using the **no debug all** command is a convenient way to ensure that you have not accidentally left any **debug** commands turned on.

Caution Because debugging output takes priority over other network traffic, and because the **debug all** command generates more output than any other **debug** command, it can severely diminish the performance of the router or even render it unusable. In virtually all cases, it is best to use more specific **debug** commands.

Generating debug Command Output

Enabling a **debug** command can result in output similar to the following example for the **debug modem** command:

```
Router# debug modem

15:25:51: TTY4: DSR came up
15:25:51: tty4: Modem: IDLE->READY
15:25:51: TTY4: Autoselect started
15:27:51: TTY4: Autoselect failed
15:27:51: TTY4: Line reset
15:27:51: TTY4: Modem: READY->HANGUP
15:27:52: TTY4: dropping DTR, hanging up
15:27:52: tty4: Modem: HANGUP->IDLE
15:27:57: TTY4: restoring DTR
15:27:58: TTY4: DSR came up
```

The router continues to generate such output until you enter the corresponding **no debug** command (in this case, the **no debug modem** command).

If you enable a **debug** command and no output is displayed, consider the following possibilities:

- The router may not be properly configured to generate the type of traffic you want to monitor. Use the **more system:running-config** EXEC command to check its configuration.
- Even if the router is properly configured, it may not generate the type of traffic you want to monitor during the particular period that debugging is turned on. Depending on the protocol you are debugging, you can use commands such as the TCP/IP **ping** EXEC command to generate network traffic.

Redirecting debug and Error Message Output

By default, the network server sends the output from **debug** commands and system error messages to the console. If you use this default, monitor debug output using a virtual terminal connection, rather than the console port.

To redirect debug output, use the **logging** command options within configuration mode as described in the following sections.

Possible destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. The syslog format is compatible with 4.3 Berkeley Standard Distribution (BSD) UNIX and its derivatives.

Note Be aware that the debugging destination you use affects system overhead. Logging to the console produces very high overhead, whereas logging to a virtual terminal produces less overhead. Logging to a syslog server produces even less, and logging to an internal buffer produces the least overhead of any method.

To configure message logging, you need to be in configuration command mode. To enter this mode, use the **configure terminal** command at the EXEC prompt.

Enabling Message Logging

To enable message logging to all supported destinations other than the console, enter the following command:

logging on

The default condition is **logging on**.

To direct logging to the console only and disable logging output to other destinations, enter the following command:

no logging on

Setting the Message Logging Levels

You can set the logging levels when logging messages to the following devices:

- Console
- Monitor
- Syslog server

Table 3 lists and briefly describes the logging levels and corresponding keywords you can use to set the logging levels for these types of messages. The highest level of message is level 0, emergencies. The lowest level is level 7, debugging, which also displays the greatest amount of messages. For information about limiting these messages, see sections later in this chapter.

Table 3 Message Logging Keywords and Levels

Level	Keyword	Description	Syslog Definition
0	**emergencies**	System is unusable.	LOG_EMERG
1	**alerts**	Immediate action is needed.	LOG_ALERT
2	**critical**	Critical conditions exist.	LOG_CRIT
3	**errors**	Error conditions exist.	LOG_ERR
4	**warnings**	Warning conditions exist.	LOG_WARNING
5	**notification**	Normal, but significant, conditions exist.	LOG_NOTICE
6	**informational**	Informational messages.	LOG_INFO
7	**debugging**	Debugging messages.	LOG_DEBUG

Limiting the Types of Logging Messages Sent to the Console

To limit the types of messages that are logged to the console, use the **logging console** router configuration command. The full syntax of this command follows:

logging console *level*

no logging console

The **logging console** command limits the logging messages displayed on the console to messages up to and including the specified severity level, which is specified by the *level* argument. The *level* argument is one of the logging keywords listed in Table 3. Keywords are listed in order from the most severe level to the least severe.

The **no logging console** command disables logging to the console.

The following example sets console logging of messages at the **debugging** level, which is the least severe level and which displays all logging messages:

```
logging console debugging
```

Logging Messages to an Internal Buffer

The default logging device is the console; all messages are displayed on the console unless otherwise specified.

To log messages to an internal buffer, use the **logging buffered** router configuration command. The full syntax of this command follows:

logging buffered

no logging buffered

The **logging buffered** command copies logging messages to an internal buffer instead of writing them to the console. The buffer is circular in nature, so newer messages overwrite older messages. To display the messages that are logged in the buffer, use the **show logging** privileged EXEC command. The first message displayed is the oldest message in the buffer.

The **no logging buffered** command cancels the use of the buffer and writes messages to the console (the default).

Limiting the Types of Logging Messages Sent to Another Monitor

To limit the level of messages logged to the terminal lines (monitors), use the **logging monitor** router configuration command. The full syntax of this command follows:

logging monitor *level*

no logging monitor

The **logging monitor** command limits the logging messages displayed on terminal lines other than the console line to messages with a level up to and including the specified *level* argument. The *level* argument is one of the logging keywords listed in Table 3. To display logging messages on a terminal (virtual console), use the **terminal monitor** privileged EXEC command.

The **no logging monitor** command disables logging to terminal lines other than the console line.

The following example sets the level of messages displayed on monitors other than the console to **notification**:

```
logging monitor notification
```

Logging Messages to a UNIX Syslog Server

To log messages to the syslog server host, use the **logging** router configuration command. The full syntax of this command follows:

logging *ip-address*

no logging *ip-address*

The **logging** command identifies a syslog server host to receive logging messages. The *ip-address* argument is the IP address of the host. By issuing this command more than once, you build a list of syslog servers that receive logging messages.

The **no logging** command deletes the syslog server with the specified address from the list of syslogs.

Limiting Messages to a Syslog Server

To limit the number of messages sent to the syslog servers, use the **logging trap** router configuration command. The full syntax of this command follows:

logging trap *level*

no logging trap

The **logging trap** command limits the logging messages sent to syslog servers to logging messages with a level up to and including the specified *level* argument. The *level* argument is one of the keywords listed in Table 3.

To send logging messages to a syslog server, specify its host address with the **logging** command.

The default trap level is **informational**.

The **no logging trap** command disables logging to syslog servers.

The current software generates four categories of syslog messages:

- Error messages about software or hardware malfunctions, displayed at the **errors** level.
- Interface up/down transitions and system restart messages, displayed at the **notification** level.
- Reload requests and low-process stack messages, displayed at the **informational** level.
- Output from the **debug** commands, displayed at the **debugging** level.

The **show logging** privileged EXEC command displays the addresses and levels associated with the current logging setup. The command output also includes ancillary statistics.

Example of Setting Up a UNIX Syslog Daemon

To set up the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the file /etc/syslog.conf:

```
local7.debugging /usr/adm/logs/tiplog
```

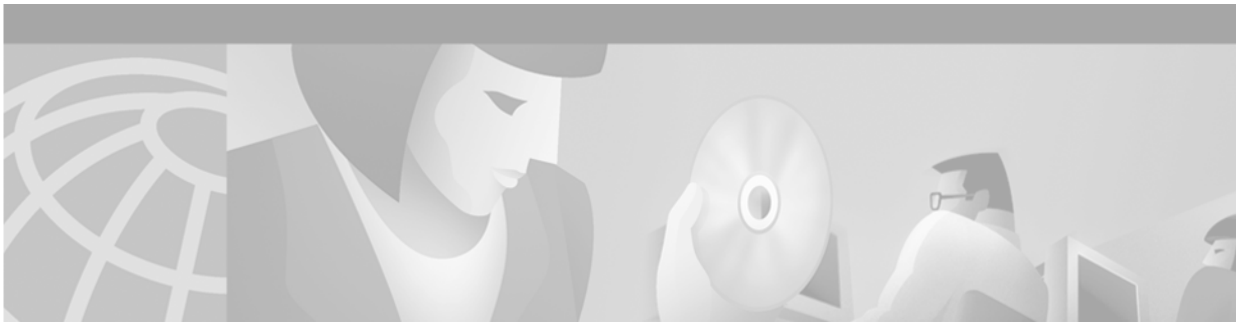
The **local7** keyword specifies the logging facility to be used.

The **debugging** keyword specifies the syslog level. See Table 3 for other keywords that can be listed.

The UNIX system sends messages at or above this level to the specified file, in this case /usr/adm/logs/tiplog. The file must already exist, and the syslog daemon must have permission to write to it.

For the System V UNIX systems, the line should read as follows:

```
local7.debug /usr/admin/logs/cisco.log
```



Conditionally Triggered Debugging

When the Conditionally Triggered Debugging feature is enabled, the router generates debugging messages for packets entering or leaving the router on a specified interface; the router will not generate debugging output for packets entering or leaving through a different interface. You can specify the interfaces explicitly. For example, you may only want to see debugging messages for one interface or subinterface. You can also turn on debugging for all interfaces that meet specified conditions. This feature is useful on dial access servers, which have a large number of ports.

Normally, the router will generate debugging messages for every interface, resulting in a large number of message that consume system resources and can make it difficult to find the specific information you need. By limiting the number of debugging messages, you can receive messages related to only the ports you want to troubleshoot.

The Conditionally Triggered Debugging feature controls the output from the following protocol-specific **debug** commands:

- **debug aaa** {**accounting** | **authorization** | **authentication**}
- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Although this feature limits the output of the listed commands, it does not automatically enable the generation of debugging output from these commands. Debugging messages are generated only when the protocol-specific **debug** command is enabled. The **debug** command output is controlled through two processes:

- The protocol-specific **debug** commands specify which protocols are being debugged. For example, the **debug dialer events** command generates debugging output related to dialer events.
- The **debug condition** commands limit these debugging messages to those related to a particular interface. For example, the **debug condition username cisco** command generates debugging output only for interfaces with packets that specify a username of cisco.

To configure Conditionally Triggered Debugging, perform the tasks described in the following sections:

- Enabling Protocol-Specific debug Commands
- Enabling Conditional Debugging Commands
- Specifying Multiple Conditions

Enabling Protocol-Specific debug Commands

To generate any debugging output, the protocol-specific **debug** command for the desired output must be enabled. Use the **show debugging** command to determine which types of debugging are enabled. Use the following commands in privileged EXEC mode to enable or disable the desired protocol-specific **debug** commands as needed:

Command	Purpose
show debugging	Determines which types of debugging are enabled.
debug *protocol*	Enables the desired debugging commands.
no debug *protocol*	Disables the debugging commands that are not desired.

If you want to have no output, disable all the protocol-specific **debug** commands.

Enabling Conditional Debugging Commands

If no **debug condition** commands are enabled, all debugging output, regardless of the interface, will be displayed for the enabled protocol-specific **debug** commands.

The first **debug condition** command you enter enables conditional debugging. The router will only display messages for interfaces that meet one of the specified conditions. If multiple conditions are specified, the interface must meet at least one of the conditions in order for messages to be displayed.

To enable messages for interfaces specified explicitly or for interfaces that meet certain conditions, perform the tasks described in the following sections:

- Displaying Messages for One Interface
- Displaying Messages for Multiple Interfaces
- Limiting Messages Based on Conditions

Displaying Messages for One Interface

To disable debugging messages for all interfaces except one, use the following command in privileged EXEC mode:

Command	Purpose
debug condition interface *interface*	Disables debugging messages for all interfaces except one.

If you enter the **debug condition interface** command, the debugging output will be turned off for all interfaces except the specified interface. To reenable debugging output for all interfaces, use the **no debug interface** command.

Displaying Messages for Multiple Interfaces

To enable debugging messages for multiple interfaces, use the following commands in privileged EXEC mode:

Command	Purpose
debug condition interface *interface*	Disables debugging messages for all interfaces except one.
debug condition interface *interface*	Enables debugging messages for additional interfaces. Repeat this task until debugging messages are enabled for all desired interfaces.

If you specify more than one interface by entering this command multiple times, debugging output will be displayed for all of the specified interfaces. To turn off debugging on a particular interface, use the **no debug interface** command. If you use the **no debug interface all** command or remove the last **debug interface** command, debugging output will be reenabled for all interfaces.

Limiting Messages Based on Conditions

The router can monitor interfaces to learn if any packets contain the specified value for one of the following conditions:

- Username
- Calling party number
- Called party number

If you enter a condition, such as calling number, debug output will be stopped for all interfaces. The router will then monitor every interface to learn if a packet with the specified calling party number is sent or received on any interfaces. If the condition is met on an interface or subinterface, **debug** command output will be displayed for that interface. The debugging output for an interface is "triggered" when the condition has been met. The debugging output continues to be disabled for the other interfaces. If at some later time the condition is met for another interface, then the debug output will become enabled for that interface as well.

Once debugging output has been triggered on an interface, the output will continue until the interface goes down. However, the session for that interface might change, resulting in a new username, called party number, or calling party number. Use the **no debug interface** command to reset the debug trigger mechanism for a particular interface. The debugging output for that interface will be disabled until the interface meets one of the specified conditions.

To limit debugging messages based on a specified condition, use the following command in privileged EXEC mode:

Command	Purpose
debug condition {**username** *username* | **called** *dial-string* | **caller** *dial-string*}	Enables conditional debugging. The router will display only messages for interfaces that meet this condition.

To reenable the debugging output for all interfaces, use the **no debug condition all** command.

Specifying Multiple Conditions

To limit debugging messages based on more than one condition, use the following commands in privileged EXEC mode as needed:

Command	Purpose
debug condition {**username** *username* | **called** *dial-string* | **caller** *dial-string*}	Enables conditional debugging and specifies the first condition.
debug condition {**username** *username* | **called** *dial-string* | **caller** *dial-string*}	Specifies the second condition. Repeat this task until all conditions are specified.

If you enter multiple **debug condition** commands, debugging output will be generated if an interface meets at least one of the conditions. If you use the **no debug condition** command to remove one of the conditions, using interfaces that meet only that condition will no longer produce debugging output. However, interfaces that meet a condition other than the removed condition will continue to generate output. Only if no active conditions are met for an interface will the output for that interface be disabled.

Conditionally Triggered Debugging Configuration Examples

In this example, four conditions have been set by the following commands:

- **debug condition interface serial 0**
- **debug condition interface serial 1**
- **debug condition interface virtual-template 1**
- **debug condition username cisco**

The first three conditions have been met by one interface. The fourth condition has not yet been met.

```
Router# show debug condition

Condition 1: interface Se0 (1 flags triggered)
        Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
        Flags: Se1
Condition 3: interface Vt1 (1 flags triggered)
        Flags: Vt1
Condition 4: username cisco (0 flags triggered)
```

When any **debug condition** command is entered, debugging messages for conditional debugging are enabled. The following debugging messages show conditions being met on different interfaces as serial interface 0 and serial interface 1 come up. For example, the second line of output indicates that serial interface 0 meets the username cisco condition.

```
*Mar  1 00:04:41.647: %LINK-3-UPDOWN: Interface Serial0, changed state to up
*Mar  1 00:04:41.715: Se0 Debug: Condition 4, username cisco triggered, count 2
*Mar  1 00:04:42.963: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to up
*Mar  1 00:04:43.271: Vi1 Debug: Condition 3, interface Vt1 triggered, count 1
*Mar  1 00:04:43.271: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Mar  1 00:04:43.279: Vi1 Debug: Condition 4, username cisco triggered, count 2
*Mar  1 00:04:43.283: Vi1 Debug: Condition 1, interface Se0 triggered, count 3
```

```
*Mar  1 00:04:44.039: %IP-4-DUPADDR: Duplicate address 172.27.32.114 on Ethernet 0,
sourced by 00e0.1e3e.2d41
*Mar  1 00:04:44.283: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
*Mar  1 00:04:54.667: %LINK-3-UPDOWN: Interface Serial1, changed state to up
*Mar  1 00:04:54.731: Se1 Debug: Condition 4, username cisco triggered, count 2
*Mar  1 00:04:54.735: Vi1 Debug: Condition 2, interface Se1 triggered, count 4
*Mar  1 00:04:55.735: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to up
```

After a period of time, the **show debug condition** command displays the revised list of conditions:

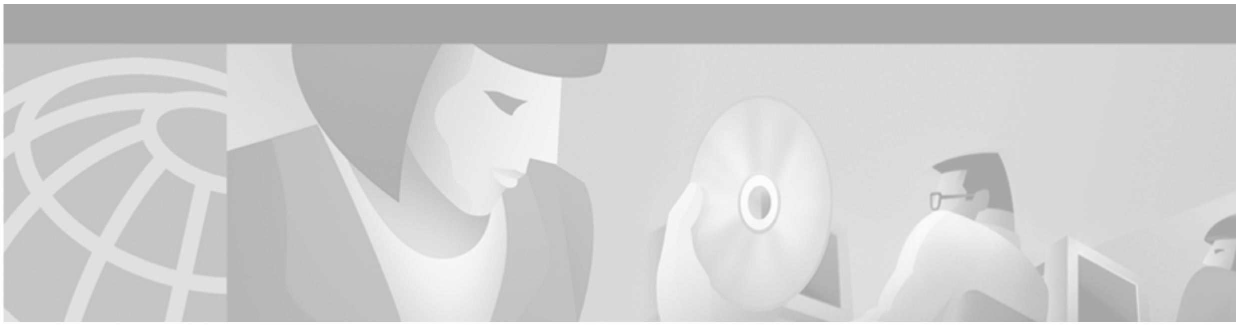
```
Router# show debug condition

Condition 1: interface Se0 (2 flags triggered)
        Flags: Se0 Vi1
Condition 2: interface Se1 (2 flags triggered)
        Flags: Se1 Vi1
Condition 3: interface Vt1 (2 flags triggered)
        Flags: Vt1 Vi1
Condition 4: username cisco (3 flags triggered)
        Flags: Se0 Vi1 Se1
```

Next, serial interface 1 and serial interface 0 go down. When an interface goes down, conditions for that interface are cleared.

```
*Mar  1 00:05:51.443: %LINK-3-UPDOWN: Interface Serial1, changed state to down
*Mar  1 00:05:51.471: Se1 Debug: Condition 4, username cisco cleared, count 1
*Mar  1 00:05:51.479: Vi1 Debug: Condition 2, interface Se1 cleared, count 3
*Mar  1 00:05:52.443: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to down
*Mar  1 00:05:56.859: %LINK-3-UPDOWN: Interface Serial0, changed state to down
*Mar  1 00:05:56.887: Se0 Debug: Condition 4, username cisco cleared, count 1
*Mar  1 00:05:56.895: Vi1 Debug: Condition 1, interface Se0 cleared, count 2
*Mar  1 00:05:56.899: Vi1 Debug: Condition 3, interface Vt1 cleared, count 1
*Mar  1 00:05:56.899: Vi1 Debug: Condition 4, username cisco cleared, count 0
*Mar  1 00:05:56.903: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
*Mar  1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to down
*Mar  1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to down
```

The final **show debug condition** output is the same as the output before the interfaces came up:

```
Router# show debug condition

Condition 1: interface Se0 (1 flags triggered)
        Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
        Flags: Se1
Condition 3: interface Vt1 (1 flags triggered)
        Flags: Vt1
Condition 4: username cisco (0 flags triggered)
```



Debug Commands

This chapter contains an alphabetical listing of the **debug** commands and their descriptions. Documentation for each command includes a brief description of its use, command syntax, usage guidelines, sample output, and a description of that output.

Output formats vary with each **debug** command. Some commands generate a single line of output per packet, whereas others generate multiple lines of output per packet. Some generate large amounts of output; others generate only occasional output. Some generate lines of text, and others generate information in field format. Thus, the way **debug** command output is documented also varies. For example, the output for **debug** commands that generate lines of text is usually described line by line, and the output for **debug** commands that generate information in field format is usually described in tables.

By default, the network server sends the output from the **debug** commands to the console. Sending output to a terminal (virtual console) produces less overhead than sending it to the console. Use the **terminal monitor** privileged EXEC command to send output to a terminal. For more information about redirecting output, see the “Using Debug Commands” chapter.

debug aaa accounting

To display information on accountable events as they occur, use the **debug aaa accounting** privileged EXEC command. To disable debugging output, use the **no** form of the command.

debug aaa accounting

no debug aaa accounting

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The information displayed by the **debug aaa accounting** command is independent of the accounting protocol used to transfer the accounting information to a server. Use the **debug tacacs** and **debug radius** protocol-specific commands to get more detailed information about protocol-level issues.

You can also use the **show accounting** command to step through all active sessions and to print all the accounting records for actively accounted functions. The **show accounting** command allows you to display the active "accountable events" on the system. It provides systems administrators a quick look at what is happening, and may also be useful for collecting information in the event of a data loss of some kind on the accounting server. The **show accounting** command displays additional data on the internal state of the authentication, authorization, and accounting (AAA) security system if **debug aaa accounting** is turned on as well.

Examples

The following is sample output from the **debug aaa accounting** command:

```
Router# debug aaa accounting

16:49:21: AAA/ACCT: EXEC acct start, line 10
16:49:32: AAA/ACCT: Connect start, line 10, glare
16:49:47: AAA/ACCT: Connection acct stop:
task_id=70 service=exec port=10 protocol=telnet address=172.31.3.78 cmd=glare bytes_in=308
bytes_out=76 paks_in=45 paks_out=54 elapsed_time=14
```

Related Commands

Command	Description
debug aaa authentication	Displays information on accountable events as they occur.
debug aaa authorization	Displays information on AAA/TACACS+ authorization.
debug radius	Displays information associated with the RADIUS.
debug tacacs	Displays information associated with the TACACS.

debug aaa authentication

To display information on AAA/Terminal Access Controller Access Control System Plus (TACACS+) authentication, use the **debug aaa authentication** privileged EXEC command. To disable debugging command, use the **no** form of the command.

debug aaa authentication

no debug aaa authentication

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to learn the methods of authentication being used and the results of these methods.

Examples

The following is sample output from the **debug aaa authentication** command. A single EXEC login that uses the "default" method list and the first method, TACACS+, is displayed. The TACACS+ server sends a GETUSER request to prompt for the username and then a GETPASS request to prompt for the password, and finally a PASS response to indicate a successful login. The number 50996740 is the session ID, which is unique for each authentication. Use this ID number to distinguish between different authentications if several are occurring concurrently.

```
Router# debug aaa authentication

6:50:12: AAA/AUTHEN: create_user user='' ruser='' port='tty19' rem_addr='172.31.60.15'
authen_type=1 service=1 priv=1
6:50:12: AAA/AUTHEN/START (0): port='tty19' list='' action=LOGIN service=LOGIN
6:50:12: AAA/AUTHEN/START (0): using "default" list
6:50:12: AAA/AUTHEN/START (50996740): Method=TACACS+
6:50:12: TAC+ (50996740): received authen response status = GETUSER
6:50:12: AAA/AUTHEN (50996740): status = GETUSER
6:50:15: AAA/AUTHEN/CONT (50996740): continue_login
6:50:15: AAA/AUTHEN (50996740): status = GETUSER
6:50:15: AAA/AUTHEN (50996740): Method=TACACS+
6:50:15: TAC+: send AUTHEN/CONT packet
6:50:15: TAC+ (50996740): received authen response status = GETPASS
6:50:15: AAA/AUTHEN (50996740): status = GETPASS
6:50:20: AAA/AUTHEN/CONT (50996740): continue_login
6:50:20: AAA/AUTHEN (50996740): status = GETPASS
6:50:20: AAA/AUTHEN (50996740): Method=TACACS+
6:50:20: TAC+: send AUTHEN/CONT packet
6:50:20: TAC+ (50996740): received authen response status = PASS
6:50:20: AAA/AUTHEN (50996740): status = PASS
```

debug aaa authorization

To display information on AAA/TACACS+ authorization, use the **debug aaa authorization** privileged EXEC command. To disable debugging output, use the **no** form of the command.

debug aaa authorization

no debug aaa authorization

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to learn the methods of authorization being used and the results of these methods.

Examples

The following is sample output from the **debug aaa authorization** command. In this display, an EXEC authorization for user “carrel” is performed. On the first line, the username is authorized. On the second and third lines, the attribute value (AV) pairs are authorized. The debug output displays a line for each AV pair that is authenticated. Next, the display indicates the authorization method used. The final line in the display indicates the status of the authorization process, which, in this case, has failed.

```
Router# debug aaa authorization

2:23:21: AAA/AUTHOR (0): user='carrel'
2:23:21: AAA/AUTHOR (0): send AV service=shell
2:23:21: AAA/AUTHOR (0): send AV cmd*
2:23:21: AAA/AUTHOR (342885561): Method=TACACS+
2:23:21: AAA/AUTHOR/TAC+ (342885561): user=carrel
2:23:21: AAA/AUTHOR/TAC+ (342885561): send AV service=shell
2:23:21: AAA/AUTHOR/TAC+ (342885561): send AV cmd*
2:23:21: AAA/AUTHOR (342885561): Post authorization status = FAIL
```

The **aaa authorization** command causes a request packet containing a series of AV pairs to be sent to the TACACS daemon as part of the authorization process. The daemon responds in one of the following three ways:

- Accepts the request as is
- Makes changes to the request
- Refuses the request, thereby refusing authorization

Table 4 describes AV pairs associated with the **debug aaa authorization** command that may show up in the debug output.

Table 4 Attribute Value Pairs for Authorization

Attribute Value	Description
service=arap	Authorization for the ARA protocol is being requested.
service=shell	Authorization for EXEC startup and command authorization is being requested.
service=ppp	Authorization for PPP is being requested.
service=slip	Authorization for SLIP is being requested.

Table 4 Attribute Value Pairs for Authorization (continued)

Attribute Value	Description
protocol=lcp	Authorization for LCP is being requested (lower layer of PPP).
protocol=ip	Used with service=slip and service=slip to indicate which protocol layer is being authorized.
protocol=ipx	Used with service=ppp to indicate which protocol layer is being authorized.
protocol=atalk	Used with service=ppp or service=arap to indicate which protocol layer is being authorized.
protocol=vines	Used with service=ppp for VINES over PPP.
protocol=unknown	Used for undefined or unsupported conditions.
cmd=*x*	Used with service=shell, if cmd=NULL, this is an authorization request to start an EXEC. If cmd is not NULL, this is a command authorization request and will contain the name of the command being authorized. For example, cmd=telnet.
cmd-arg=*x*	Used with service=shell. When performing command authorization, the name of the command is given by a cmd=*x* pair for each argument listed. For example, cmd-arg=archie.sura.net.
acl=*x*	Used with service=shell and service=arap. For ARA, this pair contains an access list number. For service=shell, this pair contains an access class number. For example, acl=2.
inacl=*x*	Used with service=ppp and protocol=ip. Contains an IP input access list for SLIP or PPP/IP. For example, inacl=2.
outacl=*x*	Used with service=ppp and protocol=ip. Contains an IP output access list for SLIP or PPP/IP. For example, outacl=4.
addr=*x*	Used with service=slip, service=ppp, and protocol=ip. Contains the IP address that the remote host should use when connecting via SLIP or PPP/IP. For example, addr=172.30.23.11.
routing=*x*	Used with service=slip, service=ppp, and protocol=ip. Equivalent in function to the /routing flag in SLIP and PPP commands. Can either be true or false. For example, routing=true.
timeout=*x*	Used with service=arap. The number of minutes before an ARA session disconnects. For example, timeout=60.
autocmd=*x*	Used with service=shell and cmd=NULL. Specifies an autocommand to be executed at EXEC startup. For example, autocmd=telnet yxz.com.
noescape=*x*	Used with service=shell and cmd=NULL. Specifies a noescape option to the username configuration command. Can be either true or false. For example, noescape=true.
nohangup=*x*	Used with service=shell and cmd=NULL. Specifies a nohangup option to the username configuration command. Can be either true or false. For example, nohangup=false.

Table 4 Attribute Value Pairs for Authorization (continued)

Attribute Value	Description
priv-lvl=*x*	Used with service=shell and cmd=NULL. Specifies the current privilege level for command authorization as a number from 0 to 15. For example, priv-lvl=15.
zonelist=*x*	Used with service=arap. Specifies an AppleTalk zonelist for ARA. For example, zonelist=5.
addr-pool=*x*	Used with service=ppp and protocol=ip. Specifies the name of a local pool from which to get the address of the remote host.

debug aaa pod

To display debug messages related to POD packets, use the **debug aaa pod** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug aaa pod

no debug aaa pod

Syntax Description

This command has no keywords or arguments.

Defaults

Debugging for POD packets is not enabled.

Command History

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

Examples

The following example shows output from a successful POD request when the **show debug** command is used.

```
Router# debug aaa pod

AAA POD packet processing debugging is on
Router# show debug

General OS:
  AAA POD packet processing debugging is on
Router#
*Jul  9 16:04:32.271:POD:10.100.1.34 request queued
*Jul  9 16:04:32.271:POD:10.100.1.34 user  0.0.0.0 sessid 0x0 key 0xA5AFA004
*Jul  9 16:04:32.271:POD:     Line     User     IDB          Session Id Key
*Jul  9 16:04:32.271:POD:Skip Se0:21   meklund  0.0.0.0      0x0        0x0
*Jul  9 16:04:32.271:POD:KILL Se0:22   meklund  0.0.0.0      0x60000020 0xA5AFA004
*Jul  9 16:04:32.271:POD:Sending ACK to 10.100.1.34/1812
---
Interface Se0:22 was killed because the pod request contained a key of
0xA5AFA004 and pod was configured with the command

aaa pod server port 1812 auth-type any server-key mykey
```

Related Commands

Command	Description
aaa pod server	Enables the POD feature.

debug alps ascu

To enable debugging for ALPS ASCUs, use the **debug alps ascu** privileged EXEC command. To disable debugging, use the **no** form of this command.

debug alps ascu {**event** | **packet** | **detail** | **all** | **format** {**ipars** | **router** | **both**}} [*interface* [*ascu id*]]

no debug alps ascu {**event** | **packet** | **detail** | **all** | **format** {**ipars** | **router** | **both**}} [*interface* [*ascu id*]]

Syntax Description

event	Displays ASCU events or protocol errors.
packet	Displays sent or received packets.
detail	Displays all ASCU protocol events.
all	Enables event, packet, and detail debugging.
format {**ipars** | **router** | **both**}	Specifies how to display ASCU addresses and the hexadecimal data in the debug output: • **ipars**—Displays the IPARS hexadecimal output, only. • **router**—Displays the router hexadecimal output, only. • **both—**Displays both the IPARS and router hexadecimal output. The only difference between the IPARS output and the router output is the format of the hexadecimal data.
interface	(Optional) Enables debugging on a specified interface. Applies only to the **event**, **packet**, **detail**, and **all** keywords.
ascu id	(Optional) Enables debugging for a specified ASCU.

Defaults

Debugging is off.

Command History

Release	Modification
11.3(6)T	This command was introduced for limited availability.
12.0(1)	This command was available for general release.
12.0(5)T	This command was modified.
12.1(2)T	The **format**, **ipars**, **router**, and **both** keywords were added. The output for this command was modified to include IPARS and router formats.

Usage Guidelines

To enable debugging for a group of ASCUs, enter a separate command for each ASCU interface and IA combination.

The *interface* option applies only to the **event**, **packet**, **detail**, and **all** keywords.

Note To specify the particular debug tracing level (**event**, **packet**, **detail** or **all**) *and* the format (router, pairs or both), you must configure the **debug alps ascu** command two times: once to configure the debug tracing level and once to configure the format.

Note To log messages to an internal buffer, use the **logging buffered** global configuration command. To display the state of logging (syslog), use the **show logging** privileged EXEC command. For information on these commands and other commands used to customize logs, refer to the *Cisco IOS Configuration Fundamentals Configuration Guide* and *Cisco IOS Configuration Fundamentals Command Reference*.

Examples

The following output is from the **debug alps ascu event** command, showing events or protocol errors in **router** format for ASCU 42 on interface Serial7:

```
Router# debug alps ascu format router

Router# debug alps ascu event Serial7 42

ALPS ASCU: T1 expired for ascu 42 on i/f Serial7
ALPS ASCU: DOWN event while UP for ascu 42 on i/f Serial7 : C1 count = 1
```

Note If you specify the **ipars** or **both** format for the **event** or **detail** tracing level, both the IPARS and router formats will be displayed.

The following output is from the **debug alps ascu event** command, showing events or protocol errors in **ipars** format for ASCU 42 on interface Serial7:

```
Router# debug alps ascu format ipars

Router# debug alps ascu event Serial7 42

ALPS ASCU: T1 expired for ascu 42/2F on i/f Serial7
ALPS ASCU: DOWN event while UP for ascu 42/2F on i/f Serial7 : C1 count = 1
```

The following output is from the **debug alps ascu detail** command, showing all protocol events in **router** format for ASCU 42 on interface Serial6:

```
Router# debug alps ascu format router

Router# debug alps ascu detail Serial6 42

ALPS ASCU: Tx ALC POLL MSG (+ 0 pad bytes) to ascu 42 on i/f Serial6
ALPS ASCU: ALC GO AHD MSG rcvd from ascu 42 on i/f Serial6
ALPS ASCU: Tx ALC POLL MSG (+ 0 pad bytes) to ascu 42 on i/f Serial6
ALPS ASCU: ALC GO AHD MSG rcvd from ascu 42 on i/f Serial6
ALPS ASCU: Tx ALC POLL MSG (+ 0 pad bytes) to ascu 42 on i/f Serial6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42 on i/f Serial6, fwd to ckt
RTP_MATIP
ALPS ASCU: ALC GO AHD MSG rcvd from ascu 42 on i/f Serial6
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42 on i/f Serial6
ALPS ASCU: Tx ALC POLL MSG (3 bytes + CCC + 0 pad bytes) to ascu 42 on i/f Serial6
```

Note If you specify the **ipars** or **both** format for the **event** or **detail** tracing level, both the IPARS and router formats will be displayed.

The following output is from the **debug alps ascu detail** command, showing all protocol events in **both** format for ASCU 42 on interface Serial6:

```
Router# debug alps ascu format both

Router# debug alps ascu detail Serial6 42

ALPS ASCU: Tx ALC POLL MSG (+ 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS ASCU: ALC GO AHD MSG rcvd from ascu 42/2F on i/f Serial6
ALPS ASCU: Tx ALC POLL MSG (+ 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS ASCU: ALC GO AHD MSG rcvd from ascu 42/2F on i/f Serial6
ALPS ASCU: Tx ALC POLL MSG (+ 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42/2F on i/f Serial6, fwd to ckt
RTP_MATIP
ALPS ASCU: ALC GO AHD MSG rcvd from ascu 42/2F on i/f Serial6
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS ASCU: Tx ALC POLL MSG (3 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
```

The following output is from the **debug alps ascu packet** command, showing all packets sent or received in **router** format for ASCU 42 on interface Serial6:

```
Router# debug alps ascu packet Serial6 42

ALPS ASCU: Tx ALC SERVICE MSG (18 bytes + CCC + 0 pad bytes) to ascu 42 on i/f Serial6
02321D26 0C261616
140C0D18 26163135 0611C6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42 on i/f Serial6, fwd ckt
RTP_MATIP
42607866 65717866
65717966 755124
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42 on i/f Serial6
022038 26253138
26253139 263511E4
```

The following output is from the **debug alps ascu packet** command, showing all packets sent or received in **ipars** format for ASCU 42 on interface Serial6:

```
Router# debug alps ascu packet Serial6 42

ALPS ASCU: Tx ALC SERVICE MSG (18 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS IPARS Format:
2F2C1126 33262525
35331339 26251C14 271DC6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42/2F on i/f Serial6, fwd ckt
RTP_MATIP
ALPS IPARS Format:
2F3E3826 161C3826
161C1826 141D24
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS IPARS Format:
2F3E38 26161C38
26161C18 26141DE4
```

The following output is from the **debug alps ascu packet** command, showing all packets sent or received in **both** format for ASCU 42 on interface Serial6:

```
Router# debug alps ascu packet Serial6 42
```

```
ALPS ASCU: Tx ALC SERVICE MSG (18 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS Router Format:
02321D26 0C261616
140C0D18 26163135 0611C6
ALPS IPARS Format:
2F2C1126 33262525
35331339 26251C14 271DC6
ALPS ASCU: Rx ALC DATA MSG (14 bytes + CCC) from ascu 42/2F on i/f Serial6, fwd ckt
RTP_MATIP
ALPS Router Format:
42607866 65717866
65717966 755124
ALPS IPARS Format:
2F3E3826 161C3826
161C1826 141D24
ALPS ASCU: Tx ALC DATA MSG (14 bytes + CCC + 0 pad bytes) to ascu 42/2F on i/f Serial6
ALPS Router Format:
022038 26253138
26253139 263511E4
ALPS IPARS Format:
2F3E38 26161C38
26161C18 26141DE4
```

debug alps circuit event

To enable event debugging for ALPS circuits, use the **debug alps circuit event** privileged EXEC command. To disable debugging, use the **no** form of this command.

debug alps circuit event [*name*]

no debug alps circuit event [*name*]

Syntax Description

name	(Optional) Name given to identify an ALPS circuit on the remote CPE.

Defaults

If no circuit name is specified, then debugging is enabled for every ALPS circuit.

Command History

Release	Modification
11.3 T	This command was introduced.

Usage Guidelines

To enable debugging for a single ALPS circuit, specify the name of the circuit.

To enable debugging for a group of circuits, enter a separate command for each circuit name.

Examples

The following is sample output from the **debug alps circuit event** command for circuit RTP_AX25:

```
alps-rcpe# debug alps circuit event RTP_AX25

ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= OPEN, Event= DISABLE:
(CloseAndDisable)->DISC
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= DISC, Event= ENABLE:
(TmrStartNullRetry)->INOP
ALPS P1024 CKT: Ckt= RTP_AX25, Open - peer set to 200.100.40.2
ALPS P1024 CKT: Ckt= RTP_AX25, Open - peer open.
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= INOP, Event= RETRY_TIMEOUT:
(Open)->OPNG
ALPS P1024 CKT: FSM - Ckt= RTP_AX25, State= OPNG, Event= CKT_OPEN_CFM:
(CacheAndFwdAscuData)->OPEN

alps-ccpe# debug alps circuit event RTP_AX25

ALPS AX.25 FSM: Ckt= RTP_AX25, State= OPEN, Event= CktClose, Rsn= 12:
(PvcKill,CktRemove,TmrStartClose)->INOP
ALPS AX.25 FSM: Ckt= RTP_AX25, State= INOP, Event= X25PvcInact, Rsn= 0:
(-,-,-)->INOP
ALPS AX.25 FSM: Ckt= RTP_AX25, State= INOP, Event= X25VcDeleted, Rsn= 0:
(-,CktDestroy,TmrStop)->INOP
ALPS AX.25 FSM: Ckt= RTP_AX25, State= INOP, Event= CktOpReq, Rsn= 4:
(PvcMake,CktAdd,TmrStartOpen)->OPNG
ALPS AX.25 FSM: Ckt= RTP_AX25, State= OPNG, Event= X25ResetTx, Rsn= 0:
(-,-,-)->OPNG
ALPS AX.25 FSM: Ckt= RTP_AX25, State= OPNG, Event= X25VcUp, Rsn= 0:
(-,OpnCfm,TmrStop)->OPEN
```

debug alps peer

To enable event or packet debugging for ALPS peers, use the **debug alps peer** privileged EXEC command. To disable debugging, use the **no** form of this command.

debug alps peer {**event** | **packet**} [*ip-address*]

no debug alps peer {**event** | **packet**} [*ip-address*]

Syntax Description

event	Specifies debugging for an event.
packet	Specifies debugging for a packet.
ip-address	(Optional) Remote peer IP address.

Defaults

If no IP address is specified, then debugging is enabled for every peer connection.

Command History

Release	Modification
11.3(6)T	This command was introduced for limited availability.
12.0(1)	This command was available for general release.
12.0(5)T	The **packet** keyword was added. The format for the output was modified for consistency.

Usage Guidelines

To enable debugging for a single remote ALPS peer, specify the peer IP address.

To enable debugging for a set of remote peers, enter the command for each peer IP address.

Examples

The following output is from the **debug alps peer packet** command:

```
Router# debug alps peer packet

ALPS PEER:Peer (10.227.50.106, MATIP_A_CKT-1) - TX Peer Data Msg (18 bytes)
040A5320:                        01 00001241
040A5330:45546B5F 6F4F7757 67477B5B 51
ALPS PEER:Peer (10.227.50.106, MATIP_A_CKT-1) - RX Peer Data Msg (18 bytes)
04000550:         01000012 4145546B 5F6F4F77
04000560:5767477B 5B51
ALPS PEER:Peer (10.227.50.106, MATIP_A_CKT-1) - TX Peer Data Msg (18 bytes)
0409F6E0:               01 00001241 45546B5F
0409F6F0:6F4F7757 67477B5B 51
ALPS PEER:Peer (10.227.50.106, MATIP_A_CKT-1) - RX Peer Data Msg (18 bytes)
04000680:                  01000012 4145546B
04000690:5F6F4F77 5767477B 5B51
```

debug alps peer event

To enable event debugging for ALPS peers, use the **debug alps peer event** privileged EXEC command. To disable debugging, use the **no** form of this command.

debug alps peer event *ipaddr*

no debug alps peer event *ipaddr*

Syntax Description

ipaddr	(Optional) Peer IP address.

Defaults

If no IP address is specified, then debugging is enabled for every peer connection.

Command History

Release	Modification
11.3 T	This command was introduced.

Usage Guidelines

To enable debugging for a single remote ALPS peer, specify the peer IP address.

To enable debugging for a set of remote peers, enter the command for each peer IP address.

Examples

The following is sample output from the **debug alps peer event** command:

```
Router# debug alps peer event
ALPS PEER: FSM - Peer 200.100.25.2, Event ALPS_CLOSED_IND, State OPENED
ALPS PEER: peer 200.100.25.2 closed - closing peer circuits.
ALPS PEER: Promiscuous peer created for 200.100.25.2
ALPS PEER: TCP Listen - passive open 200.100.25.2(11003) -> 10000
ALPS PEER: FSM - Peer 200.100.25.2, Event ALPS_OPEN_IND, State DISCONN
ALPS PEER: peer 200.100.25.2 opened OK.
```

debug alps snmp

To enable debugging for ALPS SNMP agents, use the **debug alps snmp** privileged EXEC command. To disable debugging, use the **no** form of this command.

debug alps snmp

no debug alps snmp

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for SNMP agents is not enabled.

Command History

Release	Modification
11.3(6)T	This command was introduced for limited availability.
12.0(1)T	This command was available for general release.
12.0(5)T	This command was added to the documentation.
12.1(2)T	The output for this command was modified to reflect MIB and SNMP changes.

Examples

The following output is from the **debug alps snmp** command. The first line shows a circuit event status change. The second line shows an ASCU status change. The third line shows a peer connection status change.

```
ALPS CktStatusChange Notification for circuit CKT-1
ALPS AscuStatusChange Notification for ascu (Serial3, 41)
PeerConnStatusChange Notification for peer (10.227.50.106, MATIP_A_CKT-1)
```

The following output is from the **debug alps snmp** command, showing that an open failure has occurred on circuit 1:

```
ALPS CktOpenFailure Notification for circuit CKT1
```

The following output is from the **debug alps snmp** command, showing that a partial rejection to an ALPS circuit peer open request has occurred on circuit 1:

```
ALPS CktPartialReject Notification for ascu (Serial2, 41) on circuit CKT1
```

debug apple arp

To enable debugging of the AppleTalk Address Resolution Protocol (AARP), use the **debug apple arp** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple arp [*type number*]

no debug apple arp [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

This command is helpful when you experience problems communicating with a node on the network you control (a neighbor). If the **debug apple arp** display indicates that the router is receiving AARP probes, you can assume that the problem does not reside at the physical layer.

Examples

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

```
Router# debug apple arp

Ether0: AARP: Sent resolve for 4160.26
Ether0: AARP: Reply from 4160.26(0000.0c00.0453) for 4160.154(0000.0c00.8ea9)
Ether0: AARP: Resolved waiting request for 4160.26(0000.0c00.0453)
Ether0: AARP: Reply from 4160.19(0000.0c00.0082) for 4160.154(0000.0c00.8ea9)
Ether0: AARP: Resolved waiting request for 4160.19(0000.0c00.0082)
Ether0: AARP: Reply from 4160.19(0000.0c00.0082) for 4160.154(0000.0c00.8ea9)
```

Explanations for representative lines of output follow.

The following line indicates that the router has requested the hardware MAC address of the host at network address 4160.26:

```
Ether0: AARP: Sent resolve for 4160.26
```

The following line indicates that the host at network address 4160.26 has replied, giving its MAC address (0000.0c00.0453). For completeness, the message also shows the network address to which the reply was sent and its hardware MAC address (also in parentheses).

```
Ether0: AARP: Reply from 4160.26(0000.0c00.0453) for 4160.154(0000.0c00.8ea9)
```

The following line indicates that the MAC address request is complete:

```
Ether0: AARP: Resolved waiting request for 4160.26(0000.0c00.0453)
```

debug apple domain

To enable debugging of the AppleTalk domain activities, use the **debug apple domain** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple domain

no debug apple domain

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug apple domain** command to observe activity for domains and subdomains. Use this command in conjunction with the **debug apple remap** command to observe interaction between remapping and domain activity. Messages are displayed when the state of a domain changes, such as creating a new domain, deleting a domain, and updating a domain.

Examples

The following is sample output from the **debug apple domain** command intermixed with output from the **debug apple remap** command; the two commands show related events:

```
Router# debug apple domain

Router# debug apple remap

AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1
AT-REMAP: ReshuffleRemapList for subdomain 1
AT-REMAP: Could not find a remap for cable 3000-3001
AT-DOMAIN: atdomain_DisablePort for Tunnel0
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: Disabling interface Ethernet1
AT-DOMAIN: atdomain_DisablePort for Ethernet1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-REMAP: Remap for net 70 inbound subdomain 1 has been deleted
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: CleanSubDomain for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1 Remapped Net 10000
AT-REMAP: Remap for net 50 outbound subdomain 1 has been deleted
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: CleanSubDomain for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
```

Related Commands

Command	Description
debug apple remap	Enables debugging of the AppleTalk remap activities.

debug apple eigrp-all

To enable debugging output from the Enhanced IGRP routines, use the **debug apple eigrp-all** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple eigrp-all

no debug apple eigrp-all

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug apple eigrp-all** command can be used to monitor acquisition of routes, aging route table entries, and advertisement of known routes through Enhanced IGRP.

Caution Because the **debug apple eigrp-all** command can generate many messages, use it only when the CPU utilization of the router is less than 50 percent.

Examples

The following is sample output from the **debug apple eigrp-all** command:

```
Router# debug apple eigrp-all

3:54:34: atigrp2_router: peer is 83.195
3:54:37: AT: atigrp2_write: about to send packet
3:54:37: Ethernet2: output AT packet: enctype UNKNOWN, size 65
3:54:37: 07FFFFFF0000FFFFFFFFFFFF00000C1485B00046|0041ACD100000053FF8F58585802059110
3:54:37: 000000000000000000000000000000010001000C010001000000000F0204000C0053005300
3:54:37: AT: atigrp2, src=Ethernet2:83.143, dst=83-83, size=52, EIGRP pkt sent
3:54:39: atigrp2_router: peer is 83.195
3:54:42: AT: atigrp2_write: about to send packet
3:54:42: Ethernet2: output AT packet: enctype UNKNOWN, size 65
3:54:42: 07FFFFFF0000FFFFFFFFFFFF00000C1485B00046|0041ACD100000053FF8F58585802059110
3:54:42: 000000000000000000000000000000010001000C010001000000000F0204000C0053005300
3:54:42: AT: atigrp2, src=Ethernet2:83.143, dst=83-83, size=52, EIGRP pkt sent
```

Table 5 describes the significant fields shown in the display.

Table 5 debug apple eigrp Field Descriptions

Field	Description
atigrp2_router:	AppleTalk address of the neighbor.
AT:	Indicates that this is an AppleTalk packet.
Ethernet2:	Name of the interface through which the router received the packet.
src=	Name of the interface sending the Enhanced IGRP packet, as well at its AppleTalk address.
dst=	Cable range of the destination of the packet.
size=	Size of the packet (in bytes).

debug apple errors

To display errors occurring in the AppleTalk network, use the **debug apple errors** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug apple errors [*type number*]

no debug apple errors [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

In a stable AppleTalk network, the **debug apple errors** command produces little output.

To solve encapsulation problems, enable **debug apple errors** and **debug apple packet** together.

Examples

The following is sample output from the **debug apple errors** command when a router is brought up with a zone that does not agree with the zone list of other routers on the network:

```
Router# debug apple errors

%AT-3-ZONEDISAGREES: Ethernet0: AppleTalk port disabled; zone list incompatible with
4160.19
%AT-3-ZONEDISAGREES: Ethernet0: AppleTalk port disabled; zone list incompatible with
4160.19
%AT-3-ZONEDISAGREES: Ethernet0: AppleTalk port disabled; zone list incompatible with
4160.19
```

As the output suggests, a single error message indicates zone list incompatibility; this message is sent out periodically until the condition is corrected or the **debug apple errors** command is turned off.

Most of the other messages that the **debug apple errors** command can generate are obscure or indicate a serious problem with the AppleTalk network. Some of these other messages follow.

In the following message, RTMPRsp, RTMPReq, ATP, AEP, ZIP, ADSP, or SNMP could replace NBP, and “llap dest not for us” could replace “wrong encapsulation”:

```
Packet discarded, src 4160.12-254,dst 4160.19-254,NBP,wrong encapsulation
```

In the following message, in addition to an invalid echo packet error, other possible errors are unsolicited AEP echo reply, unknown echo function, invalid ping packet, unknown ping function, and bad responder packet type:

```
Ethernet0: AppleTalk packet error; no source address available
AT: pak_reply: dubious reply creation, dst 4160.19
AT: Unable to get a buffer for reply to 4160.19

Processing error, src 4160.12-254,dst 4160.19-254,AEP, invalid echo packet
```

The **debug apple errors** command can print out additional messages when other debugging commands are also turned on. When you turn on both the **debug apple errors** and **debug apple events** commands, the following message can be generated:

```
Proc err, src 4160.12-254,dst 4160.19-254,ZIP,NetInfo Reply format is invalid
```

In the preceding message, in addition to the NetInfo Reply format is invalid error, other possible errors are NetInfoReply not for me, NetInfoReply ignored, NetInfoReply for operational net ignored, NetInfoReply from invalid port, unexpected NetInfoReply ignored, cannot establish primary zone, no primary has been set up, primary zone invalid, net information mismatch, multicast mismatch, and zones disagree.

When you turn on both the **debug apple errors** and **debug apple nbp** commands, the following message can be generated:

```
Processing error,...,NBP,NBP name invalid
```

In the preceding message, in addition to the NBP name invalid error, other possible errors are NBP type invalid, NBP zone invalid, not operational, error handling brrq, error handling proxy, NBP fwdreq unexpected, No route to srcnet, Proxy to “*” zone, Zone “*” from extended net, No zone info for “*”, and NBP zone unknown.

When you turn on both the **debug apple errors** and **debug apple routing** commands, the following message can be generated:

```
Processing error,...,RTMPReq, unknown RTMP request
```

In the preceding message, in addition to an unknown RTMP request error, other possible errors are RTMP packet header bad, RTMP cable mismatch, routed RTMP data, RTMP bad tuple, and Not Req or Rsp.

debug apple events

To display information about AppleTalk special events, neighbors becoming reachable or unreachable, and interfaces going up or down, use the **debug apple events** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple events [*type number*]

no debug apple events [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

Only significant events (for example, neighbor and route changes) are logged.

The **debug apple events** command is useful for solving AppleTalk network problems because it provides an overall picture of the stability of the network. In a stable network, the **debug apple events** command does not return any information. If the command generates numerous messages, those messages can indicate possible sources of the problems.

When configuring or making changes to a router or interface for AppleTalk, enable the **debug apple events** command to alert you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

The **debug apple events** command is also useful to determine whether network flapping (nodes toggling online and offline) is occurring. If flapping is excessive, look for routers that only support 254 networks.

When you enable the **debug apple events** command, you will see any messages that the **apple event-logging** configuration command normally displays. Turning on the **debug apple events** command, however, does not cause the **apple event-logging** command to be maintained in nonvolatile memory. Only turning on the **apple event-logging** command explicitly stores it in nonvolatile memory. Furthermore, if the **apple event-logging** command is already enabled, turning on or off the **debug apple events** command does not affect the **apple event-logging** command.

Examples

The following is sample output from the **debug apple events** command that describes a nonseed router coming up in discovery mode:

Discovery mode state changes

```
router# debug apple events

Ether0: AT: Resetting interface address filters
%AT-5-INTRESTART: Ether0: AppleTalk port restarting; protocol restarted
Ether0: AppleTalk state changed; unknown -> restarting
Ether0: AppleTalk state changed; restarting -> probing
%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 65401.148
Ether0: AppleTalk state changed; probing -> acquiring
%AT-6-ACQUIREMODE: Ether0: AT port initializing; acquiring net configuration
Ether0: AppleTalk state changed; acquiring -> restarting
Ether0: AppleTalk state changed; restarting -> line down
Ether0: AppleTalk state changed; line down -> restarting
Ether0: AppleTalk state changed; restarting -> probing
%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 4160.148
Ether0: AppleTalk state changed; probing -> acquiring
%AT-6-ACQUIREMODE: Ether0: AT port initializing; acquiring net configuration
Ether0: AppleTalk state changed; acquiring -> requesting zones
Ether0: AT: Resetting interface address filters
%AT-5-INTRESTART: Ether0: AppleTalk port restarting; protocol restarted
Ether0: AppleTalk state changed; requesting zones -> verifying
AT: Sent GetNetInfo request broadcast on Ethernet0
Ether0: AppleTalk state changed; verifying -> checking zones
Ether0: AppleTalk state changed; checking zones -> operational
```

S2542

As the output shows, the **debug apple events** command is useful in tracking the discovery mode state changes through which an interface progresses. When no problems are encountered, the state changes progress as follows:

1. Line down.
2. Restarting.
3. Probing (for its own address [node ID] using AARP).
4. Acquiring (sending out GetNetInfo requests).
5. Requesting zones (the list of zones for its cable).
6. Verifying (that the router's configuration is correct. If not, a port configuration mismatch is declared).
7. Checking zones (to make sure its list of zones is correct).
8. Operational (participating in routing).

Explanations for individual lines of output follow.

The following message indicates that a port is set. In this case, the zone multicast address is being reset.

```
Ether0: AT: Resetting interface address filters
```

The following messages indicate that the router is changing to restarting mode:

```
%AT-5-INTRESTART: Ether0: AppleTalk port restarting; protocol restarted
Ether0: AppleTalk state changed; unknown -> restarting
```

The following message indicates that the router is probing in the startup range of network numbers (65280 to 65534) to discover its network number:

```
Ether0: AppleTalk state changed; restarting -> probing
```

The following message indicates that the router is enabled as a nonrouting node using a provisional network number within its startup range of network numbers. This type of message only appears if the network address the router will use differs from its configured address. This is always the case for a discovery-enabled router; it is rarely the case for a nondiscovery-enabled router.

```
%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 65401.148
```

The following messages indicate that the router is sending out GetNetInfo requests to discover the default zone name and the actual network number range in which its network number can be chosen:

```
Ether0: AppleTalk state changed; probing -> acquiring
%AT-6-ACQUIREMODE: Ether0: AT port initializing; acquiring net configuration
```

Now that the router has acquired the cable configuration information, the following message indicates that it restarts using that information:

```
Ether0: AppleTalk state changed; acquiring -> restarting
```

The following messages indicate that the router is probing for its actual network address:

```
Ether0: AppleTalk state changed; restarting -> line down
Ether0: AppleTalk state changed; line down -> restarting
Ether0: AppleTalk state changed; restarting -> probing
```

The following message indicates that the router has found an actual network address to use:

```
%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 4160.148
```

The following messages indicate that the router is sending out GetNetInfo requests to verify the default zone name and the actual network number range from which its network number can be chosen:

```
Ether0: AppleTalk state changed; probing -> acquiring
%AT-6-ACQUIREMODE: Ether0: AT port initializing; acquiring net configuration
```

The following message indicates that the router is requesting the list of zones for its cable:

```
Ether0: AppleTalk state changed; acquiring -> requesting zones
```

The following messages indicate that the router is sending out GetNetInfo requests to make sure its understanding of the configuration is correct:

```
Ether0: AppleTalk state changed; requesting zones -> verifying
AT: Sent GetNetInfo request broadcast on Ethernet0
```

The following message indicates that the router is rechecking its list of zones for its cable:

```
Ether0: AppleTalk state changed; verifying -> checking zones
```

The following message indicates that the router is now fully operational as a routing node and can begin routing:

```
Ether0: AppleTalk state changed; checking zones -> operational
```

The following shows sample **debug apple events** output that describes a nondiscovery-enabled router coming up when no other router is on the wire.

```
router# debug apple events

Ethernet1: AT: Resetting interface address filters
%AT-5-INTRESTART: Ethernet1: AppleTalk port restarting; protocol restarted
Ethernet1: AppleTalk state changed; unknown -> restarting
Ethernet1: AppleTalk state changed; restarting -> probing
%AT-6-ADDRUSED: Ethernet1: AppleTalk node up; using address 4165.204
Ethernet1: AppleTalk state changed; probing -> verifying
AT: Sent GetNetInfo request broadcast on Ethernet1
Ethernet1: AppleTalk state changed; verifying -> operational
%AT-6-ONLYROUTER: Ethernet1: AppleTalk port enabled; no neighbors found
```

Indicates a nondiscovery-enabled router with no other router on the wire

S2543

As the output shows, a nondiscovery-enabled router can come up when no other router is on the wire; however, it must assume that its configuration (if accurate syntactically) is correct, because no other router can verify it. Notice that the last line indicates this situation.

The following is sample output from the **debug apple events** command that describes a discovery-enabled router coming up when there is no seed router on the wire:

```
Router# debug apple events

Ether0: AT: Resetting interface address filters
%AT-5-INTRESTART: Ether0: AppleTalk port restarting; protocol restarted
Ether0: AppleTalk state changed; unknown -> restarting
Ether0: AppleTalk state changed; restarting -> probing
%AT-6-ADDRUSED: Ether0: AppleTalk node up; using address 65401.148
Ether0: AppleTalk state changed; probing -> acquiring
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
AT: Sent GetNetInfo request broadcast on Ether0
```

As the output shows, when you attempt to bring up a nonseed router without a seed router on the wire, it never becomes operational; instead, it hangs in the acquiring mode and continues to send out periodic GetNetInfo requests.

The following is sample output from the **debug apple events** command when a nondiscovery-enabled router is brought up on an AppleTalk internetwork that is in compatibility mode (set up to accommodate extended as well as nonextended AppleTalk) and the router has violated internetwork compatibility:

.

```
router# debug apple events

E0: AT: Resetting interface address filters
%AT-5-INTRESTART: E0: AppleTalk port restarting; protocol restarted
E0: AppleTalk state changed; restarting -> probing
%AT-6-ADDRUSED: E0: AppleTalk node up; using address 41.19
E0: AppleTalk state changed; probing -> verifying
AT: Sent GetNetInfo request broadcast on Ethernet0
%AT-3-ZONEDISAGREES: E0: AT port disabled; zone list incompatible with 41.19
AT: Config error for E0, primary zone invalid
E0: AppleTalk state changed; verifying -> config mismatch
```

Indicates configuration mismatch

S2545

The following three configuration command lines indicate the part of the configuration of the router that caused the configuration mismatch:

```
lestat(config)#interface ethernet 0
lestat(config-if)#apple cab 41-41
lestat(config-if)#apple zone Marketing
```

The router shown had been configured with a cable range of 41-41 instead of 40-40, which would have been accurate. Additionally, the zone name was configured incorrectly; it should have been “Marketing,” rather than being misspelled as “Markting.”

debug apple nbp

To display debugging output from the Name Binding Protocol (NBP) routines, use the **debug apple nbp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug apple nbp [*type number*]

no debug apple nbp [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

To determine whether the router is receiving NBP lookups from a node on the AppleTalk network, enable **debug apple nbp** at each node between the router and the node in question to determine where the problem lies.

Caution Because the **debug apple nbp** command can generate many messages, use it only when the CPU utilization of the router is less than 50 percent.

Examples

The following is sample output from the **debug apple nbp** command:

```
Router# debug apple nbp

AT: NBP ctrl = LkUp, ntuples = 1, id = 77
AT: 4160.19, skt 2, enum 0, name: =:ciscoRouter@Low End SW Lab
AT: LkUp =:ciscoRouter@Low End SW Lab

AT: NBP ctrl = LkUp-Reply, ntuples = 1, id = 77
AT: 4160.154, skt 254, enum 1, name: lestat.Ether0:ciscoRouter@Low End SW Lab

AT: NBP ctrl = LkUp, ntuples = 1, id = 78
AT: 4160.19, skt 2, enum 0, name: =:IPADDRESS@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 79
AT: 4160.19, skt 2, enum 0, name: =:IPGATEWAY@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 83
AT: 4160.19, skt 2, enum 0, name: =:ciscoRouter@Low End SW Lab
AT: LkUp =:ciscoRouter@Low End SW Lab

AT: NBP ctrl = LkUp, ntuples = 1, id = 84
AT: 4160.19, skt 2, enum 0, name: =:IPADDRESS@Low End SW Lab

AT: NBP ctrl = LkUp, ntuples = 1, id = 85
AT: 4160.19, skt 2, enum 0, name: =:IPGATEWAY@Low End SW Lab
AT: NBP ctrl = LkUp, ntuples = 1, id = 85
AT: 4160.19, skt 2, enum 0, name: =:IPGATEWAY@Low End SW Lab
```

The first three lines describe an NBP lookup request:

```
AT: NBP ctrl = LkUp, ntuples = 1, id = 77
AT: 4160.19, skt 2, enum 0, name: =:ciscoRouter@Low End SW Lab
AT: LkUp =:ciscoRouter@Low End SW Lab
```

Table 6 describes the fields in the first line of output.

Table 6 debug apple nbp Field Descriptions

Field	Description
AT: NBP	Indicates that this message describes an AppleTalk NBP packet.
ctrl = LkUp	Identifies the type of NBP packet. Possible values are as follows: • LkUp—NBP lookup request. • LkUp-Reply—NBP lookup reply.
ntuples = 1	Indicates the number of name-address pairs in the lookup request packet. Range: 1 to 31 tuples.
id = 77	Identifies an NBP lookup request value.

Table 7 describes the fields in the second line of output.

Table 7 debug apple nbp Field Descriptions

Field	Description
AT:	Indicates that this message describes an AppleTalk packet.
4160.19	Indicates the network address of the requester.
skt 2	Indicates the internet socket address of the requester. The responder will send the NBP lookup reply to this socket address.
enum 0	Indicates the enumerator field. Used to identify multiple names registered on a single socket. Each tuple is assigned its own enumerator, incrementing from 0 for the first tuple.
name: =:ciscoRouter@Low End SW Lab	Indicates the entity name for which a network address has been requested. The AppleTalk entity name includes three components: • Object (in this case, a wildcard character [=], indicating that the requester is requesting name-address pairs for all objects of the specified type in the specified zone). • Type (in this case, ciscoRouter). • Zone (in this case, Low End SW Lab).

The third line in the output essentially reiterates the information in the two lines above it, indicating that a lookup request has been made regarding name-address pairs for all objects of the ciscoRouter type in the Low End SW Lab zone.

Because the router is defined as an object of type ciscoRouter in zone Low End SW Lab, the router sends an NBP lookup reply in response to this NBP lookup request. The following two lines of output show the response of the router:

```
AT: NBP ctrl = LkUp-Reply, ntuples = 1, id = 77
AT: 4160.154, skt 254, enum 1, name: lestat.Ether0:ciscoRouter@Low End SW Lab
```

In the first line, ctrl = LkUp-Reply identifies this NBP packet as an NBP lookup request. The same value in the id field (id = 77) associates this lookup reply with the previous lookup request. The second line indicates that the network address associated with the entity name of the router (lestat.Ether0:ciscoRouter@Low End SW Lab) is 4160.154. The fact that no other entity name/network address is listed indicates that the responder only knows about itself as an object of type ciscoRouter in zone Low End SW Lab.

debug apple packet

To display per-packet debugging output, use the **debug apple packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple packet [*type number*]

no debug apple packet [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

With this command, you can monitor the types of packets being slow switched. It displays at least one line of debugging output per AppleTalk packet processed.

The output reports information online when a packet is received or a transmission is attempted.

When invoked in conjunction with the **debug apple routing**, **debug apple zip**, and **debug apple nbp** commands, the **debug apple packet** command adds protocol processing information in addition to generic packet details. It also reports successful completion or failure information.

When invoked in conjunction with the **debug apple errors** command, the **debug apple packet** command reports packet-level problems, such as those concerning encapsulation.

Caution Because the **debug apple packet** command can generate many messages, use it only when the CPU utilization of the router is less than 50 percent.

Examples

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

```
Router# debug apple packet

Ether0: AppleTalk packet: enctype SNAP, size 60, encaps000000000000000000000000
AT: src=Ethernet0:4160.47, dst=4160-4160, size=10, 2 rtes, RTMP pkt sent
AT: ZIP Extended reply rcvd from 4160.19
AT: ZIP Extended reply rcvd from 4160.19
AT: src=Ethernet0:4160.47, dst=4160-4160, size=10, 2 rtes, RTMP pkt sent
Ether0: AppleTalk packet: enctype SNAP, size 60, encaps000000000000000000000000
Ether0: AppleTalk packet: enctype SNAP, size 60, encaps000000000000000000000000
```

Table 8 describes the fields in the first line of output.

Table 8 debug apple packet Field Descriptions

Field	Description
Ether0:	Name of the interface through which the router received the packet.
AppleTalk packet	Indicates that this is an AppleTalk packet.
enctype SNAP	Encapsulation type for the packet.

Table 8 debug apple packet Field Descriptions (continued)

Field	Description
size 60	Size of the packet (in bytes).
encaps000000000000000000000000	Encapsulation.

Table 9 describes the fields in the second line of output.

Table 9 debug apple packet Field Descriptions

Field	Description
AT:	Indicates that this is an AppleTalk packet.
src=Ethernet0:4160.47	Name of the interface sending the packet and its AppleTalk address.
dst=4160-4160	Cable range of the destination of the packet.
size=10	Size of the packet (in bytes.)
2 rtes	Indicates that two routes in the routing table link these two addresses.
RTMP pkt sent	Type of packet sent.

The third line indicates the type of packet received and its source AppleTalk address. This message is repeated in the fourth line because AppleTalk hosts can send multiple replies to a given GetNetInfo request.

debug apple remap

To enable debugging of the AppleTalk remap activities, use the **debug apple remap** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple remap

no debug apple remap

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug apple remap** command with the **debug apple domain** command to observe activity between domains and subdomains. Messages from the **debug apple remap** command are displayed when a particular remapping function occurs, such as creating remaps or deleting remaps.

Examples

The following is sample output from the **debug apple remap** command intermixed with output from the **debug apple domain** command; the two commands show related events.

```
Router# debug apple remap

Router# debug apple domain

AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1
AT-REMAP: ReshuffleRemapList for subdomain 1
AT-REMAP: Could not find a remap for cable 3000-3001
AT-DOMAIN: atdomain_DisablePort for Tunnel0
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: Disabling interface Ethernet1
AT-DOMAIN: atdomain_DisablePort for Ethernet1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-REMAP: Remap for net 70 inbound subdomain 1 has been deleted
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: CleanSubDomain for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-REMAP: RemapProcess for net 30000 domain AURP Domain 1 Remaped Net 10000
AT-REMAP: Remap for net 50 outbound subdomain 1 has been deleted
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
AT-DOMAIN: CleanUpDomain for domain 1 [AURP Domain 1]
AT-DOMAIN: CleanSubDomain for inbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for inbound subdomain 1
AT-DOMAIN: CleanSubDomain for outbound subdomain 1
AT-DOMAIN: DeleteRemapTable for subdomain 1
AT-DOMAIN: DeleteAvRemapList for outbound subdomain 1
```

Related Commands

Command	Description
debug apple domain	Enables debugging of the AppleTalk domain activities.

debug apple routing

To enable debugging output from the Routing Table Maintenance Protocol (RTMP) routines, use the **debug apple routing** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple routing [*type number*]

no debug apple routing [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

This command can be used to monitor acquisition of routes, aging of routing table entries, and advertisement of known routes. It also reports conflicting network numbers on the same network if the network is misconfigured.

Caution Because the **debug apple routing** command can generate many messages, use it only when router CPU utilization is less than 50 percent.

Examples

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

```
Router# debug apple routing

AT: src=Ethernet0:4160.41, dst=4160-4160, size=19, 2 rtes, RTMP pkt sent
AT: src=Ethernet1:41069.25, dst=41069, size=427, 96 rtes, RTMP pkt sent
AT: src=Ethernet2:4161.23, dst=4161-4161, size=427, 96 rtes, RTMP pkt sent
AT: Route ager starting (97 routes)
AT: Route ager finished (97 routes)
AT: RTMP from 4160.19 (new 0,old 94,bad 0,ign 0, dwn 0)
AT: RTMP from 4160.250 (new 0,old 0,bad 0,ign 2, dwn 0)
AT: RTMP from 4161.236 (new 0,old 94,bad 0,ign 1, dwn 0)
AT: src=Ethernet0:4160.41, dst=4160-4160, size=19, 2 rtes, RTMP pkt sent
```

Table 10 describes the fields in the first line of sample **debug apple routing** output.

Table 10 debug apple routing Field Descriptions

Field	Description
AT:	Indicates that this is AppleTalk debugging output.
src=Ethernet0:4160.41	Indicates the source router interface and network address for the RTMP update packet.
dst=4160-4160	Indicates the destination network address for the RTMP update packet.

Table 10 debug apple routing Field Descriptions (continued)

Field	Description
size=19	Displays the size of this RTMP packet (in bytes).
2 rtes	Indicates that this RTMP update packet includes information on two routes.
RTMP pkt sent	Indicates that this type of message describes an RTMP update packet that the router has sent (rather than one that it has received).

The following two messages indicate that the ager has started and finished the aging process for the routing table and that this table contains 97 entries:

```
AT: Route ager starting (97 routes)
AT: Route ager finished (97 routes)
```

Table 11 describes the fields in the following line of the **debug apple routing** command output:

```
AT: RTMP from 4160.19 (new 0,old 94,bad 0,ign 0, dwn 0)
```

Table 11 debug apple routing Field Descriptions

Field	Description
AT:	Indicates that this is AppleTalk debugging output.
RTMP from 4160.19	Indicates the source address of the RTMP update the router received.
new 0	Displays the number of routes in this RTMP update packet that the router did not already know about.
old 94	Displays the number of routes in this RTMP update packet that the router already knew about.
bad 0	Displays the number of routes the other router indicates have gone bad.
ign 0	Displays the number of routes the other router ignores.
dwn 0	Displays the number of poisoned tuples included in this packet.

debug apple zip

To display debugging output from the Zone Information Protocol (ZIP) routines, use the **debug apple zip** privileged EXEC command. The **no** form of this command disables debugging output.

debug apple zip [*type number*]

no debug apple zip [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Usage Guidelines

This command reports significant events such as the discovery of new zones and zone list queries. It generates information similar to that generated by the debug apple routing command, but generates it for ZIP packets instead of RTMP packets.

You can use the **debug apple zip** command to determine whether a ZIP storm is taking place in the AppleTalk network. You can detect the existence of a ZIP storm when you see that no router on a cable has the zone name corresponding to a network number that all the routers have in their routing tables.

Examples

The following is sample output from the **debug apple zip** command:

```
Router# debug apple zip

AT: Sent GetNetInfo request broadcast on Ether0
AT: Recvd ZIP cmd 6 from 4160.19-6
AT: 3 query packets sent to neighbor 4160.19
AT: 1 zones for 31902, ZIP XReply, src 4160.19
AT: net 31902, zonelen 10, name US-Florida
```

The first line indicates that the router has received an RTMP update that includes a new network number and is now requesting zone information:

```
AT: Sent GetNetInfo request broadcast on Ether0
```

The second line indicates that the neighbor at address 4160.19 replies to the zone request with a default zone:

```
AT: Recvd ZIP cmd 6 from 4160.19-6
```

The third line indicates that the router responds with three queries to the neighbor at network address 4160.19 for other zones on the network:

```
AT: 3 query packets sent to neighbor 4160.19
```

The fourth line indicates that the neighbor at network address 4160.19 responds with a ZIP extended reply, indicating that one zone has been assigned to network 31902:

```
AT: 1 zones for 31902, ZIP XReply, src 4160.19
```

The fifth line indicates that the router responds that the zone name of network 31902 is US-Florida, and the zone length of that zone name is 10:

```
AT: net 31902, zonelen 10, name US-Florida
```

debug appn all

To turn on all possible debugging messages for Advanced Peer-to-Peer Networking (APPN), use the **debug appn all** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn all

no debug appn all

Note Refer to the other forms of the **debug appn** command to enable specific debug output selectively.

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command shows all APPN events. Use other forms of the **debug appn** command to display specific types of events.

Caution Because the **debug appn all** command can generate many messages and alter timing in the network node, use it only when instructed by authorized support personnel.

Caution Debugging output takes priority over other network traffic. The **debug appn all** command generates more output than any other **debug appn** command and can alter timing in the network node. This command can severely diminish router performance or even render it unusable. In virtually all cases, it is best to use specific **debug appn** commands.

Examples

Refer to the documentation for specific **debug appn** commands for examples and explanations.

Related Commands

Command	Description
debug appn cs	Displays the APPN CS component activity.
debug appn ds	Displays debugging information on APPN DS component activity.
debug appn hpr	Displays information related to HPR code execution.
debug appn ms	Displays debugging information on APPN MS component activity.
debug appn nof	Displays information on APPN NOF component activity.
debug appn pc	Displays debugging information on APPN PC component activity.
debug appn ps	Displays debugging information on APPN PS component activity.
debug appn scm	Displays debugging information on APPN SCM component activity.
debug appn ss	Displays SS events.
debug appn trs	Displays debugging information on APPN TRS component activity.

debug appn cs

To display APPN Configuration Services (CS) component activity, use the **debug appn cs** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn cs

no debug appn cs

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The CS component is responsible for defining link stations, ports, and connection networks. It is responsible for the activation and deactivation of ports and link stations and handles status queries for these resources.

Examples

The following is sample output from the **debug appn cs** command. In this example a link station is being stopped.

```
Router# debug appn cs

Turned on event 008000FF

Router# appn stop link PATTY

APPN: ----- CS ----- Deq STOP_LS message
APPN: ----- CS ----- FSM LS: 75 17 5 8
APPN: ----- CS ----- Sending DEACTIVATE_AS - station PATTY
APPN: ----- CS ----- deactivate_as_p->ips_header.lpid = A80A60
APPN: ----- CS ----- deactivate_as_p->ips_header.lpid = A80A60
APPN: ----- CS ----- Sending DESTROY_TG to PC - station PATTY - lpid=A80A60
APPN: ----- CS ----- Deq DESTROY_TG - station PATTY
APPN: ----- CS ----- FSM LS: 22 27 8 0
APPN: ----- CS ----- Sending TG update for LS PATTY to TRS
APPN: ----- CS ----- ENTERING XID_PROCESSING: 4
%APPN-6-APPNSENDMSG: Link Station PATTY stopped
```

Table 12 describes the significant fields and messages shown in the display.

Table 12 debug appn cs Field Descriptions

Field	Description
APPN	APPN debugging output.
CS	CS component output.
Deq	CS received a message from another component.
FSM LS	Link station finite state machine is being referenced.
Sending	CS is sending a message to another component.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn ds

To display debugging information on APPN Directory Services (DS) component activity, use the **debug appn ds** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn ds

no debug appn ds

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The DS component manages searches for resources in the APPN network. DS is also responsible for registration of resources within the network.

Examples

The following is sample output from the **debug appn ds** command. In this example a search has been received.

```
Router# debug appn ds

Turned on event 080000FF
APPN: NEWDS: LS: search from: NETA.PATTY
APPN: NEWDS: pcid: DD3321E8B5667111
APPN: NEWDS: Invoking FSM NNSolu
APPN: NEWDS: LSfsm_NNSolu: 00A67AA0 pcid: DD3321E8B5667111 row: 0 col: 0 inp: 80200000
APPN: NEWDS: LSfsm_parent: 00A89940 row: 0 col: 0 inp: 80000000
APPN: NEWDS: Rcvd a LMRQ
APPN: NEWDS: LSfsm_NNSolu: 00A67AA0 pcid: DD3321E8B5667111 row: 12 col: 1 inp: 40000000
APPN: NEWDS: LSfsm_parent: 00A89940 row: 8 col: 1 inp: 40000000
APPN: NEWDS: LSfsm_child: 00A89BE8 row: 0 col: 0 inp: 80000080
APPN: NEWDS: PQenq REQUEST_ROUTE(RQ) to TRS
APPN: NEWDS: LSfsm_child: 00A8A1C0 row: 1 col: 0 inp: 80000008
APPN: NEWDS: LSfsm_NNSolu: 00A67AA0 pcid: DD3321E8B5667111 row: 5 col: 1 inp: 80C04000
APPN: NEWDS: LSfsm_child: 00A8A1C0 row: 7 col: 1 inp: 80844008
APPN: NEWDS: Rcvd a LMRY
APPN: NEWDS: LSfsm_NNSolu: 00A67AA0 pcid: DD3321E8B5667111 row: 16 col: 6 inp: 40800000
APPN: NEWDS: LSfsm_child: 00A8A1C0 row: 14 col: 5 inp: 40800000
APPN: NEWDS: LSfsm_parent: 00A89940 row: 3 col: 1 inp: 80840000
APPN: NEWDS: send locate to node: NETA.PATTY
```

Table 13 describes the significant fields in the display.

Table 13 debug appn ds Field Descriptions

Field	Description
APPN	APPN debugging output.
NEWDS	DS component output.
search from	Locate was received from NETA.PATTY.
LSfsm_	Locate Search finite state machine is being referenced.
PQenq	Message was sent to another component.

Table 13 debug appn ds Field Descriptions (continued)

Field	Description
Rcvd	Message was received from another component.
send locate	Locate will be sent to NETA.PATTY.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn hpr

To display debugging information related to High Performance Routing (HPR) code execution, use the **debug appn hpr** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn hpr

no debug appn hpr

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug appn hpr** command:

```
Router# debug appn hpr

APPN: -- ncl.ncl_map_dlc_type() -- mapping TOKEN_RING(4) to NCL_TR(3)
APPN: -- ncl.ncl_port() -- called with port_type:3, cisco_idb:893A14, hpr_ssap:C8
APPN: -- ncl.process_port_change() -- port coming up
APPN: -- ncl.process_port_change() -- PORT_UP
APPN: -- ncl.ncl_port_fsm -- FSM Invoked: Input:0, State:0->1, Action:0
APPN: -- ncl.ncl_port_fsm -- FSM Invoked: Input:1, State:1->2, Action:1
APPN: -- ncl.ncl_unmap_dlc_type() -- mapping NCL(3) to CLS(3)
APPN: ----- ANR  ----- Sending ACTIVATE_SAP.req
APPN: -- cswncsnd.main() -- received LSA_IPS ips.
APPN: -- ncl.ncl_port_fsm -- FSM Invoked: Input:3, State:2->3, Action:4
APPN: -- ncl.ncl_assign_anr() -- Assigned ANR,anr:8002
APPN: -- ncl.ncl_map_dlc_type() -- mapping TOKEN_RING(4) to NCL_TR(3)
APPN: -- ncl.ncl_populate_anr() -- anr:8002, dlc_type:3, idb 893A14
APPN: -- ncl.ncl_populate_anr() -- send anr_tbl_update to owning cswncsnd
APPN: -- ncl.ncl_ls_fsm -- FSM Invoked: Input:0, State:0->1, Action:0
APPN: ncl.ncl_send_reqopn_stn_req
APPN: -- ncl.ncl_unmap_dlc_type() -- mapping NCL(3) to CLS(3)
APPN: -- ncl.ncl_ls_fsm() -- send anr_tbl_update to owning cswncsnd
APPN: -- cswncsnd.main() -- received ANR_TBL_UPDATE ips.
APPN: -- cswncsnd.apply_anr_table_update() -- ANR:8002
APPN: -- cswncsnd.main() -- received ANR_TBL_UPDATE ips.
APPN: -- cswncsnd.apply_anr_table_update() -- ANR:8002
APPN: -- cswncsnd.main() -- received LSA_IPS ips.
APPN: -- ncl.ncl_ls_fsm -- FSM Invoked: Input:1, State:1->2, Action:1
APPN: -- ncl.ncl_ls_fsm -- P_CEP_ID:AAF638
APPN: -- ncl.ncl_ls_fsm() -- send anr_tbl_update to owning cswncsnd
APPN: -- cswncsnd.main() -- received ANR_TBL_UPDATE ips.
APPN: -- cswncsnd.apply_anr_table_update() -- ANR:8002
APPN: rtpm:  rtp_send() sent data over connection B9D5E8
APPN: hpr timer: rtt start time clocked at 135952 ms
APPN: -- cswncsnd.main() -- received NCL_SND_MSG ips.
APPN: -- cswncsnd.process_nlp_from_rtp() -- label: 8002, send to p_cep 00AAF638.
APPN: hpr timer: rtt end time clocked at 135972 ms
APPN: hpr timer: round trip time measured at 20 ms
```

Table 14 describes the significant fields shown in the display.

Table 14 debug appn hpr Field Descriptions

Field	Description
APPN	APPN debugging output.
NCL	Network control layer debugging output. Network control layer is the component that handles ANR packets.
ncl_port_fsm	Network control layer port finite state machine has been invoked.
ncl_assign_anr	ANR label has been assigned to an activating link station.
ncl_populate_anr	System is updating the ANR record with information specific to the link station.
ncl_ls_fsm	Network control layer link finite state machine has been invoked.
rtp_send	RTP is about to send a packet.
hpr timer	Debugging output related to an HPR timer.
rtt start time	RTP is measuring the round-rip time for an HPR status request packet. This is the start time.
NCL_SND_MSG	Network control layer has been requested to send a packet.
process_nlp_from_rtp	Network control layer has been requested by RTP to send a packet.
rtt end time	RTP is measuring the round trip time for an HPR status request packet. This is the time.
round trip time	Round-trip time for this HPR status exchange has been computed.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn ms

To display debugging information on APPN Management Services (MS) component activity, use the **debug appn ms** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn ms

no debug appn ms

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The MS component is responsible for generating, sending, and forwarding network management information in the form of traps and alerts to a network management focal point, such as Netview, in the APPN network.

Examples

The following is sample output from the **debug appn ms** command. In this example an error occurred that caused an alert to be generated.

```
Router# debug appn ms

APPN: ----- MSS00 ---- Deq ALERT_MSU msg
APPN: --- MSP70 --- ALERT MV FROM APPN WITH VALID LGTH
APPN: --- MSCPL --- Find Active FP
APPN: --- MSP30 --- Entering Build MS Transport
APPN: --- MSP31 --- Entering Building Routing Info.
APPN: --- MSP34 --- Entering Build GDS
APPN: --- MSP32 --- Entering Building UOW correlator
APPN: --- MSP34 --- Entering Build GDS
APPN: --- MSP30 --- Building GDS 0x1310
APPN: --- MSP30 --- Building MS Transport
APPN: --- MSP72 --- ACTIVE FP NOT FOUND, SAVE ONLY
APPN: --- MSUTL --- UOW <= 60, ALL COPIED in extract_uow
APPN: --- MSCAT --- by enq_cached_ms QUEUE SIZE OF QUEUE after enq 4
```

Table 15 describes the significant fields in the display.

Table 15 debug appn ms Field Descriptions

Field	Description
APPN	Indicates that this is APPN debugging output.
MSP	Indicates that this is MS component output.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn nof

To display debugging information on APPN Node Operator Facility (NOF) component activity, use the **debug appn nof** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn nof

no debug appn nof

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The NOF component is responsible for processing commands entered by the user such as start, stop, show, and configuration commands. NOF forwards these commands to the proper component and waits for the response.

Examples

The following is sample output from the **debug appn nof** command. In this example, an APPN connection network is being defined.

```
Router# debug appn nof

Turned on event 010000FF

Router# config term

Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# appn connection-network NETA.CISCO
Router(config-appn-cn)# port TR0
Router(config-appn-cn)# complete
router(config)#

APPN: ----- NOF ----- Define Connection Network Verb Received
APPN: ----- NOF ----- send define_cn_t ips to cs
APPN: ----- NOF ----- waiting for define_cn rsp from cs
router(config)#
```

Table 16 describes the significant fields in the display.

Table 16 debug appn nof Field Descriptions

Field	Description
APPN	APPN debugging output.
NOF	NOF component output.
Received	Configuration command was entered.
send	Message was sent to CS.
waiting	Response was expected from CS.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn pc

To display debugging information on APPN Path Control (PC) component activity, use the **debug appn pc** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn pc

no debug appn pc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The PC component is responsible for passing Message Units (MUs) between the Data Link Control (DLC) layer and other APPN components. PC implements transmission priority by passing higher priority MUs to the DLC before lower priority MUs.

Examples

The following is sample output from the **debug appn pc** command. In this example an MU is received from the network:

```
Router# debug appn pc

Turned on event 040000FF
APPN: ----- PC-----PC Deq REMOTE msg variant_name 2251
APPN: --PC-- mu received to PC lpid: A80AEC
APPN: --PC-- mu received from p_cep_id: 67C6F8
APPN: ----- PC-----PC Deq LSA_IPS from DLC
APPN: --PCX dequeued a DATA.IND
APPN: --- PC processing DL_DATA.ind
APPN: --PC-- mu_error_checker with no error, calling frr
APPN: --PC-- calling frr for packet received on LFSID: 1 2 3
APPN: ----- PC-----PC is sending MU to SC A90396
APPN: ----- SC-----send mu: A90396, rpc: 0, nws: 7, rh.b1: 90
APPN: SC: Send mu.snf: 8, th.b0: 2E, rh.b1: 90, dcf: 8
```

Table 17 describes the significant fields in the display.

Table 17 debug appn pc Field Descriptions

Field	Description
APPN	APPN debugging output.
PC	PC component output.
Deq REMOTE	Message was received from the network.
mu received	Message is an MU.
DATA.IND	MU contains data.
sending MU	MU is session traffic for an ISR session. The MU is forwarded to the Session Connector component for routing.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn ps

To display debugging information on APPN Presentation Services (PS) component activity, use the **debug appn ps** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn ps

no debug appn ps

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The PS component is responsible for managing the Transaction Programs (TPs) used by APPN. TPs are used for sending and receiving searches, receiving resource registration, and sending and receiving topology updates.

Examples

The following is sample output from the **debug appn ps** command. In this example a CP capabilities exchange is in progress.

```
Router# debug appn ps

Turned on event 200000FF
APPN: ---- CCA --- CP_CAPABILITIES_TP has started
APPN: ---- CCA --- About to wait for Partner to send CP_CAP
APPN: ---- CCA --- Partner LU name: NETA.PATTY
APPN: ---- CCA --- Mode Name: CPSVCMG
APPN: ---- CCA --- CGID: 78
APPN: ---- CCA --- About to send cp_cp_session_act to SS
APPN: ---- CCA --- Waiting for cp_cp_session_act_rsp from SS
APPN: ---- CCA --- Received cp_cp_session_act_rsp from SS
APPN: ---- CCA --- About to send CP_CAP to partner
APPN: ---- CCA --- Send to partner completed with rc=0, 0
APPN: ---- RCA --- Allocating conversation
APPN: ---- RCA --- Sending CP_CAPABILITIES
APPN: ---- RCA --- Getting conversation attributes
APPN: ---- RCA --- Waiting for partner to send CP_CAPABILITIES
APPN: ---- RCA --- Normal processing complete with cgid = 82
APPN: ---- RCA --- Deallocating CP_Capabilities conversation
```

Table 18 describes the significant fields in the display.

Table 18 debug appn ps Field Descriptions

Field	Description
APPN	APPN debugging output.
CCA	CP Capabilities TP output.
RCA	Receive CP Capabilities TP output.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn scm

To display debugging information on APPN Session Connector Manager (SCM) component activity, use the **debug appn scm** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn scm

no debug appn scm

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The SCM component is responsible for the activation and deactivation of the local resources that route an intermediate session through the router.

Examples

The following is sample output from the **debug appn scm** command. In this example an intermediate session traffic is being routed.

```
Router# debug appn scm

Turned on event 020000FF
Router#
APPN: ----- SCM-----SCM Deq a MU
APPN: ----- SCM-----SCM send ISR_INIT to SSI
APPN: ----- SCM-----(i05) Enter compare_fqpcid()
APPN: ----- SCM-----Adding new session_info table entry. addr=A93160
APPN: ----- SCM-----SCM Deq ISR_CINIT message
APPN: ----- SCM-----(i05) Enter compare_fqpcid()
APPN: ----- SCM-----SCM sends ASSIGN_LFSID to ASM
APPN: ----- SCM-----SCM Rcvd sync ASSIGN_LFSID from ASM
APPN: ----- SCM-----SCM PQenq a MU to ASM
APPN: ----- SCM-----SCM Deq a MU
APPN: ----- SCM-----(i05) Enter compare_fqpcid()
APPN: ----- SCM-----SCM PQenq BIND rsp to ASM
```

Table 19 describes the significant fields in the display.

Table 19 debug appn scm Field Descriptions

Field	Description
APPN	APPN debugging output.
SCM	SCM component output.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn ss

To display session services (SS) events, use the **debug appn ss** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn ss

no debug appn ss

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The SS component generates unique session identifiers, activates and deactivates control point-to-control point (CP-CP) sessions, and assists LUs in initiating and activating LU-LU sessions.

Examples

The following is sample output from the **debug appn ss** command. In this example CP-CP sessions between the router and another node are being activated.

```
Router# debug appn ss

Turned on event 100000FF
APPN: ----- SS ----- Deq ADJACENT_CP_CONTACTED message
APPN: ----- SS ----- Deq SESSST_SIGNAL message
APPN: ----- SS ----- Deq CP_CP_SESSION_ACT message
APPN: Sending ADJACENT_NN_1015 to SCM, adj_node_p=A6B980,cp_name=NETA.PATTY
APPN: ----- SS ----- Sending REQUEST_LAST_FRSN message to TRS
APPN: ----- SS ----- Receiving REQUEST_LAST_FRSN_RSP from TRS
APPN: ----- SS ----- Sending ACTIVE CP_STATUS CONLOSER message to DS
APPN: ----- SS ----- Sending ACTIVE CP_STATUS CONLOSER message to MS
APPN: ----- SS ----- Sending ACTIVE CP_STATUS CONLOSER message to TRS
APPN: ----- SS ----- Sending CP_CP_SESSION_ACT_RSP message to CCA TP
APPN: ----- SS ----- Sending PENDING_ACTIVE CP_STATUS CONWINNER message to DS
APPN: ----- SS ----- Sending REQUEST_LAST_FRSN message to TRS
APPN: ----- SS ----- Receiving REQUEST_LAST_FRSN_RSP from TRS
APPN: ----- SS ----- Sending ACT_CP_CP_SESSION message to RCA TP
APPN: ----- SS ----- Deq ASSIGN_PCID message
APPN: ----- SS ----- Sending ASSIGN_PCID_RSP message to someone
APPN: ----- SS ----- Deq INIT_SIGNAL message
APPN: ----- SS ----- Sending REQUEST_COS_TPF_VECTOR message to TRS
APPN: ----- SS ----- Receiving an REQUEST_COS_TPF_VECTOR_RSP from TRS
APPN: ----- SS ----- Sending REQUEST_SINGLE_HOP_ROUTE message to TRS
APPN: ----- SS ----- Receiving an REQUEST_SINGLE_HOP_ROUTE_RSP from TRS
APPN: ----- SS ----- Sending ACTIVATE_ROUTE message to CS
APPN: ----- SS ----- Deq ACTIVATE_ROUTE_RSP message
APPN: ----- SS ----- Sending CINIT_SIGNAL message to SM
APPN: ----- SS ----- Deq ACT_CP_CP_SESSION_RSP message
APPN: -- SS----SS ssp00, act_cp_cp_session_rsp received, sense_code=0, cgid=5C,
ips@=A93790
APPN: Sending ADJACENT_NN_1015 to SCM, adj_node_p=A6B980,cp_name=18s
APPN: ----- SS ----- Sending ACTIVE CP_STATUS CONWINNER message to DS
APPN: ----- SS ----- Sending ACTIVE CP_STATUS CONWINNER message to MS
APPN: ----- SS ----- Sending ACTIVE CP_STATUS CONWINNER message to TRS
```

Table 20 describes the significant fields in the display.

Table 20 debug appn ss Field Descriptions

Field	Description
APPN	APPN debugging output.
SS	SS component output.

Related Commands

Command	Description
debug appn all	Turns on all possible debugging messages for APPN.

debug appn trs

To display debugging information on APPN Topology and Routing Services (TRS) component activity, use the **debug appn trs** privileged EXEC command. The **no** form of this command disables debugging output.

debug appn trs

no debug appn trs

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The TRS component is responsible for creating and maintaining the topology database, creating and maintaining the class of service database, and computing and caching optimal routes through the network.

Examples

The following is sample output from the **debug appn trs** command:

```
Router# debug appn trs

Turned on event 400000FF
APPN: ----- TRS ----- Received a QUERY_CPNAME
APPN: ----- TRS ----- Received a REQUEST_ROUTE
APPN: ----- TRS ----- check_node node_name=NETA.LISA
APPN: ----- TRS ----- check_node node_index=0
APPN: ----- TRS ----- check_node node_weight=60
APPN: ----- TRS ----- add index 484 to origin description list
APPN: ----- TRS ----- add index 0 to dest description list
APPN: ----- TRS ----- origin tg_vector is NULL
APPN: ----- TRS ----- weight_to_origin = 0
APPN: ----- TRS ----- weight_to_dest = 0
APPN: ----- TRS ----- u_b_s_f weight = 30
APPN: ----- TRS ----- u_b_s_f prev_weight = 2147483647
APPN: ----- TRS ----- u_b_s_f origin_index = 484
APPN: ----- TRS ----- u_b_s_f dest_index = 0
APPN: ----- TRS ----- b_r_s_f weight = 30
APPN: ----- TRS ----- b_r_s_f origin_index = 484
APPN: ----- TRS ----- b_r_s_f dest_index = 0
APPN: ----- TRS ----- Received a REQUEST_ROUTE
APPN: ----- TRS ----- check_node node_name=NETA.LISA
APPN: ----- TRS ----- check_node node_index=0
APPN: ----- TRS ----- check_node node_weight=60
APPN: ----- TRS ----- check_node node_name=NETA.BART
APPN: ----- TRS ----- check_node node_index=484
APPN: ----- TRS ----- check_node node_weight=60
APPN: ----- TRS ----- add index 484 to origin description list
APPN: ----- TRS ----- add index 0 to dest description list
APPN: ----- TRS ----- origin_tg_weight to non-VN=30
APPN: ----- TRS ----- origin_node_weight to non-VN=60
APPN: ----- TRS ----- weight_to_origin = 90
APPN: ----- TRS ----- weight_to_dest = 0
APPN: ----- TRS ----- u_b_s_f weight = 120
APPN: ----- TRS ----- u_b_s_f prev_weight = 2147483647
APPN: ----- TRS ----- u_b_s_f origin_index = 484
APPN: ----- TRS ----- u_b_s_f dest_index = 0
```

```
APPN: ----- TRS ----- b_r_s_f weight = 120
APPN: ----- TRS ----- b_r_s_f origin_index = 484
APPN: ----- TRS ----- b_r_s_f dest_index = 0
```

Table 21 describes the significant fields in the display.

Table 21 debug appn trs Field Descriptions

Field	Description
APPN	APPN debugging output.
TRS	TRS component output.

debug arap

To display AppleTalk Remote Access Protocol (ARAP) events, use the **debug arap** privileged EXEC command. The **no** form of this command disables debugging output.

debug arap {**internal** | **memory** | **mnp4** | **v42bis**} [*linenum* [**aux** | **console** | **tty** | **vty**]]

no debug arap {**internal** | **memory** | **mnp4** | **v42bis**} [*linenum* [**aux** | **console** | **tty** | **vty**]]

Syntax Description

internal	Debugs internal ARA packets.
memory	Debugs memory allocation for ARA.
mnp4	Debugs low-level asynchronous serial protocol.
v42bis	Debugs V.42*bis* compression.
linenum	(Optional) Line number. The number ranges from 0 to 999, depending on what type of line is selected.
aux	(Optional) Auxiliary line.
console	(Optional) Primary terminal line.
tty	(Optional) Physical terminal asynchronous line.
vty	(Optional) Virtual terminal line.

Usage Guidelines

Use the **debug arap** command with the **debug callback** command on access servers to debug dialin and callback events.

Use the **debug modem** command to help catch problems related to ARAP autodetection (that is, **autoselect arap**). These problems are very common and are most often caused by modems, which are the most common cause of failure in ARAP connection and configuration sessions.

Examples

The following is sample output from the **debug arap internal** command:

```
Router# debug arap internal

ARAP: ---------- SRVRVERSION ----------
ARAP: ---------- ACKing 0 ----------
ARAP: ---------- AUTH_CHALLENGE ----------
arapsec_local_account setting up callback
ARAP: ---------- ACKing 1 ----------
ARAP: ---------- AUTH_RESPONSE ----------
arap_startup initiating callback ARAP 2.0
ARAP: ---------- CALLBACK ----------
TTY7 Callback process initiated, user: dialback dialstring 40
TTY7 Callback forced wait = 4 seconds
TTY7 ARAP Callback Successful - await exec/autoselect pickup
TTY7: Callback in effect
ARAP: ---------- STARTINFOFROMSERVER ----------
ARAP: ---------- ACKing 0 ----------
ARAP: ---------- ZONELISTINFO ----------
ARAP: ---------- ZONELISTINFO ----------
ARAP: ---------- ZONELISTINFO ----------
ARAP: ---------- ZONELISTINFO ----------
ARAP: ---------- ZONELISTINFO ----------
```

Related Commands

Command	Description
debug callback	Displays callback events when the router is using a modem and a chat script to call back on a terminal line.
debug modem	Observes modem line activity on an access server.

debug arp

To display information on Address Resolution Protocol (ARP) transactions, use the **debug arp** privileged EXEC command. The **no** form of this command disables debugging output.

debug arp

no debug arp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command when some nodes on a TCP/IP network are responding, but others are not. It shows whether the router is sending ARP packets and whether it is receiving ARP packets.

Examples

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

```
Router# debug arp

IP ARP: sent req src 172.16.22.7 0000.0c01.e117, dst 172.16.22.96 0000.0000.0000
IP ARP: rcvd rep src 172.16.22.96 0800.2010.b908, dst 172.16.22.7
IP ARP: rcvd req src 172.16.6.10 0000.0c00.6fa2, dst 172.16.6.62
IP ARP: rep filtered src 172.16.22.7 aa92.1b36.a456, dst 255.255.255.255 ffff.ffff.ffff
IP ARP: rep filtered src 172.16.9.7 0000.0c00.6b31, dst 172.16.22.7 0800.2010.b908
```

In the output, each line of output represents an ARP packet that the router sent or received. Explanations for the individual lines of output follow.

The first line indicates that the router at IP address 172.16.22.7 and MAC address 0000.0c01.e117 sent an ARP request for the MAC address of the host at 172.16.22.96. The series of zeros (0000.0000.0000) following this address indicate that the router is currently unaware of the MAC address.

```
IP ARP: sent req src 172.16.22.7 0000.0c01.e117, dst 172.16.22.96 0000.0000.0000
```

The second line indicates that the router at IP address 172.16.22.7 receives a reply from the host at 172.16.22.96 indicating that its MAC address is 0800.2010.b908:

```
IP ARP: rcvd rep src 172.16.22.96 0800.2010.b908, dst 172.16.22.7
```

The third line indicates that the router receives an ARP request from the host at 172.16.6.10 requesting the MAC address for the host at 172.16.6.62:

```
IP ARP: rcvd req src 172.16.6.10 0000.0c00.6fa2, dst 172.16.6.62
```

The fourth line indicates that another host on the network attempted to send the router an ARP reply for its own address. The router ignores meaningless replies. Usually, meaningless replies happen if a bridge is being run in parallel with the router and is allowing ARP to be bridged. This condition indicates a network misconfiguration.

```
IP ARP: rep filtered src 172.16.22.7 aa92.1b36.a456, dst 255.255.255.255 ffff.ffff.ffff
```

The fifth line indicates that another host on the network attempted to inform the router that it is on network 172.16.9.7, but the router does not know that the network is attached to a different router interface. The remote host (probably a PC or an X terminal) is misconfigured. If the router were to install this entry, it would deny service to the real machine on the proper cable.

```
IP ARP: rep filtered src 172.16.9.7 0000.0c00.6b31, dst 172.16.22.7 0800.2010.b908
```

debug asp packet

To display information on all asynchronous security protocols operating on the router, use the **debug asp packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug asp packet

no debug asp packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The router uses asynchronous security protocols from companies including ADT Security Systems, Inc., Adplex, and Diebold to transport alarm blocks between two devices (such as a security alarm system console and an alarm panel). The alarm blocks are transported in pass-through mode using BSTUN encapsulation.

Examples

The following is partial sample output from the **debug asp packet** command for asynchronous security protocols when packet debugging is enabled on an asynchronous line carrying Diebold alarm traffic. In this example, two polls are sent from the Diebold alarm console to two alarm panels that are multidropped from a single EIA/TIA RS-232 interface. The alarm panels have device addresses F0 and F1. The example trace indicates that F1 is responding and F0 is not responding. At this point, you need to examine the physical link and possibly use a datascope to determine why the device is not responding.

```
Router# debug asp packet

12:19:48: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FF4C42
12:19:49: ASP: Serial5: ADI-Tx: Data (1 bytes): 88
12:19:49: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FF9B94
12:20:47: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FF757B
12:20:48: ASP: Serial5: ADI-Tx: Data (1 bytes): F3
12:20:48: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FFB1BE
12:21:46: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FFE6E8
12:21:46: ASP: Serial5: ADI-Tx: Data (1 bytes): 6F
12:21:46: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FFC1CE
```

Table 22 describes the significant fields in the display.

Table 22 debug asp Packet Descriptions

Field	Description
ASP	Asyncronous security protocol packet.
Serial5	Interface receiving and sending the packet.
ADI-Rx	Packet is being received.
ADI-T	Packet is being sent.
Data (*n* bytes)	Type and size of the packet.
F1FF4c42	Alarm panel device address.

debug async async-queue

To display debug messages for asynchronous rotary line queueing, use the **debug async async-queue** command in privileged EXEC mode.

debug async async-queue

Syntax Description

This command has no arguments or keywords.

Defaults

This command has no default settings.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following example starts the asynchronous rotary line queueing debugging display:

```
Router# debug async async-queue

*Mar  2 03:50:28.377: AsyncQ: First connection to be queued - starting the AsyncQ manager
*Mar  2 03:50:28.377: AsyncQ: Enabling the AsyncQ manager
*Mar  2 03:50:28.377: AsyncQ: Started the AsyncQ manager process with pid 98
*Mar  2 03:50:28.381: AsyncQ: Created a Waiting TTY on TTY66 with pid 99
*Mar  2 03:50:30.164: WaitingTTY66: Did Authentication on waiting TTY (VTY)
*Mar  2 03:50:30.168: AsyncQ: Received ASYNCQ_MSG_ADD
*Mar  2 03:50:30.168: AsyncQ: New queue, adding this connection as the first element
*Mar  2 03:50:34.920: AsyncQ: Created a Waiting TTY on TTY67 with pid 100
*Mar  2 03:50:36.783: WaitingTTY67: Did Authentication on waiting TTY (VTY)
*Mar  2 03:50:36.787: AsyncQ: Received ASYNCQ_MSG_ADD
*Mar  2 03:50:36.787: AsyncQ: Queue exists, adding this connection to the end of the queue
```

Related Commands

Command	Description
debug ip tcp transactions	Enables the IP TCP transactions debugging display to observe significant transactions such as state changes, retransmissions, and duplicate packets.
debug modem	Enables the modem debugging display to observe modem line activity on an access server.

debug backhaul-session-manager set

To trace state changes and receive messages and events for all the available session sets or a specified session set, use the **debug backhaul-session-manager set** privileged EXEC command.

debug backhaul-session-manager set {**all** | **name** *set-name*}

Syntax Description

all	All available session sets.
name *set-name*	Specified session set.

Defaults

Debugging for backhaul session sets is not enabled.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following is output for the **debug backhaul-session-manager set all** command:

```
Router# debug backhaul-session-manager set all
Router# debug_bsm_command:DEBUG_BSM_SET_ALL

 Function set_proc_event() is called
Session-Set :test-set
Old State   :BSM_SET_OOS
New State   :BSM_SET_OOS
   Active-Grp  :NONE
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-None
   Event rcvd  :EVT_GRP_INS

BSM:Event BSM_SET_UP is sent to user
Session-Set :test-set
Old State   :BSM_SET_OOS
New State   :BSM_SET_ACTIVE_IS
   Active-Grp  :g-11
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-Active
   Event rcvd  :BSM_ACTIVE_TYPE
```

The following is output for the **debug backhaul-session-manager set all name test-set** command:

```
Router# debug backhaul-session-manager set name test-set
Router# debug_bsm_command:DEBUG_BSM_SET_NAME

Nomad-B# Function set_proc_event() is called
Session-Set :test-set
Old State   :BSM_SET_OOS
```

```
New State    :BSM_SET_OOS
   Active-Grp  :NONE
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-None
   Event rcvd  :EVT_GRP_INS

Nomad-B#BSM:Event BSM_SET_UP is sent to user
Session-Set :test-set
Old State    :BSM_SET_OOS
New State    :BSM_SET_ACTIVE_IS
   Active-Grp  :g-11
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-Active
   Event rcvd  :BSM_ACTIVE_TYPE
```

Related Commands

Command	Description
debug backhaul-session-manager session	Displays debug information for all available sessions or a specific session.

debug backhaul-session-manager session

To debug all the available sessions or a specified session, use the **debug backhaul-session-manager session** privileged EXEC command.

debug backhaul-session-manager session {**show** | **state** | **xport**} {**all** | *session-id*}

Syntax Description

show	Displays session manager states and statistics. **Note** This command only displays information about the specified session once, and does not enable debugging.
state	Shows information about state transitions. Possible states are: • SESS_SET_IDLE: A session-set has been created. • SESS_SET_OOS: Sessions have been added to session groups. No ACTIVE notification has been received from VSC. • SESS_SET_ACTIVE_IS: An ACTIVE notification has been received over one in-service session group. STANDBY notification has not been received on any available session groups. • SESS_SET_STNDBY_IS: A STANDBY notification is received, but there is no in-service active session group available. • SESS_SET_FULL_IS: A session group in-service has ACTIVE notification and at least one session group in-service has STANDBY notification. • SESS_SET_SWITCH_OVER: An ACTIVE notification is received on a session group in-service, which had received STANDBY notification.
xport	Provides traces for all PDUs (packets), application PDUs, and session manager messages. **Note** Use caution while enabling this debug command in a live system.
all	All available session sets.
session-id	Specified session.

Defaults

Debugging for backhaul session-session is not enabled.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following is output for the **debug backhaul-session-manager session all** command:

```
Router# debug backhaul-session-manager session show all

Router# debug_bsm_command:DEBUG_BSM_SESSION_SHOW

23:43:34:Session information --
  Group:g-11
Configuration:
     Local:172.18.72.198  , port:5555
    Remote:161.44.2.72    , port:5555
  Id:33,  Priority:1
  RUDP Option:Client, Conn Id:0x80BA14EC
State:
  Status:OPEN, Use-status:IS,
Statistics:
  # of resets:68
  Receive Total pkts:7, failures:0
  Transmit Total pkts:69, failures:0, blocked:0
  group-ptr:0x80B17E18, tmrid:0x8094D658, debug-mask:0x0

23:43:34:Session information --
  Group:g-12
Configuration:
     Local:172.18.72.198  , port:5575
    Remote:161.44.2.72    , port:5575
  Id:34,  Priority:1
  RUDP Option:Client, Conn Id:0x80BA12FC
State:
  Status:OPEN_WAIT, Use-status:OOS,
Statistics:
  # of resets:88
  Receive Total pkts:8, failures:0
  Transmit Total pkts:88, failures:0, blocked:0
  group-ptr:0x80B17ED0, tmrid:0x8094D678, debug-mask:0x0

Router# debug backhaul-session-manager session show 33

Router# debug_bsm_command:DEBUG_BSM_SESSION_SHOW

23:48:32:Session information --
  Group:g-11
Configuration:
     Local:172.18.72.198  , port:5555
    Remote:161.44.2.72    , port:5555
  Id:33,  Priority:1
  RUDP Option:Client, Conn Id:0x80BA14EC
State:
  Status:OPEN, Use-status:IS,
Statistics:
  # of resets:68
  Receive Total pkts:7, failures:0
  Transmit Total pkts:69, failures:0, blocked:0
  group-ptr:0x80B17E18, tmrid:0x8094D658, debug-mask:0x0

Router# debug backhaul-session-manager session all

Router# debug_bsm_command:DEBUG_BSM_SESSION_ALL
```

```
23:49:14:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:14:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:14:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:14:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:14:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:19:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:19:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:19:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:19:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:19:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:24:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:24:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:24:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:24:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:24:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:29:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:29:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:29:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:29:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:29:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:34:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:34:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:34:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:34:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:34:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:34:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 1 (CONN-FAILED)

23:49:34:SESSION:STATE:(33) old-state:OPEN, new-state:CLOSE_WAIT


Router# debug backhaul-session-manager session state all

Router# debug_bsm_command:DEBUG_BSM_SESSION_STATE_ALL

23:50:54:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:50:54:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:50:54:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:50:54:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS
```

```
Router# debug backhaul-session-manager session xport all

Router# debug_bsm_command:DEBUG_BSM_SESSION_XPORT
23:51:39:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:51:42:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 5 (CONN-RESET)

23:51:44:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
```

Related Commands

Command	Description
debug backhaul-session-manager set	Traces state changes and receives messages and events for all available session sets or a specified session set.

debug bert

To display information on the bit error rate testing (BERT) feature, use the **debug bert** privileged EXEC command. The **no** form of this command disables the debugging output.

debug bert

no debug bert

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(2)XD	This command was introduced.

Usage Guidelines

The **debug bert** command output is used primarily by Cisco technical support representatives. The **debug bert** command displays debugging messages for specific areas of executed code.

Examples

The following is output from the **debug bert** command:

```
Router# debug bert

Bit Error Rate Testing debugging is on

Router# no debug bert

Bit Error Rate Testing debugging is off
```

Related Commands

Command	Description
bert abort	Aborts a bit error rate testing session.
bert controller	Starts a bit error rate test for a particular port on a Cisco AS5300 router.
bert profile	Sets up various bit error rate testing profiles.

debug bri-interface

To display debugging information on ISDN BRI routing activity, use the **debug bri-interface** privileged EXEC command. The **no** form of this command disables debugging output.

debug bri-interface

no debug bri-interface

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug bri-interface** command indicates whether the ISDN code is enabling and disabling the B channels when attempting an outgoing call. This command is available for the low-end router products that have a multi-BRI network interface module installed.

Caution Because the **debug bri-interface** command generates a substantial amount of output, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Examples

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

```
Router# debug bri-interface

BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: write_sid: wrote 15 for subunit 0, slot 1.
BRI: write_sid: wrote 17 for subunit 0, slot 1.
BRI: write_sid: wrote 6 for subunit 0, slot 1.
BRI: write_sid: wrote 8 for subunit 0, slot 1.
BRI: write_sid: wrote 11 for subunit 0, slot 1.
BRI: write_sid: wrote 13 for subunit 0, slot 1.
BRI: write_sid: wrote 29 for subunit 0, slot 1.
BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: write_sid: wrote 15 for subunit 0, slot 1.
BRI: write_sid: wrote 17 for subunit 0, slot 1.
BRI: write_sid: wrote 20 for subunit 0, slot 1.
BRI: Starting Power Up timer for unit = 0.
BRI: write_sid: wrote 3 for subunit 0, slot 1.
BRI: Starting T3 timer after expiry of PUP timeout for unit = 0, current state is F4.
BRI: write_sid: wrote FF for subunit 0, slot 1.
BRI: Activation for unit = 0, current state is F7.
BRI: enable channel B1
BRI: write_sid: wrote 14 for subunit 0, slot 1.

%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to up.!!!
BRI: disable channel B1
BRI: write_sid: wrote 15 for subunit 0, slot 1.

%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to down
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to down
```

The following line indicates that an internal command was written to the interface controller. The subunit identifies the first interface in the slot.

```
BRI: write_sid: wrote 1B for subunit 0, slot 1.
```

The following line indicates that the power-up timer was started for the named unit:

```
BRI: Starting Power Up timer for unit = 0.
```

The following lines indicate that the channel or the protocol on the interface changed state:

```
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to up.!!!
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to down
```

The following line indicates that the channel was disabled:

```
BRI: disable channel B1
```

Lines of output not described are for use by support staff only.

Related Commands

Command	Description
debug isdn event	Displays ISDN events occurring on the user side (on the router) of the ISDN interface.
debug isdn q921	Displays data link-layer (Layer 2) access procedures that are taking place at the router on the D channel (LSPD).
debug isdn q931	Displays information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network.

debug bsc event

To display all events occurring in the Binary Synchronous Communications (Bisync) feature, use the **debug bsc event** privileged EXEC command. The **no** form of this command disables debugging output.

debug bsc event [*number*]

no debug bsc event [*number*]

Syntax Description

number	(Optional) Group number.

Usage Guidelines

This command traces all interfaces configured with a **bsc protocol-group** *number* command.

Examples

The following is sample output from the **debug bsc event** command:

```
Router# debug bsc event

BSC: Serial2         POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2         POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2         POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
0:04:32: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2         POLLEE-FSM inp:E_RxEtx old_st:CU_Down new_st:TCU_EOFile
0:04:32: BSC: Serial2 :SDI-rx: 5 bytes
BSC: Serial2         POLLEE-FSM inp:E_RxEnq old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2         POLLEE-FSM inp:E_Timeout old_st:CU_Down new_st:TCU_InFile
BSC: Serial2         POLLEE-FSM inp:E_Timeout old_st:CU_Idle new_st:TCU_InFile
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2, changed state to up
%LINK-3-UPDOWN: Interface Serial2, changed state to up
BSC: Serial2         POLLEE-FSM inp:E_Timeout old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2         POLLEE-FSM inp:E_RxEtx old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :SDI-rx: 5 bytes
BSC: Serial2         POLLEE-FSM inp:E_RxEnq old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :NDI-rx: 3 bytes
```

Related Commands

Command	Description
debug bsc packet	Displays all frames traveling through the Bisync feature.
debug bstun events	Displays BSTUN connection events and status.

debug bsc packet

To display all frames traveling through the Binary Synchronous Communications (Bisync) feature, use the **debug bsc packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug bsc packet [**group** *number*] [**buffer-size** *bytes*]

no debug bsc packet [**group** *number*] [**buffer-size** *bytes*]

Syntax Description

group *number*	(Optional) Group number.
buffer-size *bytes*	(Optional) Number of bytes displayed per packet (defaults to 20).

Defaults

The default number of bytes displayed is 20.

Usage Guidelines

This command traces all interfaces configured with a **bsc protocol-group** *number* command.

Examples

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

```
Router# debug bsc packet

0:23:33: BSC: Serial2     :NDI-rx : 27 bytes 401A400227F5C31140C11D60C8C5D3D3D51D4013
0:23:33: BSC: Serial2     :SDI-tx : 12 bytes 00323237FF3232606040402D
0:23:33: BSC: Serial2     :SDI-rx : 2 bytes 1070
0:23:33: BSC: Serial2     :SDI-tx : 27 bytes 401A400227F5C31140C11D60C8C5D3D3D51D4013
0:23:33: BSC: Serial2     :SDI-rx : 2 bytes 1061
0:23:33: BSC: Serial2     :SDI-tx : 5 bytes 00323237FF
```

Related Commands

Command	Description
debug bsc event	Displays all events occurring in the Bisync feature.
debug bstun events	Displays BSTUN connection events and status.

debug bstun events

To display BSTUN connection events and status, use the **debug bstun events** privileged EXEC command. The **no** form of this command disables debugging output.

debug bstun events [*number*]

no debug bstun events [*number*]

Syntax Description

number	(Optional) Group number.

Usage Guidelines

When you enable the **debug bstun events** command, messages showing connection establishment and other overall status messages are displayed.

You can use the **debug bstun events** command to assist you in determining whether the BSTUN peers are configured correctly and are communicating. For example, if you enable the **debug bstun packet** command and you do not see any packets, you may want to enable event debugging.

Note Also refer to the **debug bsc packet** and **debug bsc event** commands. Currently, these two commands support the only protocol working through the BSTUN tunnel. Sometimes frames do not go through the tunnel because they have been discarded at the Bisync protocol level.

Examples

The following is sample output from the **debug bstun events** command of keepalive messages working correctly. If the routers are configured correctly, at least one router will show reply messages.

```
Router# debug bstun packet

BSTUN: Received Version Reply opcode from (all[2])_172.16.12.2/1976 at 1360
BSTUN: Received Version Request opcode from (all[2])_172.16.12.2/1976 at 1379
BSTUN: Received Version Reply opcode from (all[2])_172.16.12.2/1976 at 1390
```

Note In a scenario where there is constantly loaded bi-directional traffic, you might not see keepalive messages because they are sent only when the remote end has been silent for the keepalive period.

The following is sample output from the **debug bstun events** output of an event trace in which the wrong TCP address has been specified for the remote peer. These are non-keepalive related messages.

```
Router# debug bstun packet

BSTUN: Change state for peer (C1[1])172.16.12.22/1976 (closed->opening)
BSTUN: Change state for peer (C1[1])172.16.12.22/1976 (opening->open wait)
%BSTUN-6-OPENING: CONN: opening peer (C1[1])172.16.12.22/1976, 3
BSTUN: tcpd sender in wrong state, dropping packet
BSTUN: tcpd sender in wrong state, dropping packet
BSTUN: tcpd sender in wrong state, dropping packet
```

Related Commands

Command	Description
debug bsc event	Displays all events occurring in the Bisync feature.
debug bsc packet	Displays all frames traveling through the Bisync feature.
debug bundle errors	Displays packet information on packets traveling through the BSTUN links.

debug bundle errors

To enable the display of information on bundle errors, use the **debug bundle errors** privileged EXEC command.

debug bundle errors

no debug bundle errors

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

Use this command to enable the display of error information for a bundle, such as reports of inconsistent mapping in the bundle.

Related Commands

Command	Description
bump	Configures the bumping rules for a VC class that can be assigned to a VC bundle.
bundle	Creates a bundle or modifies an existing bundle to enter bundle configuration mode.
debug bundle events	Enables display of bundle events when use occurs.

debug bundle events

To enable display of bundle events when use occurs, use the **debug bundle events** privileged EXEC command in debug mode.

debug bundle events

no debug bundle events

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

Use this command to enable the display of bundle events, such as occurrences of VC bumping, when bundles were brought up, when they were taken down, and so forth.

Related Commands

Command	Description
debug bundle errors	Enables the display of information on bundle errors.

debug bstun packet

To display packet information on packets traveling through the BSTUN links, use the **debug bstun packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug bstun packet [**group** *number*] [**buffer-size** *bytes*]

no debug bstun packet [**group** *number*] [**buffer-size** *bytes*]

Syntax Description

group *number*	(Optional) BSTUN group number.
buffer-size *bytes*	(Optional) Number of bytes displayed per packet (defaults to 20).

Defaults

The default number of bytes displayed is 20.

Examples

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

```
Router# debug bstun packet

BSTUN bsc-local-ack: 0:00:00 Serial2         SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:00 Serial2         SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:06 Serial2         NDI: Addr: 40 Data: 0227F5C31140C11D60C8
```

Related Commands

Command	Description
debug bstun events	Displays BSTUN connection events and status.

debug cable env

To display information about the Cisco uBR7246 universal broadband router physical environment, including internal temperature, midplane voltages, fan performance, and power supply voltages, use the **debug cable env** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable env

no debug cable env

Syntax Description

This command has no arguments or keywords.

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 uBR7246 console.

Examples

The following is sample output from the **debug cable env** command:

```
Router# debug cable env

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

Table 23 describes the significant fields in the display.

Table 23 debug cable env Field Descriptions

Field	Description
ps id	Power supply raw voltage reading.
pstype	Power supply type determined from the ps id, v, and r values. The Cisco uBR7246 universal broadband router contains dual power supplies, so 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.

debug cable err

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

debug cable err

no debug cable err

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is used to display unexpected DOCSIS MAC protocol messages. When the Cisco uBR7246 universal broadband router does not to expect to receive a specific MAC message, an error message and hexadecimal dump are printed. Other miscellaneous error conditions may result in output.

Examples

The following is sample output from the **debug cable err** command:

```
Router# debug cable err

This is a UCD Message
This is a MAP Message
This is a RNG_RSP Message
This is a REG_RSP Message
This is a UCC_REQ Message
This is a BPKM_RSP Message
This is a TRI_TCD Message
This is a TRI_TSI Message
This is a unrecognized MCNS message

ERROR:######TICKS PER MSLOT NOT POWER OF 2####
```

debug cable freqhop

To display debug messages for frequency hopping, use the **debug cable freqhop** privileged EXEC command. 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.

Defaults

Debugging for frequency hopping is not enabled.

Command History

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

Examples

The following is sample output from the **debug cable freqhop** command:

```
Router# debug cable freqhop

CMTS freqhop debugging is on
```

Related Commands

Command	Description
debug cable hw-spectrum	Displays debug information about spectrum management (frequency agility).
debug cable freqhop	Displays debug information about frequency hopping, which is a facet of spectrum management.

debug cable hw-spectrum

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

debug cable hw-spectrum

no debug cable hw-spectrum

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for spectrum management is not enabled.

Command History

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

Examples

The following is sample output for the **debug cable hw-spectrum** command:

```
Router# debug cable hw-spectrum

CMTS specmgmt debugging is on
```

debug cable interface

To perform debugging on a specified interface, use the **debug cable interface** privileged EXEC command. To turn off debugging on a specified interface, use the **no** form of this command.

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

no debug cable interface *interface* **mac-address** *address*

Syntax Description

interface	Specifies the cable interface to be debugged.
mac-address	(Optional) Specifies that debugging is to be done on a specified MAC address.
address	(Optional) Specifies the MAC address of the interface.
mask	(Optional) Specifies the MAC address validation address.
verbose	(Optional) Displays detailed debug information.

Command History

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

Usage Guidelines

You can repeat this debug command for other interfaces. Each time you specify a different cable interface or MAC address, debugging is turned on for this cable interface or MAC address.

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

Examples

The following example demonstrates how to enable debugging on interface c3/0:

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

The following example demonstrates how to enable detailed debugging on interface c3/0:

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

The following example demonstrates how to enable debugging on interface c3/0 for all traffic coming from modems with MAC addresses 0010.00xx.xxxx:

```
Router# debug cable interface c3/0 mac-address 0010.0000.0000 ffff.ff00.0000
```

Related Commands

Command	Description
debug cable mac-address	Enables debugging on traffic from modems with the specified MAC address or MAC address range.

debug cable keyman

To activate debugging of TEK and KEK baseline privacy key activity, use the **debug cable keyman** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable keyman

no debug cable keyman

Syntax Description

This command has no arguments or keywords.

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 will be displayed on the Cisco uBR7246 console. This command is used to display encryption key management debugging output.

Examples

The following is sample output from the **debug cable keyman** command:

```
Router# debug cable keyman

Read Verify DES failed with SID %2x
     Verify key failed with SID %2x : setvalue = %llx, readback = %llx
     Verify iv failed with SID %2x : setvalue = %llx, readback = %llx
Next TEK lifetime check is set to %u seconds.
     Next Multicast TEK lifetime check is set to 1 seconds

[UCAST_TEK] :", idbp->hw_namestring);
     show_sid_key_chain(ds, &ds->mcast_sid_key_list_hdr);

[MCAST_TEK] :", idbp->hw_namestring);
     buginf("\nSID : %4x\t", sidkey->sid);
     buginf("seq : %2x\t current : %2x\n", sidkey->key_seq_num,
          sidkey->current_key_num);
     buginf(" Status[0] : %x\tDES IV[0]  : %llx\tKey Life[0]: %u sec\n",
          sidkey->key_status[0], sidkey->des_key[0].iv,
          compute_remain_lifetime(&sidkey->des_key[0]));

     buginf(" Status[1] : %x\tDES IV[1]  : %llx\tKey Life[1]: %u sec\n",
          sidkey->key_status[1], sidkey->des_key213
1].iv,
          compute_remain_lifetime(&sidkey->des_key[1]));
```

debug cable mac

To display MAC-layer information for the specified cable modem, use the **debug cable mac** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable mac

no debug cable mac

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Caution Do not use this command if you have a large number of modems on your network. The Cisco uBR7246 universal broadband router will become flooded with console printouts.

Examples

The following example shows the return for the MAC layer:

```
Router# debug cable mac

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
```

Table 24 describes the significant fields in the display.

Table 24 debug cable mac 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....	MAC address of the specified cable modem.
Timing offset is....	Time by which to offset the frame transmission upstream so the frame arrives at the expected minislot time at the CMTS.
Power value is FE0, or 0 dB	Raw value derived from the 3137 Broadcom chip. Alternately, the decibel value specifies the relative change in the transmission power level that the cable modem needs to make so transmissions arrive at the CMTS at the desired power level. This desired power level is usually 0, but you can use the CLI to change it via the **cable power-level** command.
Freq Error = ...	Raw value derived from the 3137 Broadcom chip.
Freq offset is	Specifies the relative change in the transmission frequency that the cable modem will make to match the CMTS.

Related Commands

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

debug cable mac-address

To enable debugging for a specified MAC address, use the **debug cable mac-address** privileged EXEC command. To turn off debugging for the specified MAC address, use the **no** form of this command.

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

no debug cable mac-address *address*

Syntax Description

address	Specifies the MAC address of the interface.
mask	(Optional) Specifies the MAC address validation address.
verbose	(Optional) Displays detailed debug information.

Command History

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

Usage Guidelines

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

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

Examples

The following example demonstrates how to enable debugging for all traffic coming from all interfaces of modems with the MAC address 0010.00xx.xxxx:

```
Router# debug cable mac-address 0010.0000 ffff.ff00.000
```

Related Commands

Command	Description
debug cable interface	Enables debugging on the cable interface specified.

debug cable map

To display map debugging messages, use the **debug cable map** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable map

no debug cable map

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Examples

The following example displays all the 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
A sample Map without any data grant
------------------ 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
A sample Map with data grant(s)
------------------ 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 25 shows the significant fields in the display.

Table 25 debug cable map Field Descriptions

Field	Description
sent 5000 MAPs	Total number of maps sent.
MAPs had grant(s) Long Grants	Total number of grants considered long sized by the CMTS.
Total Short Grants	Total number of grants considered short sized by the 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.

Table 25 debug cable map Field Descriptions (continued)

Field	Description
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 cable modems 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.

Related Commands

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

debug cable-modem bpkm

To debug baseline privacy information on a Cisco uBR900 series cable access router, use the **debug cable-modem bpkm** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

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

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

Syntax Description

errors	Provides debugging information about Cisco uBR900 series privacy errors.
events	Provides debugging information about events related to cable baseline privacy.
packets	Provides debugging information about baseline privacy packets.

Command History

Release	Modification
11.3 NA	This command was introduced.

Usage Guidelines

Baseline privacy key management exchanges take place only when both the Cisco uBR900 series 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 modem. Baseline privacy code images for the Cisco uBR900 series contain "k1" in the code image name.

Examples

The following example shows debug output when the headend 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
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series cable access router.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.

Command	Description
debug cable-modem mac	Troubleshoots the Cisco uBR900 series cable access router MAC layer.
debug cable-modem map	Displays the timing from map messages to synchronize messages and the timing between map messages.

debug cable-modem bridge

To debug bridge filter processing information on a Cisco uBR900 series cable access router, use the **debug cable-modem bridge** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem bridge

no debug cable-modem bridge

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

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 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.

Examples

The following example shows sample display output for the **debug cable-modem bridge** command:

```
Router# debug cable-modem bridge

%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 bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.
debug cable-modem map	Displays the timing from MAP messages to synchronize messages and the timing between MAP messages.

debug cable-modem error

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

debug cable-modem error

no debug cable-modem error

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

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 nonreceipt of SYNC messages from the CMTS, reception errors, and bandwidth request failures.

Examples

The following example shows sample display output for the **debug cable-modem error** command:

```
Router# debug cable-modem error

*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 bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.
debug cable-modem map	Displays the timing from MAP messages to sync messages and the timing between MAP messages.

debug cable-modem interrupts

To debug Cisco uBR900 series interrupts, use the **debug cable-modem interrupts** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem interrupts

no debug cable-modem interrupts

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Examples

The following example shows sample debug output for Cisco uBR900 series interrupts:

```
Router# debug cable-modem interrupts

*** BCM3300_rx_mac_msg_interrupt ***
*** BCM3300_rx_mac_msg_interrupt ***
### BCM3300_tx_interrupt ###
*** BCM3300_rx_mac_msg_interrupt ***
### BCM3300_tx_interrupt ###
*** BCM3300_rx_mac_msg_interrupt ***
### BCM3300_tx_interrupt ###
### BCM3300_tx_interrupt ###
### BCM3300_tx_interrupt ###
### BCM3300_tx_interrupt ###
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.
debug cable-modem map	Displays the timing from MAP messages to sync messages and the timing between MAP messages.

debug cable-modem mac

To troubleshoot the Cisco uBR900 series MAC layer, use the **debug cable-modem mac** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem mac {**log** [**verbose**] | **messages**}

no debug cable-modem mac {**log** [**verbose**] | **messages**}

Syntax Description

log	Displays the real-time MAC log.
verbose	(Optional) Displays periodic MAC-layer events, such as ranging.
messages	Displays MAC layer management messages.

Command History

Release	Modification
11.3 NA	This command was introduced.

Usage Guidelines

Of all the available **debug cable-modem** commands, the most useful is **debug cable-modem mac log**.

MAC log messages are written to a circular log file even when debugging is not turned on. These messages include time stamps, 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, enter the **show controllers cable-modem** *number* **mac log** command. The same information is displayed by both commands.

If the Cisco uBR900 series 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 Cisco uBR900 series cable access server router scans for a downstream frequency. The **debug cable-modem mac log** command displays the log from the oldest to the 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 the **verbose** keyword is used.

Examples

The following example shows sample display 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.

```
Router# debug cable-modem mac log

*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
```

The line “0 events dropped due to lack of a chunk” at the end of a display indicates that no log entries were discarded due to a temporary lack of memory, which means the log is accurate and reliable.

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

```
Router# debug cable-modem mac log

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

Router# no debug cable-modem mac log verbose

Cable Modem mac log debugging is off
Router#
574684.234 CMAC_LOG_RNG_REQ_TRANSMITTED
574684.238 CMAC_LOG_RNG_RSP_MSG_RCVD
```

The following example shows display output for the **debug cable-modem mac messages** command. This command causes received cable MAC management messages to be displayed in a verbose format.

```
Router# debug cable-modem mac messages ?

  dynsrv   dynamic service mac messages
  map      map messages received
  reg-req  reg-req  messages transmitted
  reg-rsp  reg-rsp messages received
  rng-req  rng-req  messages transmitted
  rng-rsp  rng-rsp messages received
  sync     Sync messages received
  ucc-req  ucc-req messages received
  ucc-rsp  ucc-rsp messages transmitted
  ucd      UCD messages received
  <cr>
```

The **dynsrv** keyword displays Dynamic Service Add or Dynamic Service Delete messages during the off-hook/on-hook transitions of a phone connected to the Cisco uBR900 series cable access router.

In addition, sent REG-REQ messages are displayed in hexadecimal dump format. The output from this command is very verbose and is usually not needed for normal interface debugging. The command is most useful when attempting to attach a Cisco uBR900 series cable access router to a CMTS that is not DOCSIS-qualified.

For a description of the displayed fields of each message, refer to the DOCSIS Radio Frequency Interface Specification, v1.0 (SP-RFI-I04-980724).

```
Router# debug cable mac messages

*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 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 Management 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 baseline privacy information for a Cisco uBR900 series cable access router.
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for CiscouBR900 series cable access routers.
debug cable-modem map	Displays the timing from MAP messages to synchronize messages and the timing between MAP messages.

debug cable-modem map

To display the timing from MAP messages to synchronized messages and the timing between MAP messages on a Cisco uBR900 series cable access router, use the **debug cable-modem map** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem map

no debug cable-modem map

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Examples

The following example shows display output for the **debug cable-modem 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 baseline privacy information for a Cisco uBR900 series cable access router.
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for CiscouBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.

debug cable phy

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

debug cable phy

no debug cable phy

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of messages generated in the cable phy, which is the physical layer where upstream and downstream activity between the Cisco uBR7246 router and the HFC network is controlled. When this command is activated, any messages generated in the cable phy will be displayed on the Cisco uBR7246 console.

Examples

The following is sample output from the **debug cable phy** command:

```
Router# debug cable phy

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** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable privacy

no debug cable privacy

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of baseline privacy. When this command is activated, any messages generated by the spectrum manager will be displayed on the Cisco uBR7246 console.

Examples

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

```
Router# debug cable privacy
Removing both odd and even keys for sid %x.

     Invalid Len for TLV_SERIAL_NUM_TYPE : %d.

     Invalid Len for TLV_MANUF_ID_TYPE : %d.

     Invalid Len for TLV_MANUF_ID_TYPE : %d.
```

debug cable qos

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

debug cable qos

no debug cable qos

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of QoS. When this command is activated, any messages related to QoS parameters will be displayed on the Cisco uBR7246 console.

Examples

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

```
Router# debug cable qos

     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 cable modems on the HFC network, use the **debug cable range** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable range

no debug cable range

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of ranging messages from cable modems on the HFC network. When this command is activated, any ranging messages generated when cable modems request or change their upstream frequencies will be displayed on the Cisco uBR7246 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 is sample output from the **debug cable range** command when a modem first seeks to establish a link to the Cisco uBR7246 universal broadband router:

```
Router# debug cable range

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 SID value of 0 indicates that the modem has no assigned service identifier. The "CM mac address" is the MAC address of the radio frequency (RF) interface of the modem, not its Ethernet interface. The "Timing offset" is a measure of the distance between the modem and the Cisco uBR7246 universal broadband router expressed in 10.24-MHz clocks. This value is adjusted down to zero by the maintenance procedures. The first sixteen bytes of the prepended header of the message are dumped in hexadecimal.

The following is sample output when the modem is first assigned a SID during initial maintenance:

```
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 sample output when the modem is reassigned the same 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 sample output when the modem is polled by the uBR7246 universal broadband router during station maintenance. Polling happens at a minimum rate of once every 10 seconds.

```
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
```

debug cable reset

To display reset messages from cable interfaces, use the **debug cable reset** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable reset

no debug cable reset

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates display of reset messages from cable interfaces.

Examples

The following is sample output from the **debug cable reset** command when the interface is reset due to complete loss of receive packets:

```
Router# debug cable reset

Resetting CMTS interface.
```

debug cable specmgmt

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

debug cable specmgmt

no debug cable specmgmt

Syntax Description

This command has no arguments or keywords.

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 uBR7246 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 is sample output from the **debug cable specmgmt** command:

```
Router# debug cable specmgmt

cmts_next_frequency(0x60A979AC, 1, 1)
```

The following is sample output when the frequency hop was commanded:

```
add_interface_to_freq(0x60BD3734, 0x60C44F68)
```

The following is sample output when the interface was added to a the interface list of a frequency:

```
set_upstream(0x60A979AC,1,21000000,-5)
```

The following is sample output when the spectrum management has set the frequency and power level of an upstream port:

```
cmts_frequency_hop_decision(0x60B57FEC)
```

debug cable startalloc

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

debug cable startalloc

no debug cable startalloc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of any channel allocations on the HFC network. When this command is activated, any messages generated when channels are allocated to cable modems on the HFC network will be displayed on the Cisco uBR7246 console.

Examples

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

```
Router# debug cable startalloc

MAP startalloc adjusted by <n> mslots
```

This output indicates time-slot MAP processing is active.

debug cable telco-return

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

debug cable telco-return

no debug cable telco-return

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for Telco return events is not enabled.

Command History

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

Examples

```
Router# debug cable telco-return

CMTS telco-return debugging is on
```

Related Commands

Command	Description
debug cable ucc	Displays debug messages for Telco return events.

debug cable ucc

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

debug cable ucc

no debug cable ucc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of any UCC messages generated when cable modems request or are assigned a new channel. When this command is activated, any messages related to upstream channel changes will be displayed on the Cisco uBR7246 console.

Examples

The following is sample output from the **debug cable ucc** command when moving a modem from one upstream channel to another:

```
Router# debug cable ucc

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
```

debug cable ucd

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

debug cable ucd

no debug cable ucd

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates debugging of any UCD messages. UCD messages contain information about upstream channel characteristics and are sent to the cable modems 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 will be displayed on the Cisco uBR7246 console.

Examples

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

```
Router# debug cable ucd

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 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
```

debug call fallback detail

To display details of the voice fallback, use the **debug call fallback detail** EXEC command. To disable debugging output, use the **no** form of this command.

debug call fallback detail

no debug call fallback detail

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled.

Command Modes

EXEC

Command History

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

Examples

The following example depicts a call coming in to 1.1.1.4 with codec type g729r8. Because there is no cache entry for this destination, a probe is sent and values are inserted into the cache. A lookup is performed again, entry is found, and a fallback decision is made to admit the call.

```
Router# debug call fallback detail

When cache is empty:
debug call fallback detail:
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_check:no entry exists, enqueueing probe info... 1.1.1.4, codec:g729r8
2d19h:fb_main:Got FB_APP_INQ event
2d19h:fb_main:Dequeued prob info: 1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_cache_insert:insert:1.1.1.4, codec:g729r8
2d19h:fb_cache_insert:returning entry:1.1.1.4, codec:g729r8
2d19h:fb_initiate_probe:Creating probe... 1.1.1.4, codec:g729r8
2d19h:fb_initiate_probe:Created and started on probe #13, 1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
2d19h:fb_check:returned FB_CHECK_TRUE, 1.1.1.4, codec:g729r8
2d19h:fb_main:calling callback function with:TRUE
```

The following example depicts a call coming in to 1.1.1.4 with codec g729r8. A lookup is performed, entry is found, and a fallback decision is made to admit the call.

```
Router# debug call fallback detail

When cache is full:
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
```

```
2d19h:fb_check:returned FB_CHECK_TRUE, 1.1.1.4, codec:g729r8
2d19h:fb_main:calling callback function with:TRUE
```

debug call fallback probes

To display details of the voice fallback probes, use the **debug call fallback probes** EXEC command. To disable debugging output, use the **no** form of this command.

debug call fallback probes

no debug call fallback probes

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled.

Command Modes

EXEC

Command History

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

Examples

The following example depicts a call coming in to 1.1.1.4 with codec type g729r8. Because there is no cache entry for this IP address, a g729r8 probe is initiated. The probe consists of 20 packet returns with an average delay of 43 milliseconds. The "jitter out" is jitter from source to destination router and "jitter in" is jitter from destination to source router. The delay, loss, and Calculated Planning Impairment Factor (ICPIF) values following g113_calc_icpif are the instantaneous values, whereas those values following "New smoothed values" are the values after applying the smoothing with weight 65.

```
Router# debug call fallback probes

2d19h:fb_initiate_probe:Probe payload is 32
2d19h:fb_main:NumOfRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0->
1.1.1.4, codec:g729r8
2d19h:g113_calc_icpif(delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0

2d19h:fb_main:Probe timer expired, 1.1.1.4, codec:g729r8
2d19h:fb_main:NumOfRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0->
1.1.1.4, codec:g729r8
2d19h:g113_calc_icpif(delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0
2d19h:fb_main:New smoothed values:inst_weight=65, ICPIF=0, Delay=43, Loss=0 -> 1.1.1.4,
codec:g729r8
```

debug call-mgmt

To display debugging information for call accounting, including modem and time slot usage, for active and recent calls, use the **debug call-mgmt** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug call-mgmt

no debug call-mgmt

Syntax Description

This command has no arguments or keywords.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.

Examples

The following is an example of the debug output that will be received after the **debug call-mgmt** command has been enabled:

```
Router# debug call-mgmt

Call Management debugging is on
Router#
Dec 26 13:57:27.710: msg_to_calls_mgmt: msg type CPM_NEW_CALL_CSM_CONNECT received
Dec 26 13:57:27.714: In actv_c_proc_message,
   access type CPM_INSERT_NEW_CALL,
   call type CPM_ISDN_ANALOG:
     CSM completed connecting a new modem call
.
.
.
Dec 26 13:57:45.906: msg_to_calls_mgmt: msg type CPM_NEW_CALL_ISDN_CONNECT received
Dec 26 13:57:45.906: In actv_c_proc_message,
   access type CPM_INSERT_NEW_CALL,
   call type CPM_ISDN_ANALOG:
     Added a new ISDN analog call to the active-calls list
     CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
     Mdm-Slot#1, Mdm-Port#3, TTY#219
.
.
.
```

```
Dec 26 13:58:25.682: Call mgmt per minute statistics:
     active list length: 1
     history list length: 3
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 1
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 2
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 3
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 4
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 5
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 6
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 7
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 8
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 9
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 10
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 11
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 12
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 13
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 14
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 15
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 16
Dec 26 13:58:25.686:      1 timeslots active at slot 7, ctrlr 17
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 18
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 19
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 20
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 21
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 22
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 23
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 24
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 25
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 26
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 27
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 28

Router# clear int as1/03

Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received
Dec 26 13:58:26.538: In actv_c_proc_message,
   access type CPM_REMOVE_DISC_CALL,
   call type CPM_ISDN_ANALOG:
     Removed a disconnected ISDN analog call
     CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
Dec 26 13:58:26.538:      Mdm-Slot#1, Mdm-Port#3, TTY#219
```

Table 26 describes the significant fields shown in the display.

Table 26 debug call-mgmt Command Field Descriptions

Field	Description
CPM_NEW_CALL_CSM_CONNECT	Indicates the arrival of a new call.
access type CPM_INSERT_NEW_CALL, call type CPM_ISDN_ANALOG:	Indicates that the new call is an analog ISDN B-channel call (either a voice call or a call over an analog modem), rather than a digital (V.110) call.
CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Mdm-Slot#1, Mdm-Port#3, TTY#219	Indicates that the call is connected via the B-channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).

Table 26 debug call-mgmt Command Field Descriptions (continued)

Field	Description
Dec 26 13:58:25.682: Call mgmt per minute statistics: active list length: 1 history list length: 3	Displays periodic statistics that give the allocation state of each DSX1 interface present in the system, as well as the number of current (active) and recent (history) calls.
Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received	Indicates that the analog ISDN B-channel call has been disassociated from a modem.
access type CPM_REMOVE_DISC_CALL, call type CPM_ISDN_ANALOG: Removed a disconnected ISDN analog call	Indicates that the analog ISDN B-channel call has been disconnected.
CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Dec 26 13:58:26.538: Mdm-Slot#1, Mdm-Port#3, TTY#219	Indicates that the call has been disconnected via the B-channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).

debug call rsvp-sync events

To display events that occur during RSVP setup, use the **debug call rsvp-sync events** privileged EXEC command. To restore the default condition, use the **no** form of this command.

debug call rsvp-sync events

no debug call rsvp-sync events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI1	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

It is highly recommended that you log the output from the **debug call rsvp-sync events** command to a buffer, rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows a portion of sample output for a call initiating RSVP when using the **debug call rsvp-sync events** command:

```
00:03:25:  Parameters: localip: 10.19.101.117 :localport: 16660

00:03:25:  Parameters: remoteip: 10.19.101.116 :remoteport: 17568

00:03:25:  QoS Primitive Event for Call id 0x1 : QoS Listen
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498

00:03:25: Hashed entry 0x1 in call table 0x61FC2498

00:03:25: Entry Not found

00:03:25:  Parameters: localip: 10.19.101.117

00:03:25:  remoteip: 10.19.101.116

00:03:25:  QoSpcb : 0x61FC34D8

00:03:25:  Response Status : 0
Starting timer for call with CallId 0x1 for 10000 secs

00:03:25:  Handling QoS Primitive QoS Listen

00:03:25:  Establishing RSVP RESV state : rsvp_request_reservation()

00:03:25:  For streams from 10.19.101.116:17568 to 10.19.101.117:16660
```

```
00:03:25:  RSVP Confirmation required

00:03:25:  QoS Primitive Event for Call id 0x1 : QoS Resv
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498

00:03:25: Hashed entry 0x1 in call table 0x61FC2498

00:03:25:  Initiating RVSP PATH messages to be Sent : reg_invoke_rsvp_advertise_sender()

00:03:25:  Advertizing for streams to 10.19.101.116:17568 from 10.19.101.117:16660

00:03:25:  RESV notification event received is : 2

00:03:25: Received RESVCONFIRM

00:03:25:  RESV CONFIRM message received from 10.19.101.116 for RESV setup from
10.19.101.117

00:03:25:  RESV event received is : 0

00:03:25:  RESV message received from 10.19.101.116:17568 for streams from
10.19.101.117:16660

00:03:25:  RESERVATIONS ESTABLISHED : CallId: 1Stop timer and notify Session Protocol of
Success (ie. if notification requested)

00:03:25:  Invoking spQoSresvCallback with Success
```

Related Commands

Command	Description
call rsvp-sync	Enables synchronization between RSVP and the H.323 voice signalling protocol.
call rsvp-sync resv-timer	Sets the timer for RSVP reservation setup.
debug call rsvp-sync func-trace	Displays messages about the software functions called by RSVP synchronization.
show call rsvp-sync conf	Displays the RSVP synchronization configuration.
show call rsvp-sync stats	Displays statistics for calls that attempted RSVP reservation.

debug call rsvp-sync func-trace

To display messages about software functions called by RSVP, use the **debug call rsvp-sync func-trace** privileged EXEC command. To restore the default condition, use the **no** form of this command.

debug call rsvp-sync func-trace

no debug call rsvp-sync func-trace

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI1	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

It is highly recommended that you log the output from the **debug call rsvp-sync func-trace** command to a buffer, rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows a portion of sample output for a call initiating RSVP when using the **debug call rsvp-sync func-trace** command in conjunction with the **debug call rsvp-sync events** command:

```
00:03:41: Entering Function QoS_Listen

00:03:41: Parameters:localip:10.10.101.116 :localport:17568

00:03:41:remoteip:10.10.101.117 :remoteport:0

00:03:41: Entering Function qos_dequeue_event

00:03:41: Entering Function process_queue_event

00:03:41: QoS Primitive Event for Call id 0x2 :QoS Listen
00:03:41: Entering Function get_pcb

00:03:41: Entering Function hash_tbl_lookup

00:03:41:Lookup to be done on hashkey 0x2 in hash table 0x61FAECD8

00:03:41: Entering Function hash_func

00:03:41:Hashed entry 0x2 in call table 0x61FAECD8

00:03:41:Entry Not found

00:03:41: Entering Function qos_dequeue_pcb
```

```
00:03:41: Entering Function qos_initialize_pcb

00:03:41: Parameters:localip:10.10.101.116

00:03:41:remoteip:10.10.101.117

00:03:41: QoSpcb :0x61FAFD18

00:03:41: Response Status :0

00:03:41: Entering Function hash_tbl_insert_entry

00:03:41: Entering Function hash_func

00:03:41: Handling QoS Primitive QoS Listen

00:03:41: Entering Function qos_dequeue_hash_port_entry

00:03:41: Entering Function qos_port_tbl_insert_entry

00:03:41: Entering Function hash_func

00:03:41: Doing RSVP Listen :rsvp_add_ip_listen_api()
```

Related Commands

Command	Description
call rsvp-sync	Enables synchronization between RSVP and the H.323 voice signaling protocol.
call rsvp-sync resv-timer	Sets the timer for RSVP reservation setup.
debug call rsvp-sync events	Displays the events that occur during RSVP synchronization.
show call rsvp-sync conf	Displays the RSVP synchronization configuration.
show call rsvp-sync stats	Displays statistics for calls that attempted RSVP reservation.

debug callback

To display callback events when the router is using a modem and a chat script to call back on a terminal line, use the **debug callback** privileged EXEC command. The **no** form of this command disables debugging output.

debug callback

no debug callback

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is useful for debugging chat scripts on PPP and ARAP lines that use callback mechanisms. The output provided by the **debug callback** command shows you how the call is progressing when used with the **debug ppp** or **debug arap** commands.

Examples

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

```
Router# debug callback

TTY7 Callback process initiated, user: exec_test dialstring 123456
TTY7 Callback forced wait = 4 seconds
TTY7 Exec Callback Successful - await exec/autoselect pickup
TTY7: Callback in effect
```

Related Commands

Command	Description
debug arap	Displays ARAP events.
debug ppp	Displays information on traffic and exchanges in an internetwork implementing the PPP.

debug ccaal2 session

To display the ccaal2 function calls during call setup and teardown, use the **debug ccaal2 session** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug ccaal2 session

no debug ccaal2 session

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for AAL2 sessions is not enabled.

Command History

Release	Modification
12.1(1)XA	This command was introduced on the Cisco MC380 series.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command when troubleshooting an AAL2 trunk setup or teardown problem.

Examples

The following example shows sample output from the **debug ccaal2 session** command for a forced shutdown of a voice port:

```
Router# debug ccaal2 session

Router(config)# voice-port 1/1

Router(config-voiceport)# shutdown

Router(config-voiceport)#
3d21h:%Voice port in use.  Force shutdown.
3d21h:%Voice-port 1/1 is down.
3d21h:ccaal2_call_disconnect:peer tag 0
3d21h:ccaal2_evhandle_call_disconnect:Entered
3d21h:ccaal2_call_cleanup:freeccb 1, call_disconnected 1ccaal2_receive:xmitFunc is NULL
ccaal2_receive:xmitFunc is NULL

3d21h:starting incoming timer:Setting accept_incoming to FALSE and
3d21h:timer 2:(0x126AD48)starts - delay (70000)
3d21h:ccaal2_call_cleanup:Generating Call record
3d21h:cause=81 tcause=81    cause_text=unspecified
3d21h:ccaal2_call_cleanup:ccb 0x1506A84, vdbPtr 0x15ACFD0
        freeccb_flag=1, call_disconnected_flag=1
3d21h:%LINK-3-UPDOWN:Interface FXS 1/1, changed state to Administrative Shutdown
```

The following example shows sample output from the **debug ccaal2 session** command for a trunk setup on a voice port:

```
router(config-voiceport)# no shutdown
router(config-voiceport)#
3d21h:%Voice-port 1/1 is up.
3d21h:%LINK-3-UPDOWN:Interface FXS 1/1, changed state to up
```

```
3d21h:ccaal2_call_setup_request:Entered
3d21h:ccaal2_evhandle_call_setup_request:Entered
3d21h:ccaal2_initialize_ccb:preferred_codec set(-1)(0)
3d21h:ccaal2_evhandle_call_setup_request:preferred_codec set(5)(40). VAD is 0
3d21h:ccaal2_call_setup_trunk:subchannel linking successful

3d21h:ccaal2_caps_ind:PeerTag = 2007
3d21h:     codec(preferred) = 1, fax_rate = 2, vad = 1
3d21h:     cid = 25, config_bitmask = 0, codec_bytes = 40, signal_type=8
3d21h:encap VOAAL2
3d21h:%HTSP-5-UPDOWN:Trunk port(channel) [1/1] is up
```

Related Commands

Command	Description
show debug	Displays which debug commands are enabled.

debug ccfrf11 session

To display the ccfrf11 function calls during call setup and teardown, use the **debug ccfrf11 session** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccfrf11 session

no debug ccfrf11 session

Syntax Description

This command has no keywords or arguments.

Command History

Release	Modification
12.0(3)XG	This command was introduced on the Cisco 2600 and Cisco 3600 series routers.
12.0(4)T	This command was integrated into Cisco IOS Release 12.0(4)T.
12.0(7)XK	This command was first supported on the Cisco MC3810 series.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command to display debug information about the various FRF.11 VoFR service provider interface (SPI) functions. Note that this debug command does not display any information regarding the proprietary Cisco switched-VoFR SPI.

This debug is useful only when the session protocol is “frf11-trunk.”

Examples

The following example shows sample output from the **debug ccfr11 session** command:

```
Router# debug ccfrf11 session

INCOMING CALL SETUP (port setup for answer-mode):
*Mar  6 18:04:07.693:ccfrf11_process_timers:scb (0x60EB6040) timer (0x60EB6098) expired
*Mar  6 18:04:07.693:Setting accept_incoming to TRUE
*Mar  6 18:04:11.213:ccfrf11_incoming_request:peer tag 800:callingNumber=+2602100,
        calledNumber=+3622110
*Mar  6 18:04:11.213:ccfrf11_initialize_ccb:preffered_codec set(-1)(0)
*Mar  6 18:04:11.213:ccfrf11_evhandle_incoming_call_setup_request:calling +2602100,
        called +3622110 Incoming Tag 800
*Mar  6 18:04:11.217:ccfrf11_caps_ind:PeerTag = 800
*Mar  6 18:04:11.217:     codec(preferred) = 4, fax_rate = 2, vad = 2
*Mar  6 18:04:11.217:     cid = 30, config_bitmask = 0, codec_bytes = 20, signal_type=2
*Mar  6 18:04:11.217:     required_bandwidth 8192
*Mar  6 18:04:11.217:ccfrf11_caps_ind:Bandwidth reservation of 8192 bytes succeeded.
*Mar  6 18:04:11.221:ccfrf11_evhandle_call_connect:Entered

CALL SETUP (MASTER):
5d22h:ccfrf11_call_setup_request:Entered
5d22h:ccfrf11_evhandle_call_setup_request:Entered
5d22h:ccfrf11_initialize_ccb:preffered_codec set(-1)(0)
5d22h:ccfrf11_evhandle_call_setup_request:preffered_codec set(9)(24)
5d22h:ccfrf11_call_setup_trunk:subchannel linking successful
5d22h:ccfrf11_caps_ind:PeerTag = 810
5d22h:     codec(preferred) = 512, fax_rate = 2, vad = 2
```

```
5d22h:     cid = 30, config_bitmask = 1, codec_bytes = 24, signal_type=2
5d22h:     required_bandwidth 6500
5d22h:ccfrf11_caps_ind:Bandwidth reservation of 6500 bytes succeeded.

CALL TEARDOWN:
*Mar  6 18:09:14.805:ccfrf11_call_disconnect:peer tag 0
*Mar  6 18:09:14.805:ccfrf11_evhandle_call_disconnect:Entered
*Mar  6 18:09:14.805:ccfrf11_call_cleanup:freeccb 1, call_disconnected 1
*Mar  6 18:09:14.805:ccfrf11_call_cleanup:Setting accept_incoming to FALSE and starting
        incoming timer
*Mar  6 18:09:14.809:timer 2:(0x60EB6098)starts - delay (70000)
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:Alive timer stopped
*Mar  6 18:09:14.809:timer 1:(0x60F64104) stops
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:Generating Call record
*Mar  6 18:09:14.809:cause=10 tcause=10    cause_text="normal call clearing."
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:Releasing 8192 bytes of reserved bandwidth
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:ccb 0x60F6404C, vdbPtr 0x610DB7A4
        freeccb_flag=1, call_disconnected_flag=1
```

Related Commands

Command	Description
debug call-mgmt	Displays the ccswvoice function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug vtsp session	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug cch323 h225

To trace the state transition of the H.225 state machine based on the processed event, use the **debug cch323 h225** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cch323 h225

no debug cch323 h225

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

State Descriptions

The state definitions of the different states of the H.225 state machine are as follows:

- H225_IDLE—This is the initial state of the H.225 state machine. The H.225 state machine is in this state before issuing a call setup request (for the outbound IP call case) or when ready to receive an incoming IP call.
- H225_SETUP—This is the call setup state. The state machine changes to this state after sending out a call setup request or after the reception of an incoming call indication.
- H225_ALERT—This is the call alerting state. The state machine changes to this state after sending the alerting message or after the reception of an alerting message from the peer.
- H225_CALLPROC—This is the call proceeding state.
- H225_ACTIVE—This is the call connected state. In this state, the call is active. The state machine changes to this state after sending the connect message to the peer or after the reception of the connect message from the peer.
- H225_WAIT_FOR_ARQ—This is the state where the H.225 state machine is waiting for the completion of the ARQ process from the RAS state machine.
- H225_WAIT_FOR_DRQ—This is the state where the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.
- H225_WAIT_FOR_H245—This is the state where the H.225 state machine is waiting for the success or failure from the H.245 state machine.

Events Description

The event definitions of the different events of the H.225 state machine are as follows:

- H225_EVENT_NONE— No event.
- H225_EVENT_ALERT—This event indicates to the H.225 state machine to send an alert message to the peer.
- H225_EVENT_ALERT_IND—This event indicates to the H.225 state machine that an alert message arrived from the peer.

- H225_EVENT_CALLPROC—This event indicates to the H.225 state machine to send a call proceeding message to the peer.
- H225_EVENT_CALLPROC_IND—This event indicates to the H.225 state machine that a call proceeding message is received from the peer.
- H225_EVENT_REJECT—This event indicates to the H.225 state machine to reject the call setup request from the peer.
- H225_EVENT_REJECT_IND—This event indicates to the H.225 state machine that a call setup request to the peer is rejected.
- H225_EVENT_RELEASE—This event indicates to the H.225 state machine to send a release complete message to the peer.
- H225_EVENT_RELEASE_IND—This event indicates to the H.225 state machine that a release complete message is received from the peer.
- H225_EVENT_SETUP—This event indicates to the H.225 state machine to send a setup message to the peer.
- H225_EVENT_SETUP_IND—This event indicates to the H.225 state machine that a setup message is received from the peer.
- H225_EVENT_SETUP_CFM—This event indicates to the H.225 state machine to send a connect message to the peer.
- H225_EVENT_SETUP_CFM_IND—This event indicates to the H.225 state machine that a connect message arrived from the peer.
- H225_EVENT_RAS_SUCCESS—This event indicates to the H.225 state machine that the pending RAS operation is successful.
- H225_EVENT_RAS_FAILED—This event indicates to the H.225 state machine that the pending RAS operation failed.
- H225_EVENT_H245_SUCCESS—This event indicates to the H.225 state machine that the pending H.245 operation is successful.
- H225_EVENT_H245_FAILED—This event indicates to the H.225 state machine that the pending H.245 operation failed.

Examples

The following is example output from the **debug cch323 h225** command.

```
Router# debug cch323 h225

20:59:17:Set new event H225_EVENT_SETUP
20:59:17:H225 FSM:received event H225_EVENT_SETUP while at state H225_IDLE
20:59:17:Changing from H225_IDLE state to H225_SETUP state
20:59:17:cch323_h225_receiver:received msg of type SETUPCFM_CHOSEN
20:59:17:H225 FSM:received event H225_EVENT_SETUP_CFM_IND while at state
H225_SETUP
20:59:17:Changing from H225_SETUP state to H225_ACTIVE state
20:59:17:Set new event H225_EVENT_H245_SUCCESS
20:59:17:H225 FSM:received event H225_EVENT_H245_SUCCESS while at state
H225_ACTIVE
20:59:20:Set new event H225_EVENT_RELEASE
20:59:20:H225 FSM:received event H225_EVENT_RELEASE while at state
H225_ACTIVE
20:59:20:Changing from H225_ACTIVE state to H225_WAIT_FOR_DRQ state
20:59:20:Set new event H225_EVENT_RAS_SUCCESS
20:59:20:H225 FSM:received event H225_EVENT_RAS_SUCCESS while at state
H225_WAIT_FOR_DRQ
```

```
20:59:20:Changing from H225_WAIT_FOR_DRQ state to H225_IDLE state
```

debug cch323 h245

To trace the state transition of the H.245 state machine based on the processed events, use the **debug cch323 h245** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cch323 h245

no debug cch323 h245

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

The H.245 state machines include the following three state machines:

- Master slave determination (MSD) state machine
- Capability exchange (CAP) state machine
- Open logical channel (OLC) state machine

State Definitions

The state definitions are as follows:

- H245_MS_NONE—This is the initial state of the master slave determination state machine.
- H245_MS_WAIT—In this state, a Master Slave Determination message is sent, and the device is waiting for the reply.
- H245_MS_DONE— The result is in.
- H245_CAP_NONE—This is the initial state of the capability exchange state machine.
- H245_CAP_WAIT—In this state, a capability exchange message is sent, and the device is waiting for reply.
- H245_CAP_DONE—The result is in.
- H245_OLC_NONE—This is the initial state of the open logical channel state machine.
- H245_OLC_WAIT: OLC message sent, and the device is waiting for reply.
- H245_OLC_DONE: OLC done.

Event Definitions

The event definitions are as follows:

- H245_EVENT_MSD—Send MSD message
- H245_EVENT_MS_CFM—Send MSD acknowledge message
- H245_EVENT_MS_REJ—Send MSD reject message
- H245_EVENT_MS_IND—Received MSD message
- H245_EVENT_CAP—Send CAP message

- H245_EVENT_CAP_CFM—Send CAP acknowledge message
- H245_EVENT_CAP_REJ—Send CAP reject message
- H245_EVENT_CAP_IND—Received CAP message
- H245_EVENT_OLC—Send OLC message
- H245_EVENT_OLC_CFM—Send OLC acknowledge message
- H245_EVENT_OLC_REJ—Send OLC reject message
- H245_EVENT_OLC_IND—Received OLC message

Examples

The following is sample output for the **debug cch323 h245** command.

```
Router# debug cch323 h245

20:58:23:Changing to new event H245_EVENT_MSD
20:58:23:H245 MS FSM:received event H245_EVENT_MSD while at state
H245_MS_NONE
20:58:23:changing from H245_MS_NONE state to H245_MS_WAIT state
20:58:23:Changing to new event H245_EVENT_CAP
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP while at state
H245_CAP_NONE
20:58:23:changing from H245_CAP_NONE state to H245_CAP_WAIT state
20:58:23:cch323_h245_receiver:received msg of type
M_H245_MS_DETERMINE_INDICATION
20:58:23:Changing to new event H245_EVENT_MS_IND
20:58:23:H245 MS FSM:received event H245_EVENT_MS_IND while at state
H245_MS_WAIT
20:58:23:cch323_h245_receiver:received msg of type
M_H245_CAP_TRANSFER_INDICATION
20:58:23:Changing to new event H245_EVENT_CAP_IND
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_IND while at state
H245_CAP_WAIT
20:58:23:cch323_h245_receiver:received msg of type
M_H245_MS_DETERMINE_CONFIRM
20:58:23:Changing to new event H245_EVENT_MS_CFM
20:58:23:H245 MS FSM:received event H245_EVENT_MS_CFM while at state
H245_MS_WAIT
20:58:23:changing from H245_MS_WAIT state to H245_MS_DONE state
0:58:23:cch323_h245_receiver:received msg of type M_H245_CAP_TRANSFER_CONFIRM
20:58:23:Changing to new event H245_EVENT_CAP_CFM
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_CFM while at state
H245_CAP_WAIT
20:58:23:changing from H245_CAP_WAIT state to H245_CAP_DONE state
20:58:23:Changing to new event H245_EVENT_OLC
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC while at state
H245_OLC_NONE
20:58:23:changing from H245_OLC_NONE state to H245_OLC_WAIT state
20:58:23:cch323_h245_receiver:received msg of type
M_H245_UCHAN_ESTABLISH_INDICATION
20:58:23:Changing to new event H245_EVENT_OLC_IND
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_IND while at state
H245_OLC_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_UCHAN_ESTAB_ACK
20:58:23:Changing to new event H245_EVENT_OLC_CFM
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_CFM while at state
H245_OLC_WAIT
20:58:23:changing from H245_OLC_WAIT state to H245_OLC_DONE state
```

debug cch323 ras

To trace the state transition of the RAS state machine based on the processed events, use the **debug cch323 ras** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cch323 ras

no debug cch323 ras

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

RAS operates in two state machines. One global state machine controls the overall RAS operation of the gateway. The other state machine is a per-call state machine that controls the active calls.

State Definitions

The state definitions of the different states of the RAS state machine are as follows:

- CCH323_RAS_STATE_NONE—This is the initial state of the RAS state machine.
- CCH323_RAS_STATE_GRQ—The state machine is in the GRQ state. In this state, the gateway is discovering a gatekeeper.
- CCH323_RAS_STATE_RRQ—The state machine is in the RRQ state. In this state, the gateway is registering with a gatekeeper.
- CCH323_RAS_STATE_IDLE—The global state machine is in the idle state.
- CCH323_RAS_STATE_URQ—The state machine is in the URQ state. In this state, the gateway is in the process of unregistering with a gatekeeper.
- CCH323_RAS_STATE_ARQ—The per-call state machine is in the process of admitting a new call.
- CCH323_RAS_STATE_ACTIVE—The per-call state machine is in the call active state.
- CCH323_RAS_STATE_DRQ—The per-call state machine is in the process of disengaging an active call.

Event Definitions

The event definitions of the different states of the RAS state machine are as follows:

- CCH323_RAS_EVENT_NONE—Nothing
- CCH323_RAS_EVENT_GWUP—Gateway is coming up
- CCH323_RAS_EVENT_GWDWN—Gateway is going down
- CCH323_RAS_EVENT_NEWCALL:—New call
- CCH323_RAS_EVENT_CALLDISC—Call disconnect
- CCH323_RAS_EVENT_GCF—Received GCF
- CCH323_RAS_EVENT_GRJ—Received GRJ

- CCH323_RAS_EVENT_ACF—Received ACF
- CCH323_RAS_EVENT_ARJ—Received ARJ
- CCH323_RAS_EVENT_SEND_RRQ—Send RRQ
- CCH323_RAS_EVENT_RCF—Received RCF
- CCH323_RAS_EVENT_RRJ—Received RRJ
- CCH323_RAS_EVENT_SEND_URQ—Send URQ
- CCH323_RAS_EVENT_URQ—Received URQ
- CCH323_RAS_EVENT_UCF—Received UCF
- CCH323_RAS_EVENT_SEND_UCF—Send UCF
- CCH323_RAS_EVENT_URJ—Received URJ
- CCH323_RAS_EVENT_BCF—Received BCF
- CCH323_RAS_EVENT_BRJ—Received BRJ
- CCH323_RAS_EVENT_DRQ—Received DRQ
- CCH323_RAS_EVENT_DCF—Received DCF
- CCH323_RAS_EVENT_SEND_DCF—Send DCF
- CCH323_RAS_EVENT_DRJ—Received DRJ
- CCH323_RAS_EVENT_IRQ—Received IRQ
- CCH323_RAS_EVENT_IRR—Send IRR
- CCH323_RAS_EVENT_TIMEOUT—Message timeout

Examples

The following is sample output from the **debug cch323 ras** command.

```
Router# debug cch323 ras

20:58:49:Changing to new event CCH323_RAS_EVENT_SEND_RRQ
cch323_run_ras_sm:received event CCH323_RAS_EVENT_SEND_RRQ while at CCH323_RAS_STATE_IDLE
state
cch323_run_ras_sm:changing to CCH323_RAS_STATE_RRQ state
cch323_ras_receiver:received msg of type RCF_CHOSEN
cch323_run_ras_sm:received event CCH323_RAS_EVENT_RCF while at CCH323_RAS_STATE_RRQ state
cch323_run_ras_sm:changing to CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_NEWCALL while at
CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_ARQ
cch323_ras_receiver:received msg of type ACF_CHOSEN
20:58:59:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_ACF while at
CCH323_RAS_STATE_ARQ state
20:58:59:cch323_percall_ras_sm:changing to new state
CCH323_RAS_STATE_ACTIVE
20:59:02:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_CALLDISC while
at CCH323_RAS_STATE_ACTIVE state
20:59:02:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_DRQ
cch323_ras_receiver:received msg of type DCF_CHOSEN
20:59:02:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_DCF while at
CCH323_RAS_STATE_DRQ state
20:59:02:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_IDLE
20:59:04:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_IRR while at
CCH323_RAS_STATE_ACTIVE state
20:59:04:cch323_percall_ras_sm:changing to new state
```

```
CCH323_RAS_STATE_ACTIVE
```

debug ccsip all

To enable all SIP-related debugging, use the **debug ccsip all** EXEC command. To disable all debugging output, use the **no** form of this command.

debug ccsip all

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.1.(3)T	The output of the command was changed.

Usage Guidelines

The **debug ccsip all** command enables the following debug SIP commands:

- **debug ccsip calls**
- **debug ccsip error**
- **debug ccsip events**
- **debug ccsip messages**
- **debug ccsip states**

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip all

All SIP call tracing enabled
Router#
*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_NONE, SUBSTATE_NONE)  to
(STATE_IDLE, SUBSTATE_NONE)
*Mar  6 14:10:42:  Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_idle_call_setup
*Mar  6 14:10:42:  act_idle_call_setup:Not using Voice Class Codec

*Mar  6 14:10:42: act_idle_call_setup: preferred_codec set[0] type :g711ulaw bytes: 160
*Mar  6 14:10:42:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_NONE)  to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Mar  6 14:10:42: REQUEST CONNECTION TO IP:166.34.245.231 PORT:5060

*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_CONNECTING)  to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_idle_connection_created
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_idle_connection_created: Connid(1) created to
166.34.245.231:5060, local_port 54113
```

```
*Mar  6 14:10:42: sipSPIAddLocalContact
*Mar  6 14:10:42:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_CONNECTING)  to
(STATE_SENT_INVITE, SUBSTATE_NONE)
*Mar  6 14:10:42: Sent:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Cisco-Guid: 2881152943-2184249548-0-483039712
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427042
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  6 14:10:42: Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  6 14:10:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_sentinvite_new_message
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:42:  Roundtrip delay 4 milliseconds for method INVITE

*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_SENT_INVITE, SUBSTATE_NONE)  to
(STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
*Mar  6 14:10:42: Received:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137
```

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  6 14:10:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:42:  Roundtrip delay 8 milliseconds for method INVITE

*Mar  6 14:10:42: HandleSIP1xxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec      : g711ulaw , bytes :160
Inband Alerting       : 0

*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_PROCEEDING)  to (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
*Mar  6 14:10:46: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  6 14:10:46: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:46:  Roundtrip delay 3536 milliseconds for method INVITE

*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  act_recdproc_new_message: SDP MediaTypes negotiation
successful!
Negotiated Codec      : g711ulaw , bytes :160

*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sipSPIReconnectConnection
*Mar  6 14:10:46:  Queued event from SIP SPI : SIPSPI_EV_RECONNECT_CONNECTION
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  recv_200_OK_for_invite
*Mar  6 14:10:46:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:10:46: 0x624CFEF8 : State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_ALERTING)  to (STATE_ACTIVE, SUBSTATE_NONE)
*Mar  6 14:10:46: The Call Setup Information is :
```

```
Call Control Block (CCB) : 0x624CFEF8
 State of The Call        : STATE_ACTIVE
 TCP Sockets Used         : NO
 Calling Number           : 3660110
 Called Number            : 3660210
 Negotiated Codec         : g711ulaw
 Source IP Address (Media): 166.34.245.230
 Source IP Port    (Media): 20208
 Destn  IP Address (Media): 166.34.245.231
 Destn  IP Port    (Media): 20038
 Destn SIP Addr (Control) : 166.34.245.231
 Destn SIP Port (Control) : 5060
 Destination Name         : 166.34.245.231

*Mar  6 14:10:46: HandleUdpReconnection: Udp socket connected for fd: 1 with
166.34.245.231:5060
*Mar  6 14:10:46: Sent:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 137
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  6 14:10:46: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  6 14:10:46: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  6 14:10:50: Received:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:36:44 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0



*Mar  6 14:10:50: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:54835
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  act_active_new_message
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sact_active_new_message_request
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:10:50:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sipSPIInitiateCallDisconnect : Initiate call
disconnect(16) for outgoing call
```

```
*Mar  6 14:10:50: 0x624CFEF8 : State change from (STATE_ACTIVE, SUBSTATE_NONE)  to
(STATE_DISCONNECTING, SUBSTATE_NONE)
*Mar  6 14:10:50: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612207
Content-Length: 0
CSeq: 101 BYE



*Mar  6 14:10:50:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  act_disconnecting_disconnect
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  6 14:10:50:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
*Mar  6 14:10:50: CLOSE CONNECTION TO CONNID:1

*Mar  6 14:10:50: sipSPIIcpifUpdate :CallState: 4 Playout: 1755 DiscTime:48305031 ConnTime
48304651

*Mar  6 14:10:50: 0x624CFEF8 : State change from (STATE_DISCONNECTING, SUBSTATE_NONE)  to
(STATE_DEAD, SUBSTATE_NONE)
*Mar  6 14:10:50: The Call Setup Information is :

        Call Control Block (CCB) : 0x624CFEF8
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20208
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20038
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:10:50:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200

*Mar  6 14:10:50: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
Router#
```

From the other side of the call, the debug output is as follows:

```
3660-2#debug ccsip all
All SIP call tracing enabled
3660-2#
*Mar  8 17:36:40: Received:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:10:42 GMT
```

```
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Cisco-Guid: 2881152943-2184249548-0-483039712
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427042
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  8 17:36:40: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:54113
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sipSPISipIncomingCall
*Mar  8 17:36:40: 0x624D8CCC : State change from (STATE_NONE, SUBSTATE_NONE)  to
(STATE_IDLE, SUBSTATE_NONE)
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  act_idle_new_message
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sact_idle_new_message_invite
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:36:40:  sact_idle_new_message_invite:Not Using Voice Class Codec

*Mar  8 17:36:40: sact_idle_new_message_invite: Preferred codec[0] type: g711ulaw Bytes
:160
*Mar  8 17:36:40: sact_idle_new_message_invite: Media Negotiation successful for an
incoming call

*Mar  8 17:36:40: sact_idle_new_message_invite: Negotiated Codec      : g711ulaw, bytes
:160
Preferred Codec       : g711ulaw, bytes :160

*Mar  8 17:36:40:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:40: Num of Contact Locations 1 3660110 166.34.245.230 5060

*Mar  8 17:36:40: 0x624D8CCC : State change from (STATE_IDLE, SUBSTATE_NONE)  to
(STATE_RECD_INVITE, SUBSTATE_RECD_INVITE_CALL_SETUP)
*Mar  8 17:36:40: Sent:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  8 17:36:40:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_PROCEEDING
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  act_recdinvite_proceeding
*Mar  8 17:36:40:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_ALERTING
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  8 17:36:40: ccsip_caps_ind: codec(negotiated) = 5(Bytes 160)
*Mar  8 17:36:40: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  8 17:36:40: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
```

```
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  act_recdinvite_alerting
*Mar  8 17:36:40:  180 Ringing with SDP - not likely

*Mar  8 17:36:40:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:40: 0x624D8CCC : State change from (STATE_RECD_INVITE,
SUBSTATE_RECD_INVITE_CALL_SETUP)  to (STATE_SENT_ALERTING, SUBSTATE_NONE)
*Mar  8 17:36:40: Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  8 17:36:44:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_CONNECT
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  act_sentalert_connect
*Mar  8 17:36:44: sipSPIAddLocalContact
*Mar  8 17:36:44:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:44: 0x624D8CCC : State change from (STATE_SENT_ALERTING, SUBSTATE_NONE)  to
(STATE_SENT_SUCCESS, SUBSTATE_NONE)
*Mar  8 17:36:44: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  8 17:36:44: Received:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Max-Forwards: 6
```

```
Content-Type: application/sdp
Content-Length: 137
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  8 17:36:44: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 
166.34.245.230:54113
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  act_sentsucc_new_message
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:36:44: 0x624D8CCC : State change from (STATE_SENT_SUCCESS, SUBSTATE_NONE)  to 
(STATE_ACTIVE, SUBSTATE_NONE)
*Mar  8 17:36:44: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D8CCC
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20038
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20208
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:36:47:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  act_active_disconnect
*Mar  8 17:36:47:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_NONE)  to 
(STATE_ACTIVE, SUBSTATE_CONNECTING)
*Mar  8 17:36:47: REQUEST CONNECTION TO IP:166.34.245.230 PORT:5060

*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_CONNECTING)  to 
(STATE_ACTIVE, SUBSTATE_CONNECTING)
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  act_active_connection_created
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection: Connid(1) created to 
166.34.245.230:5060, local_port 54835
*Mar  8 17:36:47:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_CONNECTING)  to 
(STATE_DISCONNECTING, SUBSTATE_NONE)
*Mar  8 17:36:47: Sent:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:36:44 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0
```

```
*Mar  8 17:36:47: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612207
Content-Length: 0
CSeq: 101 BYE



*Mar  8 17:36:47: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:54113
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  act_disconnecting_new_message
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sact_disconnecting_new_message_response
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:47:  Roundtrip delay 4 milliseconds for method BYE

*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  8 17:36:47:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
*Mar  8 17:36:47: CLOSE CONNECTION TO CONNID:1

*Mar  8 17:36:47: sipSPIIcpifUpdate :CallState: 4 Playout: 1265 DiscTime:66820800 ConnTime
66820420

*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_DISCONNECTING, SUBSTATE_NONE)  to
(STATE_DEAD, SUBSTATE_NONE)
*Mar  8 17:36:47: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D8CCC
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20038
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20208
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:36:47:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200

*Mar  8 17:36:47: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
```

Related Commands

Command	Description
debug ccsip calls	Displays all SIP SPI call tracing and traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Displays SIP SPI errors and traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UAC and the access server.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip calls

To show all SIP Service Provider Interface (SPI) call tracing, use the **debug ccsip calls** command.

debug ccsip calls

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.1.(3)T	The output of the command was changed.

Usage Guidelines

This command traces the SIP call details as they are updated in the SIP call control block.

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip calls

SIP Call statistics tracing is enabled
Router#
*Mar  6 14:12:33: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D078C
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20644
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20500
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:12:40: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D078C
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20644
```

```
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20500
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:12:40:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200

Router#
```

From the other side of the call, the debug output is as follows:

```
Router#debug ccsip calls
SIP Call statistics tracing is enabled
Router#
*Mar  8 17:38:31: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D9560
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20500
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20644
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:38:38: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D9560
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20500
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20644
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:38:38:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip error	Displays SIP SPI errors. This command traces all error messages generated from errors encountered by the SIP subsystem.

Command	Description
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip error

To show SIP SPI errors, use the **debug ccsip error** EXEC command.

debug ccsip error

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.1.(3)T	The output of the command was changed.

Usage Guidelines

This command traces all error messages generated from errors encountered by the SIP subsystem.

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip error

SIP Call error tracing is enabled
Router#

*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_idle_call_setup
*Mar  6 14:16:41:  act_idle_call_setup:Not using Voice Class Codec

*Mar  6 14:16:41: act_idle_call_setup: preferred_codec set[0] type :g711ulaw bytes: 160
*Mar  6 14:16:41: REQUEST CONNECTION TO IP:166.34.245.231 PORT:5060

*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_idle_connection_created
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_idle_connection_created: Connid(1) created to 
166.34.245.231:5060, local_port 55674
*Mar  6 14:16:41: sipSPIAddLocalContact
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:16:41: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 
166.34.245.231:5060
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_sentinvite_new_message
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:41:  Roundtrip delay 4 milliseconds for method INVITE

*Mar  6 14:16:41: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr: 
166.34.245.231:5060
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:41:  Roundtrip delay 8 milliseconds for method INVITE
```

```
*Mar  6 14:16:41: HandleSIP1xxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec      : g711ulaw , bytes :160
Inband Alerting       : 0

*Mar  6 14:16:45: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:45:  Roundtrip delay 3844 milliseconds for method INVITE

*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  act_recdproc_new_message: SDP MediaTypes negotiation
successful!
Negotiated Codec      : g711ulaw , bytes :160

*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sipSPIReconnectConnection
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  recv_200_OK_for_invite
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:16:45: HandleUdpReconnection: Udp socket connected for fd: 1 with
166.34.245.231:5060
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  6 14:16:45: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  6 14:16:45: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  6 14:16:49: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:56101
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  act_active_new_message
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sact_active_new_message_request
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sipSPIInitiateCallDisconnect : Initiate call
disconnect(16) for outgoing call
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  act_disconnecting_disconnect
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  6 14:16:49: CLOSE CONNECTION TO CONNID:1

*Mar  6 14:16:49: sipSPIIcpifUpdate :CallState: 4 Playout: 2945 DiscTime:48340988 ConnTime
48340525

*Mar  6 14:16:49: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
Router#
```

From the other side of the call, the debug output is as follows:

```
Router#debug ccsip error
SIP Call error tracing is enabled
Router#
*Mar  8 17:42:39: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:55674
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sipSPISipIncomingCall
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  act_idle_new_message
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sact_idle_new_message_invite
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:42:39:  sact_idle_new_message_invite:Not Using Voice Class Codec

*Mar  8 17:42:39: sact_idle_new_message_invite: Preferred codec[0] type: g711ulaw Bytes
:160
*Mar  8 17:42:39: sact_idle_new_message_invite: Media Negotiation successful for an
incoming call
```

```
*Mar  8 17:42:39: sact_idle_new_message_invite: Negotiated Codec      : g711ulaw, bytes
:160
Preferred Codec       : g711ulaw, bytes :160

*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:39: Num of Contact Locations 1 3660110 166.34.245.230 5060

*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  act_recdinvite_proceeding
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  8 17:42:39: ccsip_caps_ind: codec(negotiated) = 5(Bytes 160)
*Mar  8 17:42:39: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  8 17:42:39: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  act_recdinvite_alerting
*Mar  8 17:42:39:  180 Ringing with SDP - not likely

*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  act_sentalert_connect
*Mar  8 17:42:42: sipSPIAddLocalContact
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:55674
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  act_sentsucc_new_message
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  act_active_disconnect
*Mar  8 17:42:47: REQUEST CONNECTION TO IP:166.34.245.230 PORT:5060

*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  act_active_connection_created
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection: Connid(1) created to
166.34.245.230:5060, local_port 56101
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:42:47: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:55674
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  act_disconnecting_new_message
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sact_disconnecting_new_message_response
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:47:  Roundtrip delay 0 milliseconds for method BYE

*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  8 17:42:47: CLOSE CONNECTION TO CONNID:1

*Mar  8 17:42:47: sipSPIIcpifUpdate :CallState: 4 Playout: 1255 DiscTime:66856757 ConnTime
66856294

*Mar  8 17:42:47: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Displays all SIP Service Provider Interface (SPI) call tracing and traces the SIP call details as they are updated in the SIP call control block.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.

Command	Description
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip events

To show all SIP SPI events tracing, use the **debug ccsip events** command.

debug ccsip events

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Usage Guidelines

This command traces the events posted to SIP SPI from all interfaces.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip events

SIP Call events tracing is enabled
Router#

*Mar  6 14:17:57:  Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
*Mar  6 14:17:57:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  6 14:17:57:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:18:00:  Queued event from SIP SPI : SIPSPI_EV_RECONNECT_CONNECTION
*Mar  6 14:18:00:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:18:04:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:18:04:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  6 14:18:04:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
Router#
```

From the other side of the call, the debug output is as follows:

```
Router# deb ccsip events

SIP Call events tracing is enabled
Router#

*Mar  8 17:43:55:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:43:55:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_PROCEEDING
*Mar  8 17:43:55:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_ALERTING
*Mar  8 17:43:55:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:43:58:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_CONNECT
*Mar  8 17:43:58:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:44:01:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  8 17:44:01:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  8 17:44:01:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:44:01:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Shows all SIP Service Provider Interface (SPI) call tracing. This command traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Shows SIP SPI errors. This command traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip messages	Shows all SIP SPI message tracing. This command traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.
debug ccsip states	Shows all SIP SPI state tracing. This command traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip messages

To show all SIP SPI message tracing, use the **debug ccsip message**s command.

debug ccsip messages

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

This command traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.

Examples

From one side of the call, the debug output is as follows:

```
Router#debug ccsip message

SIP Call messages tracing is enabled
Router#

*Mar  6 14:19:14: Sent:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Cisco-Guid: 2881152943-2184249568-0-483551624
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427554
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  6 14:19:14: Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:55820
```

```
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  6 14:19:14: Received:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  6 14:19:16: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  6 14:19:16: Sent:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 138
CSeq: 101 ACK
```

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  6 14:19:19: Received:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:45:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612717
CSeq: 101 BYE
Content-Length: 0



*Mar  6 14:19:19: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:19:19 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612717
Content-Length: 0
CSeq: 101 BYE

Router#
```

From the other side of the call, the debug output is as follows:

```
Router#debug ccsip message

SIP Call messages tracing is enabled
Router#

*Mar  8 17:45:12: Received:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Cisco-Guid: 2881152943-2184249568-0-483551624
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427554
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
```

```
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  8 17:45:12: Sent:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  8 17:45:12: Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  8 17:45:14: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  8 17:45:14: Received:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
```

```
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 138
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  8 17:45:17: Sent:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:45:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612717
CSeq: 101 BYE
Content-Length: 0



*Mar  8 17:45:17: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:19:19 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612717
Content-Length: 0
CSeq: 101 BYE
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Displays all SIP Service Provider Interface (SPI) call tracingand traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Displays SIP SPI errors and traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip states

To show all SIP SPI state tracing, use the **debug ccsip states** EXEC command.

debug ccsip states

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

This command traces the state machine changes of SIP SPI and displays the state transitions.

Examples

The following is sample output for the **debug ccsip states** command.

```
Router# debug ccsip states

SIP Call states tracing is enabled
Router#

*Jan 2 18:34:37.793:0x6220C634 :State change from (STATE_NONE, SUBSTATE_NONE) to
(STATE_IDLE, SUBSTATE_NONE)
*Jan 2 18:34:37.797:0x6220C634 :State change from (STATE_IDLE, SUBSTATE_NONE) to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Jan 2 18:34:37.797:0x6220C634 :State change from (STATE_IDLE, SUBSTATE_CONNECTING) to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Jan 2 18:34:37.801:0x6220C634 :State change from (STATE_IDLE, SUBSTATE_CONNECTING) to
(STATE_SENT_INVITE, SUBSTATE_NONE)
*Jan 2 18:34:37.809:0x6220C634 :State change from (STATE_SENT_INVITE, SUBSTATE_NONE) to
(STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
*Jan 2 18:34:37.853:0x6220C634 :State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_PROCEEDING) to (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
*Jan 2 18:34:38.261:0x6220C634 :State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_ALERTING) to (STATE_ACTIVE, SUBSTATE_NONE)
*Jan 2 18:35:09.860:0x6220C634 :State change from (STATE_ACTIVE, SUBSTATE_NONE) to
(STATE_DISCONNECTING, SUBSTATE_NONE)
*Jan 2 18:35:09.868:0x6220C634 :State change from (STATE_DISCONNECTING, SUBSTATE_NONE) to
(STATE_DEAD, SUBSTATE_NONE)
*Jan 2 18:28:38.404: Queued event from SIP SPI :SIPSPI_EV_CLOSE_CONNECTION
```

PSTN Cause Code and SIP Event Mappings

Table 27 lists the PSTN cause codes that can be sent as an ISDN cause information element (IE) and the corresponding SIP event for each.

Table 27 PSTN Cause Code to SIP Event Mappings

PSTN Cause Code	Description	SIP Event
1	Unallocated number	410 Gone
3	No route to destination	404 Not found
16	Normal call clearing	BYE
17	User busy	486 Busy here
18	No user responding	480 Temporarily unavailable
19	No answer from the user	
21	Call rejected	603 Decline
22	Number changed	301Moved temporarily
27	Destination out of order	404 Not found
28	Address incomplete	484 Address incomplete
29	Facility rejected	501 Not implemented
31	Normal unspecified	404 Not found
34	No circuit available	503 Service unavailable
38	Network out of order	
41	Temporary failure	
42	Switching equipment congestion	
44	Requested channel not available	
47	Resource unavailable	
55	Incoming class barred within CUG	603 Decline
57	Bearer capability not authorized	501 Not implemented
58	Bearer capability not available	
63	Service or option unavailable	503 Service unavailable
65	Bearer cap not implemented	501 Not implemented
79	Service or option not implemented	
87	User not a member of CUG	603 Decline
88	Incompatible destination	400 Bad request
95	Invalid message	
102	Recover on timer expiry	408 Request timeout
111	Protocol error	400 Bad request
127	Interworking unspecified	500 Internal server error
Any code other than those listed		500 Internal server error

Table 28 lists the SIP events and the corresponding PSTN cause codes for each.

Table 28 SIP Event to PSTN Cause Code Mapping

SIP Event	PSTN Cause Code	Description
400 Bad request	127	Interworking
401 Unauthorized	57	Bearer cap not authorized
402 Payment required	21	Call rejected
403 Forbidden	57	Bearer cap not authorized
404 Not found	1	Unallocated number
405 Method not allowed	127	Interworking
406 Not acceptable		
407 Proxy authentication required	21	Call rejected
408 Request timeout	102	Recover on timer expiry
409 Conflict	41	Temporary failure
410 Gone	1	Unallocated number
411 Length required	127	Interworking
413 Request entity too long		
414 Request URI too long		
415 Unsupported media type	79	Service or option not available
420 Bad extension	127	Interworking
480 Temporarily unavailable	18	No user response
481 Call leg does not exist	127	Interworking
482 Loop detected		
483 Too many hops		
484 Address incomplete	28	Address incomplete
485 Address ambiguous	1	Unallocated number
486 Busy here	17	User busy
500 Internal server error	41	Temporary failure
501 Not implemented	79	Service or option not implemented
502 Bad gateway	38	Network out of order
503 Service unavailable	63	Service or option not available
504 Gateway timeout	102	Recover on timer expiry
505 Version not implemented	127	Interworking
600 Busy everywhere	17	User busy
603 Decline	21	Call rejected
604 Does not exist anywhere	1	Unallocated number
606 Not acceptable	58	Bearer cap not available

debug ccswvoice vofr-debug

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-debug** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-debug

no debug ccswvoice vofr-debug

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

This command does not apply to the Cisco MC3810 networking device.

This command should be used when attempting to troubleshoot a Voice over Frame Relay (VoFR) call that uses the "cisco-switched" session protocol. It provides the same information as the **debug ccswvoice vofr-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswvoice vofr-debug** command:

```
Router# debug ccswvoice vofr-debug

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 03:02:08.719:ccswvofr_bridge_drop:dropping bridge calls src 17 dst 16 dlci 100
      cid 9 state ACTIVE
*Mar  1 03:02:08.727:ccswvofr:callID 17 dlci 100 cid 9 state ACTIVE event O/G REL
*Mar  1 03:02:08.735:ccswvofr:callID 17 dlci 100 cid 9 state RELEASE event I/C RELCOMP
*Mar  1 03:02:08.735:ccswvofr_store_call_history_entry:cause=22 tcause=22
      cause_text=no circuit.
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 03:03:22.651:ccswvofr:callID 23 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 03:03:22.651:ccswvofr_out_callinit_setup:callID 23 using dlci 100 cid 10
*Mar  1 03:03:22.659:ccswvofr:callID 23 dlci 100 cid 10 state O/G INIT event I/C PROC
*Mar  1 03:03:22.667:ccswvofr:callID 23 dlci 100 cid 10 state O/G PROC event I/C CONN
ccfrf11_caps_ind:codec(preferred) = 0
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug frame-relay fragment	Displays information related to Frame Relay fragmentation on a PVC.

Command	Description
debug voice vofr	Displays Cisco trunk and FRF.11 trunk call setup attempts and displays which dial peer is used in the call setup.
debug vpm error	Displays the behavior of the Holst state machine.
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Displays all SIP Service Provider Interface (SPI) call tracing and traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Displays SIP SPI errors and traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.

debug ccswvoice vofr-session

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-session** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-session

no debug ccswvoice vofr-session

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

This command does not apply to the Cisco MC3810 networking device.

This command can be used to show the state transitions of the cisco-switched-vofr state machine as a call is processed. It should be used when attempting to troubleshoot a Voice over Frame Relay (VoFR) call that uses the “cisco-switched” session protocol.

Examples

The following example shows sample output from the **debug ccswvoice vofr-session** command:

```
Router# debug ccswvoice vofr-session

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 02:58:13.203:ccswvofr:callID 14 dlci 100 cid 8 state ACTIVE event O/G REL
*Mar  1 02:58:13.215:ccswvofr:callID 14 dlci 100 cid 8 state RELEASE event I/C RELCOMP
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 02:59:46.551:ccswvofr:callID 17 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 02:59:46.559:ccswvofr:callID 17 dlci 100 cid 9 state O/G INIT event I/C PROC
*Mar  1 02:59:46.567:ccswvofr:callID 17 dlci 100 cid 9 state O/G PROC event I/C CONN
3640_vofr(config-voiceport)#
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccsip all	Displays the ccswvoice function calls during call setup and teardown.
debug voice vofr	Displays Cisco trunk and FRF.11 trunk call setup attempts and displays which dial peer is used in the call setup.
debug vpm error	Displays the behavior of the Holst state machine.
debug vtsp port	Displays the behavior of the VTSP state machine.

debug ccswvoice vo-debug

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice voo-debug** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice voatm-debug

no debug ccswvoice voatm-debug

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(1)MA	This command was introduced on the Cisco MC3810 networking device.
12.0(7)XK	This command was first supported on the Cisco 3600 series router.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command when attempting to troubleshoot a Vo call that uses the "cisco-switched" session protocol. This command provides the same information as the **debug ccswvoice vo-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswvoice vo-debug** command:

```
Router# debug ccswvoice vo-debug

2w2d: ccswvoice: callID 529927 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccswvoice_out_callinit_setup: callID 529927 using pvcid 1 cid 15
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G INIT event I/C PROC
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G PROC event I/C
ALERTccfrf11_caps_ind: codec(preferred) = 1

2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G ALERT event I/C CONN
2w2d: ccswvoice_bridge_drop: dropping bridge calls src 529927 dst 529926 pvcid 1 cid 15
state ACTIVE
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state ACTIVE event O/G REL
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state RELEASE event I/C RELCOMP
2w2d: ccswvo_store_call_history_entry: cause=10 tcause=10    cause_text=normal call
clearing.
```

Related Commands

Command	Description
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.

debug ccswvoice vo-session

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vo-session** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vo-session

no debug ccswvoice vo-session

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(1)MA	This command was introduced on the Cisco MC3810 networking device.
12.0(7)XK	This command was first supported on the Cisco 3600 series router.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command to show the state transitions of the cisco-switched-vo state machine as a call is processed. This command should be used when attempting to troubleshoot a Vo call that uses the "cisco-switched" session protocol.

Examples

The following example shows sample output from the **debug ccswvoice vo-session** command:

```
Router# debug ccswvoice vo-session

2w2d: ccswvoice: callID 529919 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G INIT event I/C PROC
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G PROC event I/C ALERT
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G ALERT event I/C CONN
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state ACTIVE event O/G REL
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state RELEASE event I/C RELCOMP
```

Related Commands

Command	Description
debug call-mgmt	Displays the ccswvoice function calls during call setup and teardown.

debug ccswvoice vofr-debug

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-debug** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-debug

no debug ccswvoice vofr-debug

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced on the Cisco 2600 and Cisco 3600 series routers.
12.0(4)T	This command was integrated into Cisco IOS Release 12.0(4)T.
12.0(7)XK	This command was first supported on the Cisco MC3810 networking device.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command when troubleshooting a VoFR call that uses the “cisco-switched” session protocol. This command provides the same information as the **debug ccswvoice vofr-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswvoice vofr-debug** command:

```
Router# debug ccswvoice vofr-debug

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 03:02:08.719:ccswvofr_bridge_drop:dropping bridge calls src 17 dst 16 dlci 100
      cid 9 state ACTIVE
*Mar  1 03:02:08.727:ccswvofr:callID 17 dlci 100 cid 9 state ACTIVE event O/G REL
*Mar  1 03:02:08.735:ccswvofr:callID 17 dlci 100 cid 9 state RELEASE event I/C RELCOMP
*Mar  1 03:02:08.735:ccswvofr_store_call_history_entry:cause=22 tcause=22
      cause_text=no circuit.
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 03:03:22.651:ccswvofr:callID 23 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 03:03:22.651:ccswvofr_out_callinit_setup:callID 23 using dlci 100 cid 10
*Mar  1 03:03:22.659:ccswvofr:callID 23 dlci 100 cid 10 state O/G INIT event I/C PROC
*Mar  1 03:03:22.667:ccswvofr:callID 23 dlci 100 cid 10 state O/G PROC event I/C CONN
ccfrf11_caps_ind:codec(preferred) = 0
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug vtsp session	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug ccswvoice vofr-session

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-session** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-session

no debug ccswvoice vofr-session

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced on the Cisco 2600 and Cisco 3600 series routers.
12.0(4)T	This command was integrated into Cisco IOS Release 12.0(4)T.
12.0(7)XK	This command was first supported on the Cisco MC3810 networking device.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command to show the state transitions of the cisco-switched-vofr state machine as a call is processed, and when attempting to troubleshoot a VoFR call that uses the "cisco-switched" session protocol.

Examples

The following example shows sample output from the **debug ccswvoice vofr-session** command:

```
Router# debug ccswvoice vofr-session

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 02:58:13.203:ccswvofr:callID 14 dlci 100 cid 8 state ACTIVE event O/G REL
*Mar  1 02:58:13.215:ccswvofr:callID 14 dlci 100 cid 8 state RELEASE event I/C RELCOMP
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 02:59:46.551:ccswvofr:callID 17 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 02:59:46.559:ccswvofr:callID 17 dlci 100 cid 9 state O/G INIT event I/C PROC
*Mar  1 02:59:46.567:ccswvofr:callID 17 dlci 100 cid 9 state O/G PROC event I/C CONN
3640_vofr(config-voiceport)#
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug call-mgmt	Displays the ccswvoice function calls during call setup and teardown.
debug vtsp session	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug cdapi

To display information about the call distributor application programming interface (CDAPI), use the **debug cdapi** privileged EXEC command.

debug cdapi {*detail* | *events*}

Syntax Description

detail	Displays when applications register or unregister with CDAPI, when calls are added or deleted from the CDAPI routing table, and when CDAPI messages are created and freed. It is useful for determining if messages are being lost (or not freed) and the size of the raw messages passed between CDAPI and applications so that you can check that the correct number of bytes is being passed.
events	Displays the events passing between CDAPI and an application or signalling stack. This debug is useful for determining if certain ISDN messages are not being received by an application and if calls are not being directed to an application.

Defaults

Disabled

Command History

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

Examples

The following example shows output for the **debug cdapi** command:

```
003909 ISDN Se123 RX <-  SETUP pd = 8  callref = 0x06BB
003909          Bearer Capability i = 0x9090A2
003909          Channel ID i = 0xA18381
003909          Facility i =
0x9FAA068001008201008B0100A1180202274C020100800F534341524C415454492D3530303733
003909          Progress Ind i = 0x8183 - Origination address is non-ISDN
003909          Calling Party Number i = 0xA1, '50073'
003909          Called Party Number i = 0xC1, '3450070'
003909 CDAPI Se123 TX -> CDAPI_MSG_CONNECT_IND to TSP CDAPI Application call = 0x24
003909        From Appl/Stack = ISDN
003909        Call Type = VOICE
003909        B Channel = 0
003909        Cause     = 0
003909        Calling Party Number = 50073
003909        Called Party Number = 3450070
003909 CDAPI Se123 TX -> CDAPI_MSG_CONNECT_RESP to ISDN call = 0x24
003909        From Appl/Stack = TSP CDAPI Application
003909        Call Type = VOICE
003909        B Channel = 0
003909        Cause     = 0
003909 CDAPI-ISDN Se123 RX <- CDAPI_MSG_CONNECT_RESP from TSP CDAPI Application call =
0x24
003909        Call Type = VOICE
003909        B Channel = 0
003909        Cause     = 0
```

```
003909 CDAPI Se123 TX -> CDAPI_MSG_SUBTYPE_CALL_PROC_REQ to ISDN call = 0x24
003909       From Appl/Stack = TSP CDAPI Application
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
003909 CDAPI-ISDN Se123 RX <- CDAPI_MSG_SUBTYPE_CALL_PROC_REQ from TSP CDAPI Application
call = 0x24
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
003909 ISDN Se123 TX ->  CALL_PROC pd = 8  callref = 0x86BB
003909         Channel ID i = 0xA98381
```

Related Commands

Command	Description
debug cdapi	Displays information about the CDAPI.
debug voip rawmsg	Displays the raw message owner, length, and pointer.

debug cdp

To enable debugging of the Cisco Discovery Protocol (CDP), use the **debug cdp** privileged EXEC command. The **no** form of this command disables debugging output.

debug cdp {**packets** | **adjacency** | **events**}

no debug cdp {**packets** | **adjacency** | **events**}

Syntax Description

packets	Enables packet-related debugging output.
adjacency	Enables adjacency-related debugging output.
events	Enables output related to error messages, such as detecting a bad checksum.

Usage Guidelines

Use **debug cdp** commands to display information about CDP packet activity, activity between CDP neighbors, and various CDP events.

Examples

The following is sample output from **debug cdp packets**, **debug cdp adjacency,** and **debug cdp events** commands:

```
Router# debug cdp packets

CDP packet info debugging is on
Router# debug cdp adjacency

CDP neighbor info debugging is on
Router# debug cdp events

CDP events debugging is on

CDP-PA: Packet sent out on Ethernet0
CDP-PA: Packet received from gray.cisco.com on interface Ethernet0

CDP-AD: Deleted table entry for violet.cisco.com, interface Ethernet0
CDP-AD: Interface Ethernet2 coming up

CDP-EV: Encapsulation on interface Serial2 failed
```

debug cdp ip

To enable debug output for the IP routing information that is carried and processed by the Cisco Discovery Protocol (CDP), use the **debug cdp ip** privileged EXEC command. The **no** form of this command disables debugging output.

debug cdp ip

no debug cdp ip

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CDP is a media- and protocol-independent device-discovery protocol that runs on all Cisco routers.

You can use the **debug cdp ip** command to determine the IP network prefixes CDP is advertising and whether CDP is correctly receiving this information from neighboring routers.

Use the **debug cdp ip** command with the **debug ip routing** command to debug problems that occur when on-demand routing (ODR) routes are not installed in the routing table at a hub router. You can also use the **debug cdp ip** command with the **debug cdp packet** and **debug cdp adjacency** commands along with encapsulation-specific debug commands to debug problems that occur in the receipt of CDP IP information.

Examples

The following is sample output from the **debug cdp ip** command. This example shows the transmission of IP-specific information in a CDP update. In this case, three network prefixes are being sent, each with a different network mask.

```
Router# debug cdp ip

CDP-IP: Writing prefix 172.1.69.232.112/28
CDP-IP: Writing prefix 172.19.89.0/24
CDP-IP: Writing prefix 11.0.0.0/8
```

In addition to these messages, you might see the following messages:

- This message indicates that CDP is attempting to install the prefix 172.16.1.0/24 into the IP routing table:

  ```
  CDP-IP: Updating prefix 172.16.1.0/24 in routing table
  ```

- This message indicates a protocol error occurred during an attempt to decode an incoming CDP packet:

  ```
  CDP-IP: IP TLV length (3) invalid
  ```

- This message indicates the receipt of the IP prefix 172.16.1.0/24 from a CDP neighbor connected via Ethernet interface 0/0. The neighbor IP address is 10.0.01.

  ```
  CDP-IP: Reading prefix 172.16.1.0/24 source 10.0.0.1 via Ethernet0/0
  ```

Related Commands

Command	Description
debug ip routing	Displays information on RIP routing table updates and route cache updates.

debug channel events

To display processing events on Cisco 7000 series routers that occur on the channel adapter interfaces of all installed adapters, use the **debug channel events** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug channel events

no debug channel events

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

This command displays CMCC adapter events that occur on the CIP or CPA and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the **debug channel events** command does not return any information. If the command generates numerous messages, the messages can indicate the possible source of the problems. To observe the statistic message (cip_love_letter) sent every 10 seconds, use the **debug channel love** command.

When configuring or making changes to a router or interface that supports IBM channel attach, enable the **debug channel events** command. Doing so alerts you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

Examples

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

```
Router# debug channel events

Channel3/0: cip_reset(), state administratively down
Channel3/0: cip_reset(), state up
Channel3/0: sending nodeid
Channel3/0: sending command for vc 0, CLAW path C700, device C0
```

The following line indicates that the CIP is being reset to an administrative down state:

```
Channel3/0: cip_reset(), state administratively down
```

The following line indicates that the CIP is being reset to an administrative up state:

```
Channel3/0: cip_reset(), state up
```

The following line indicates that the node ID is being sent to the CIP. This information is the same as the "Local Node" information under the s**how extended channel** *slot*/*port* **subchannels** command. The CIP needs to send this information to the host mainframe.

```
Channel3/0: sending nodeid
```

The following line indicates that a CLAW subchannel command is being sent from the RP to the CIP. The value vc 0 indicates that the CIP will use virtual circuit number 0 with this device. The virtual circuit number also shows up when you use the **debug channel packets** command.

```
Channel3/0: sending command for vc 0, CLAW path C700, device C0
```

The following is a sample output that is generated by the **debug channel events** command when a CMPC+ IP TG connection is activated with the host:

```
1d05h:Channel4/2:Received route UP  for tg (768)
1d05h:Adding STATIC ROUTE for vc:768
```

The following is a sample output from the **debug channel events** command when a CMPC+ IP TG connection is deactivated:

```
1d05h:Channel4/2:Received route DOWN  for tg (768)
1d05h:Deleting STATIC ROUTE for vc:768
```

Related Commands

Command	Description
debug channel ilan	Displays CIP love letter events.
debug channel packets	Displays per-packet debugging output.

debug channel ilan

To display messages relating to configuration and bridging using CMCC internal LANs and to help debug source-route bridging (SRB) problems related to CMCC internal LANs, use the **debug channel ilan** privileged EXEC command. The **no** form of this command disables debugging output.

debug channel ilan

no debug channel ilan

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.0(3)	This command was introduced.

Usage Guidelines

The **debug channel ilan** command displays events related to CMCC internal LANs. This command is useful for debugging problems associated with CMCC internal LAN configuration. It is also useful for debugging problems related to SRB packet flows through internal LANs.

Examples

The following sample output is from the **debug channel ilan** command:

```
Router# debug channel ilan

Channel internal LANs debugging is on
```

The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the LLC end station in Cisco IOS software did not exist:

```
CIP ILAN(Channel3/2-Token): Packet dropped - NULL LLC
```

The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the CMCC had not yet acknowledged the internal MAC adapter configuration command:

```
Channel3/2: ILAN Token-Ring 3 - CIP internal MAC adapter not acknowledged
DMAC(4000.7000.0001) SMAC(0c00.8123.0023)
```

Related Commands

Command	Description
debug source bridge	Displays information about packets and frames transferred across a source-route bridge.
debug channel events	Displays processing that occurs on the channel adapter interfaces of all installed adapters.

debug channel love

To display Channel Interface Processor (CIP) love letter events, use the **debug channel love** privileged EXEC command. The **no** form of this command disables debugging output.

debug channel love

no debug channel love

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command displays CIP events that occur on the CIP interface processor and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the **debug channel love** command returns a statistic message (cip_love_letter) that is sent every 10 seconds. This command is valid for the Cisco 7000 series routers only.

Examples

The following is sample output from the **debug channel love** command:

```
Router# debug channel love

Channel3/1: love letter received, bytes 3308
Channel3/0: love letter received, bytes 3336
cip_love_letter: received ll, but no cip_info
```

The following line indicates that data was received on the CIP:

```
Channel3/1: love letter received, bytes 3308
```

The following line indicates that the interface is enabled, but there is no configuration for it. It does not normally indicate a problem, just that the Route Processor (RP) got statistics from the CIP but has no place to store them.

```
cip_love_letter: received ll, but no cip_info
```

Related Commands

Command	Description
debug channel events	Displays processing that occur on the channel adapter interfaces of all installed adapters.
debug channel packets	Displays per-packet debugging output.

debug channel packets

To display per-packet debugging output, use the **debug channel packets** privileged EXEC command. The output reports information when a packet is received or a transmission is attempted. The **no** form of this command disables debugging output.

debug channel packets

no debug channel packets

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug channel packets** command displays all process-level Channel Interface Processor (CIP) packets for both outbound and inbound packets. You will need to disable fast switching and autonomous switching to obtain debugging output. This command is useful for determining whether packets are received or sent correctly.

This command is valid for the Cisco 7000 series routers only.

Examples

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

```
Router# debug channel packets

(Channel3/0)-out size = 104, vc = 0000, type = 0800, src 172.24.0.11, dst 172.24.1.58
(Channel3/0)-in size = 48, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
(Channel3/0)-in size = 48, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
(Channel3/0)-out size = 71, vc = 0000, type = 0800, src 172.24.15.197, dst 172.24.1.58
(Channel3/0)-in size = 44, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
```

Table 29 describes the significant fields in the display.

Table 29 debug channel packets Field Descriptions

Field	Description
(Channel3/0)	Interface slot and port.
in/out	“In” is a packet from the mainframe to the router. “Out” is a packet from the router to the mainframe.
size =	Number of bytes in the packet, including internal overhead.
vc =	Value from 0 to 511 that maps to the **claw** interface configuration command. This information is from the MAC layer.
type =	Encapsulation type in the MAC layer. The value 0800 indicates an IP datagram.
src	Origin, or source, of the packet, as opposed to the previous hop address.
dst	Destination of the packet, as opposed to the next hop address.

debug clns esis events

To display uncommon End System-to-Intermediate System (ES-IS) events, including previously unknown neighbors, neighbors that have aged out, and neighbors that have changed roles (ES-IS, for example), use the **debug clns esis events** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns esis events

no debug clns esis events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns esis events

ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```

The following line indicates that the router received a hello packet (ISH) from the IS at MAC address aa00.0400.2c05 on Ethernet interface 1. The hold time (or number of seconds to consider this packet valid before deleting it) for this packet is 30 seconds.

```
ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
```

The following line indicates that the router received a hello packet (ESH) from the ES at MAC address aa00.0400.9105 on the Ethernet interface 1. The hold time is 150 seconds.

```
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
```

The following line indicates that the router sent an IS hello packet on the Ethernet interface 0 to all ESs on the network. The network entity title (NET) address of the router is 49.0001.0400.AA00.6904.00; the hold time for this packet is 299 seconds; and the header length of this packet is 20 bytes.

```
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```

debug clns esis packets

To enable display information on End System-to-Intermediate System (ES-IS) packets that the router has received and sent, use the **debug clns esis packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns esis packets

no debug clns esis packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns esis packets

ES-IS: ISH sent to All ESs (Ethernet0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
ES-IS: ISH sent to All ESs (Ethernet1): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
ES-IS: ISH from aa00.0400.6408 (Ethernet0), HT 299
ES-IS: ISH sent to All ESs (Tunnel0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.O906.4023.00, HT 299, HLEN 34
IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
```

The following line indicates that the router has sent an IS hello packet on Ethernet interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Ethernet0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
```

The following line indicates that the router has sent an IS hello packet on Ethernet interface 1 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Ethernet1): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```

The following line indicates that the router received a hello packet on Ethernet interface 0 from an intermediate system, aa00.0400.6408. The hold time for this packet is 299 seconds.

```
ES-IS: ISH from aa00.0400.6408 (Ethernet0), HT 299
```

The following line indicates that the router has sent an IS hello packet on Tunnel interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Tunnel0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```

The following line indicates that on Ethernet interface 0, the router received a hello packet from an end system with an SNPA of 0000.0c00.bda8. The hold time for this packet is 300 seconds.

```
IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
```

debug clns events

To display CLNS events that are occurring at the router, use the **debug clns events** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns events

no debug clns events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns events

CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00
         via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
CLNS: Forwarding packet size 117
      from 39.0001.2222.2222.2222.00
      to 49.0002.0001.AAAA.AAAA.AAAA.00
      via 49.0002 (Ethernet3 0000.0c00.b5a3)
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18,
      redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```

The following line indicates that the router received an echo PDU on Ethernet interface 3 from source network service access point (NSAP) 39.0001.2222.2222.2222.00. The exclamation point at the end of the line has no significance.

```
CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
```

The following lines indicate that the router at source NSAP 39.0001.3333.3333.3333.00 is sending a CLNS echo packet to destination NSAP 39.0001.2222.2222.2222.00 via an IS with system ID 2222.2222.2222. The packet is being sent on Ethernet interface 3, with a MAC address of 0000.0c00.3a18.

```
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00
         via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
```

The following lines indicate that a CLNS echo packet 117 bytes in size is being sent from source NSAP 39.0001.2222.2222.2222.00 to destination NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 via the router at NSAP 49.0002. The packet is being forwarded on the Ethernet interface 3, with a MAC address of 0000.0c00.b5a3.

```
CLNS: Forwarding packet size 117
      from 39.0001.2222.2222.2222.00
      to 49.0002.0001.AAAA.AAAA.AAAA.00
      via 49.0002 (Ethernet3 0000.0c00.b5a3)
```

The following lines indicate that the router sent a redirect packet on the Ethernet interface 3 to the NSAP 39.0001.2222.2222.2222.00 at MAC address 0000.0c00.3a18 to indicate that NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 can be reached at MAC address 0000.0c00.b5a3.

```
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18,
      redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```

debug clns igrp packets

To display debugging information on all ISO-IGRP routing activity, use the **debug clns igrp packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns igrp packets

no debug clns igrp packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns igrp packets

ISO-IGRP: Hello sent on Ethernet3 for DOMAIN_green1
ISO-IGRP: Received hello from 39.0001.3333.3333.3333.00, (Ethernet3), ht 51
ISO-IGRP: Originating level 1 periodic update
ISO-IGRP: Advertise dest: 2222.2222.2222
ISO-IGRP: Sending update on interface: Ethernet3
ISO-IGRP: Originating level 2 periodic update
ISO-IGRP: Advertise dest: 0001
ISO-IGRP: Sending update on interface: Ethernet3
ISO-IGRP: Received update from 3333.3333.3333 (Ethernet3)
ISO-IGRP: Opcode: area
ISO-IGRP: Received level 2 adv for 0001 metric 1100
ISO-IGRP: Opcode: station
ISO-IGRP: Received level 1 adv for 3333.3333.3333 metric 1100
```

The following line indicates that the router is sending a hello packet to advertise its existence in the DOMAIN_green1 domain:

```
ISO-IGRP: Hello sent on Ethernet3 for DOMAIN_green1
```

The following line indicates that the router received a hello packet from a certain network service access point (NSAP) on Ethernet interface 3. The hold time for this information is 51 seconds.

```
ISO-IGRP: Received hello from 39.0001.3333.3333.3333.00, (Ethernet3), ht 51
```

The following lines indicate that the router is generating a Level 1 update to advertise reachability to destination NSAP 2222.2222.2222 and that it is sending that update to all systems that can be reached through Ethernet interface 3:

```
ISO-IGRP: Originating level 1 periodic update
ISO-IGRP: Advertise dest: 2222.2222.2222
ISO-IGRP: Sending update on interface: Ethernet3
```

The following lines indicate that the router is generating a Level 2 update to advertise reachability to destination area 1 and that it is sending that update to all systems that can be reached through Ethernet interface 3:

```
ISO-IGRP: Originating level 2 periodic update
ISO-IGRP: Advertise dest: 0001
ISO-IGRP: Sending update on interface: Ethernet3
```

The following lines indicate that the router received an update from NSAP 3333.3333.3333 on Ethernet interface 3. This update indicated the area that the router at this NSAP could reach.

```
ISO-IGRP: Received update from 3333.3333.3333 (Ethernet3)
ISO-IGRP: Opcode: area
```

The following lines indicate that the router received an update advertising that the source of that update can reach area 1 with a metric of 1100. A station opcode indicates that the update included system addresses.

```
ISO-IGRP: Received level 2 adv for 0001 metric 1100
ISO-IGRP: Opcode: station
```

debug clns packet

To display information about packet receipt and forwarding to the next interface, use the **debug clns packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns packet

no debug clns packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns packet

CLNS: Forwarding packet size 157
      from 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.00 STUPI-RBS
      to 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
CLNS: Echo PDU received on Ethernet0 from
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00!
CLNS: Sending from 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00 to
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

In the following lines, the first line indicates that a Connectionless Network Service (CLNS) packet of size 157 bytes is being forwarded. The second line indicates the network service access point (NSAP) and system name of the source of the packet. The third line indicates the destination NSAP for this packet. The fourth line indicates the next hop system ID, interface, and SNPA of the router interface used to forward this packet.

```
CLNS: Forwarding packet size 157
      from 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.00 STUPI-RBS
      to 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

In the following lines, the first line indicates that the router received an echo PDU on the specified interface from the source NSAP. The second line indicates which source NSAP is used to send a CLNS packet to the destination NSAP, as shown on the third line. The fourth line indicates the next hop system ID, interface, and SNPA of the router interface used to forward this packet.

```
CLNS: Echo PDU received on Ethernet0 from
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00!
CLNS: Sending from 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00 to
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

debug clns routing

To display debugging information for all Connectionless Network Service (CLNS) routing cache updates and activities involving the CLNS routing table, use the **debug clns routing** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns routing

no debug clns routing

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns routing

CLNS-RT: cache increment:17
CLNS-RT: Add 47.0023.0001.0000.0000.0003.0001 to prefix table, next hop 1920.3614.3002
CLNS-RT: Aging cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
CLNS-RT: Deleting cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
```

The following line indicates that a change to the routing table has resulted in an addition to the fast-switching cache:

```
CLNS-RT: cache increment:17
```

The following line indicates that a specific prefix route was added to the routing table, and indicates the next hop system ID to that prefix route. In other words, when the router receives a packet with the prefix 47.0023.0001.0000.0000.0003.0001 in the destination address of that packet, it forwards that packet to the router with the MAC address 1920.3614.3002.

```
CLNS-RT: Add 47.0023.0001.0000.0000.0003.0001 to prefix table, next hop 1920.3614.3002
```

The following lines indicate that the fast-switching cache entry for a certain network service access point (NSAP) has been invalidated and then deleted:

```
CLNS-RT: Aging cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
CLNS-RT: Deleting cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
```

debug cls message

To display information about Cisco Link Services (CLS) messages, use the **debug cls message** privileged EXEC command. The **no** form of this command disables debugging output.

debug cls message

no debug cls message

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug cls message** command displays the primitives (state), selector, header length, and data size.

Examples

The following is sample output from the **debug cls message** command. For example, CLS-->DLU indicates the direction of the flow that is described by the status. From CLS to DLU, a request was established to the connection endpoint. The header length is 48 bytes, and the data size is 104 bytes.

```
Router# debug cls message

(FRAS Daemon:CLS-->DLU):
     ID_STN.Ind to uSAP: 0x607044C4 sel: LLC hlen: 40, dlen: 54
(FRAS Daemon:CLS-->DLU):
     ID_STN.Ind to uSAP: 0x6071B054 sel: LLC hlen: 40, dlen: 46
(FRAS Daemon:DLU-->SAP):
     REQ_OPNSTN.Req to pSAP: 0x608021F4 sel: LLC hlen: 48, dlen: 104
(FRAS Daemon:CLS-->DLU):
     REQ_OPNSTN.Cfm(NO_REMOTE_STN) to uCEP: 0x607FFE84 sel: LLC hlen: 48, dlen: 104
```

The status possibilities include the following: enabled, disabled, request open station, open station, close station, activate SA, deactivate SAP, XID, XID station, connect station, signal station, connect, disconnect, connected, data, flow, unnumbered data, modify SAP, test, activate ring, deactivate ring, test station, and unnumbered data station.

Related Commands

Command	Description
debug fras error	Displays information about FRAS protocol errors.
debug fras message	Displays general information about FRAS messages.
debug fras state	Displays information about FRAS data-link control state changes.

debug cls vdlc

To display information about Cisco Link Services (CLS) Virtual Data Link Control (VDLC), use the **debug cls vdlc** privileged EXEC command. The **no** form of this command disables debugging output.

debug cls vdlc

no debug cls vdlc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug cls message** command displays primitive state transitions, selector, and source and destination MAC and service access points (SAPs).

Also use the **show cls** command to display additional information on CLS VDLC.

Caution Use the **debug cls vdlc** command with caution because it can generate a substantial amount of output.

Examples

The following messages are sample output from the **debug cls vdlc** command. In the following scenario, the SNA service point—also called *native service point (NSP)*—is setting up two connections through VDLC and data-link switching (DLSw): one from NSP to VDLC and one from DLSw to VDLC. VDLC joins the two.

The NSP initiates a connection from 4000.05d2.0001 as follows:

```
VDLC: Req Open Stn Req PSap 0x7ACE00, port 0x79DF98
     4000.05d2.0001(0C)->4000.1060.1000(04)
```

In the next message, VDLC sends a test station request to DLSw for destination address 4000.1060.1000.

```
VDLC: Send UFrame E3: 4000.05d2.0001(0C)->4000.1060.1000(00)
```

In the next two messages, DLSw replies with test station response, and NSP goes to a half-open state. NSP is waiting for the DLSw connection to VDLC.

```
VDLC: Sap to Sap TEST_STN_RSP VSap 0x7B68C0 4000.1060.1000(00)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPENING->VDLC_HALF_OPEN
```

The NSP sends an exchange identification (XID) and changes state as follows:

```
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_HALF_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to SAP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04) via bridging SAP (DLSw)
```

In the next several messages, DLSw initiates its connection, which matches the half-open connection with NSP:

```
VDLC: Req Open Stn Req PSap 0x7B68C0, port 0x7992A0
     4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: two-way connection established
VDLC: 4000.1060.1000(04)->4000.05d2.0001(0C): VDLC_IDLE->VDLC_OPEN
```

In the following messages, DLSw sends an XID response, and NSP's connection goes from the state XID Response Pending to Open. The XID exchange follows:

```
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
```

When DLSw is ready to connect, the front-end processor (FEP) sends a set asynchronous balanced mode extended (SABME) command as follows:

```
VDLC: CEP to CEP CONNECT_REQ 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
```

In the following messages, NSP accepts the connection and sends an unnumbered acknowledgment (UA) to the FEP:

```
VDLC: CEP to CEP CONNECT_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: FlowReq QUENCH OFF 4000.1060.1000(04)->4000.05d2.0001(0C)
```

The following messages show the data flow:

```
VDLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)
.
.
.
VDLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)
```

Related Commands

Command	Description
debug cls message	Displays information about CLS messages

debug compress

To debug compression, enter the **debug compress** privileged EXEC configuration command. To disable debugging output, use the **no** form of this command.

debug compress

no debug compress

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
10.0	This command was introduced.

Usage Guidelines

Use this command to display output from the compression and decompression configuration you made. Live traffic must be configured through the Cisco 2600 access router with a data compression Advanced Interface Module (AIM) installed for this command to work.

Examples

The following example is output from the **debug compress** command, which shows that compression is taking place on a Cisco 2600 access router using data compression AIM hardware compression is configured correctly:

```
Router# debug compress

COMPRESS debugging is on
Router#compr-in:pak:0x810C6B10 npart:0 size:103
pak:0x810C6B10 start:0x02406BD4 size:103 npart:0
compr-out:pak:0x8118C8B8 stat:0x00000000 npart:1 size:71 lcb:0xED
pak:0x8118C8B8 start:0x0259CD3E size:71 npart:1
    mp:0x8118A980 start:0x0259CD3E size:71

decmp-in:pak:0x81128B78 start:0x0255AF44 size:42 npart:1 hdr:0xC035
pak:0x81128B78 start:0x0255AF44 size:42 npart:1
    mp:0x81174480 start:0x0255AF44 size:42
decmp-out:pak:0x8118C8B8 start:0x025B2C42 size:55 npart:1 stat:0
pak:0x8118C8B8 start:0x025B2C42 size:55 npart:1
    mp:0x8118B700 start:0x025B2C42 size:55
```

Table 30 describes the significant fields in the display.

Table 30 debug compress Field Descriptions

Field	Description
compr-in	Indicates that a packet needs to be compressed.
compr-out	Indicates completion of compression of packet.

Table 30 debug compress Field Descriptions (continued)

Field	Description
decmp-in	Indicates receipt of a compressed packet that needs to be decompressed.
decmp-out	Indicates completion of decompression of a packet.
pak:0x810C6B10	Provides the address in memory of a software structure that describes the compressed packet.
start:0x02406BD4 size:103 npart:0	The "npart:0" indicates that the packet is contained in a single, contiguous area of memory. The start address of the packet is 0x02406bd4 and the size of the packet is 103.
start:0x0259CD3E size:71 npart:1	The "npart:1" indicates that the packet is contained in 1 or more regions of memory. The start address of the packet is 0x0259CD3E and the size of the packet is 71.
mp:0x8118A980 start:0x0259CD3e size:71	Describes one of these regions of memory.
mp:0x8118A980	Provides the address of a structure describing this region.
start 0x0259CD3E	Provides the address of the start of this region.

Related Commands

Command	Description
debug frame-relay	Displays debugging information about the packets that are received on a Frame Relay interface.
debug ppp	Displays information on traffic and exchanges in an internetwork implementing the PPP.
show compress	Displays compression statistics.
show diag	Displays hardware information including DRAM, SRAM, and the revision-level information on the line card.

debug condition

To limit output for some debugging commands based on specified conditions, use the **debug condition** privileged EXEC command. The **no** form of this command removes the specified condition.

debug condition {**username** *username* | **called** *dial-string* | **caller** *dial-string*}

no debug condition {*condition-id* | **all**}

Syntax Description

username *username*	Generates debugging messages for interfaces with the specified username.
called *dial-string*	Generates debugging messages for interfaces with the called party number.
caller *dial-string*	Generates debugging messages for interfaces with the calling party number.
condition-id	Removes the condition indicated.
all	Removes all debugging conditions, and conditions specified by the **debug condition interface** command. Use this keyword to disable conditional debugging and reenable debugging for all interfaces.

Defaults

All debugging messages for enabled protocol-specific **debug** commands are generated.

Usage Guidelines

Use the **debug condition** command to restrict the debug output for some commands. If any **debug condition** commands are enabled, output is only generated for interfaces associated with the specified username, called party number, or calling party number. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

The **no** form of this command removes the debug condition specified by the condition identifier. The condition identifier is displayed after you enter a **debug condition** command or in the output of the **show debug condition** command. If the last condition is removed, debugging output resumes for all interfaces. You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the **debug condition** command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions. The commands that are affected by the **debug condition** commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the **debug condition** command:

- **debug aaa** {**accounting** | **authorization** | **authentication**}
- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Examples

In the following example, the router displays debugging messages only for interfaces that use a username of fred. The condition identifier displayed after the command is entered identifies this particular condition.

```
Router# debug condition username fred

Condition 1 set
```

Related Commands

Command	Description
debug condition interface	Limits output for some debugging commands based on the interfaces.

debug condition interface

To limit output for some debugging commands based on the interface, use the **debug condition interface** privileged EXEC command. The **no** form of this command removes the interface condition and resets the interface so that it must be triggered by a condition.

debug condition interface {*interface* | **all**}

no debug condition interface {*interface* | **all**}

Syntax Description

interface	The interface type and number.
all	Displays all interfaces.

Defaults

All debug messages for enabled debugging commands are displayed.

Usage Guidelines

Use this command to restrict the debug output for some commands to output based on its related interface. When you enter this command, debugging output is turned off for all interfaces except the specified interface. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

The **no** form of the command has two functions:

- It disables the **debug condition interface** command for the specified interface. Output is no longer generated for the interface, assuming that the interface meets no other conditions. If the interface meets other active conditions, as set by another **debug condition** command, debugging output will still be generated for the interface.
- The command also resets the debugging trigger on the interface. If some other **debug condition** command has been enabled, this command resets the trigger on the interface. Output is stopped for that interface until the condition is met on the interface.

You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the **debug condition** command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions. The commands that are affected by the **debug condition** commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the **debug condition** command:

- **debug aaa** {**accounting** | **authorization** | **authentication**}
- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Examples

In this example, only **debug** command output related to serial interface 1 is displayed. The condition identifier for this command is 1.

```
Router# debug condition interface serial1
```

```
Condition 1 set
```

Related Commands

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

debug confmodem

To display information associated with the discovery and configuration of the modem attached to the router, use the **debug confmodem** privileged EXEC command. The **no** form of this command disables debugging output.

debug confmodem

no debug confmodem

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug confmodem** command is used in debugging configurations that use the **modem autoconfig** command.

Examples

The following is sample output from the **debug confmodem** command. In the first three lines, the router is searching for a speed at which it can communicate with the modem. The remaining lines show the actual sending of the modem command.

```
Router# debug confmodem

TTY4:detection speed(115200) response ------
TTY4:detection speed(57600) response ------
TTY4:detection speed(38400) response ---OK---
TTY4:Modem command: --AT&F&C1&D2S180=3S190=1S0=1--
TTY4: Modem configuration succeeded
TTY4: Done with modem configuration
```

debug cops

To display a one-line summary of each COPS message sent from and received by the router, use the **debug cops** privileged EXEC command. Use the **no** form of this command to disable the debug output.

debug cops [**detail**]

no debug cops [**detail**]

Syntax Description

detail	(Optional) Displays additional debug information, including the contents of COPS and RSVP messages.

Defaults

COPS process debugging is not enabled.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

To generate a complete record of the policy process, enter this command and, after entering a carriage return, enter the additional command **debug ip rsvp policy**.

Examples

This first example displays the one-line COPS message summaries, as the router goes through six different events.

```
Router# debug cops

COPS debugging is on
```

Event 1

The router becomes configured to communicate with a policy server:

```
Router# configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip rsvp policy cops servers 2.0.0.1
Router(config)#
15:13:45:COPS: Opened TCP connection to 2.0.0.1/3288
15:13:45:COPS: ** SENDING MESSAGE **
15:13:45:COPS OPN message, Client-type:1, Length:28. Handle:[NONE]
15:13:45:COPS: ** RECEIVED MESSAGE **
15:13:45:COPS CAT message, Client-type:1, Length:16. Handle:[NONE]
Router(config)#
```

Event 2

The router receives a PATH message:

```
15:13:53:COPS:** SENDING MESSAGE **
```

```
15:13:53:COPS REQ message, Client-type:1, Length:216. Handle:[ 00 00 04 01]
15:13:53:COPS:** RECEIVED MESSAGE **
15:13:53:COPS DEC message, Client-type:1, Length:104. Handle:[ 00 00 04 01]
Router(config)#
```

Event 3

The router receives a unicast FF RESV message:

```
15:14:00:COPS:** SENDING MESSAGE **
15:14:00:COPS REQ message, Client-type:1, Length:148. Handle:[ 00 00 05 01]
15:14:00:COPS:** RECEIVED MESSAGE **
15:14:00:COPS DEC message, Client-type:1, Length:64. Handle:[ 00 00 05 01]
15:14:00:COPS:** SENDING MESSAGE **
15:14:00:COPS RPT message, Client-type:1, Length:24. Handle:[ 00 00 05 01]
Router(config)#
```

Event 4

The router receives a RESV tear:

```
15:14:06:COPS:** SENDING MESSAGE **
15:14:06:COPS DRQ message, Client-type:1, Length:24. Handle:[ 00 00 05 01]
Router(config)#
```

Event 5

The router receives a PATH tear:

```
15:14:11:COPS:** SENDING MESSAGE **
15:14:11:COPS DRQ message, Client-type:1, Length:24. Handle:[ 00 00 04 01]
Router(config)#
```

Event 6

The router gets configured to cease communicating with the policy server:

```
Router(config)# no ip rsvp policy cops servers
15:14:23:COPS:** SENDING MESSAGE **
15:14:23:COPS CC message, Client-type:1, Length:16. Handle:[NONE]
15:14:23:COPS:Closed TCP connection to 2.0.0.1/3288
Router(config)#
```

This second example uses the **detail** keyword to display the contents of the COPS and RSVP messages, and additional debugging information:

```
Router# debug cops detail

COPS debugging is on

02:13:29:COPS:** SENDING MESSAGE **
    COPS HEADER:Version 1, Flags 0, Opcode 1 (REQ), Client-type:1, Length:216
    HANDLE (1/1) object. Length:8.    00 00 21 01
    CONTEXT (2/1) object. Length:8.   R-type:5.     M-type:1
    IN_IF (3/1) object. Length:12.   Address:10.1.2.1.    If_index:4
    OUT_IF (4/1) object. Length:12.   Address:10.33.0.1.    If_index:3
    CLIENT SI (9/1) object. Length:168.   CSI data:
02:13:29: SESSION              type 1 length 12:
02:13:29:      Destination 10.33.0.1, Protocol_Id 17, Don't Police , DstPort 44
```

```
02:13:29: HOP                  type 1 length 12:0A010201
02:13:29:                                       :00000000
02:13:29: TIME_VALUES          type 1 length 8 :00007530
02:13:29: SENDER_TEMPLATE      type 1 length 12:
02:13:29:      Source 10.31.0.1, udp_source_port 44
02:13:29: SENDER_TSPEC         type 2 length 36:
02:13:29:      version=0, length in words=7
02:13:29:      Token bucket fragment (service_id=1, length=6 words
02:13:29:          parameter id=127, flags=0, parameter length=5
02:13:29:          average rate=1250 bytes/sec, burst depth=10000 bytes
02:13:29:          peak rate   =1250000 bytes/sec
02:13:29:          min unit=0 bytes, max unit=1514 bytes
02:13:29: ADSPEC               type 2 length 84:
02:13:29: version=0  length in words=19
02:13:29: General Parameters  break bit=0  service length=8
02:13:29:                                            IS Hops:1
02:13:29:                 Minimum Path Bandwidth (bytes/sec):1250000
02:13:29:                        Path Latency (microseconds):0
02:13:29:                                           Path MTU:1500
02:13:29: Guaranteed Service  break bit=0  service length=8
02:13:29:                          Path Delay (microseconds):192000
02:13:29:                         Path Jitter (microseconds):1200
02:13:29:            Path delay since shaping (microseconds):192000
02:13:29:           Path Jitter since shaping (microseconds):1200
02:13:29: Controlled Load Service  break bit=0  service length=0
02:13:29:COPS:Sent 216 bytes on socket,
02:13:29:COPS:Message event!
02:13:29:COPS:State of TCP is 4
02:13:29:In read function
02:13:29:COPS:Read block of 96 bytes, num=104 (len=104)
02:13:29:COPS:** RECEIVED MESSAGE **
    COPS HEADER:Version 1, Flags 1, Opcode 2 (DEC), Client-type:1, Length:104
    HANDLE (1/1) object. Length:8.    00 00 21 01
    CONTEXT (2/1) object. Length:8.   R-type:1.     M-type:1
    DECISION (6/1) object. Length:8.   COMMAND cmd:1, flags:0
    DECISION (6/3) object. Length:56.   REPLACEMENT  00 10 0E 01 61 62 63 64 65 66 67
    68 69 6A 6B 6C 00 24 0C 02 00
    00 00 07 01 00 00 06 7F 00 00 05 44 9C 40 00 46 1C 40 00 49 98
    96 80 00 00 00 C8 00 00 01 C8
    CONTEXT (2/1) object. Length:8.   R-type:4.     M-type:1
    DECISION (6/1) object. Length:8.   COMMAND cmd:1, flags:0

02:13:29:Notifying client (callback code 2)
02:13:29:COPS:** SENDING MESSAGE **
    COPS HEADER:Version 1, Flags 1, Opcode 3 (RPT), Client-type:1, Length:24
    HANDLE (1/1) object. Length:8.    00 00 21 01
    REPORT (12/1) object. Length:8.   REPORT type COMMIT (1)


02:13:29:COPS:Sent 24 bytes on socket,
02:13:29:Timer for connection entry is zero
```

To see an example where the **debug cops** command is used along with the **debug ip rsvp policy** command, refer to the second example of the **debug ip rsvp policy** command.

Related Commands

Command	Description
debug ip rsvp policy	Displays debug messages for RSVP policy processing.

debug cot

To display information about the COT functionality, use the **debug cot** privileged EXEC command. The **no** form of this command disables debugging output.

debug cot {**api** | **dsp** | **queue** | **detail**}

no debug cot {**api** | **dsp** | **queue** | **detail**}

Syntax Description

api	Displays information about the COT Application Program Interface (API).
dsp	Displays information related to the COT/DSP interface. Typical DSP functions include data modems, voice codecs, fax modems and codecs, and low-level signaling such as CAS/R2.
queue	Display information related to the COT internal queue.
detail	Display information about COT internal detail; summary of the **debug cot api**, **debug cot dsp**, and **debug cot queue** commands.

Command History

Release	Modification
11.3(7)	This command was introduced.

Examples

The following is sample output of the **debug cot api** command.

Figure 2 Sample debug cot api Command Output

```
Router# debug cot api

COT API debugging is on
08:29:55: cot_request_handler(): CDB@0x60DEDE14, req(COT_CHECK_TONE_ON):
08:29:55:      shelf 0 slot 0 appl_no 1 ds0 1
08:29:55:      freqTX 2010 freqRX 1780 key 0xFFF1 duration 60000
```

Table 31 describes the significant fields in the display.

Table 31 debug cot api Field Descriptions

Field	Description
CDB	Internal controller information.
req	Type of COT operation requested.
shelf	Shelf ID of the COT operation request.
slot	Designates the slot number, 1 to 4.
appl-no	Hardware unit that provides the external interface connections from a router to the network.
ds0	Number of the COT operation request.
key	COT operation identifier.
duration	Timeout duration of the COT operation.

Table 31 debug cot api Field Descriptions (continued)

Field	Description
freqTX	Requested transmit tone frequency.
freqRX	Requested receive tone frequency.

The following is sample output of the **debug cot dsp** command.

Figure 3 Sample debug cot dsp Command Output

```
Router# debug cot dsp

Router#
00:10:42:COT:DSP (1/1) Allocated
00:10:43:In cot_callback
00:10:43:  returned key 0xFFF1, status = 0
00:10:43:COT:Received DSP Q Event
00:10:43:COT:DSP (1/1) Done
00:10:43:COT:DSP (1/1) De-allocated
```

Table 32 describes the significant fields in the display.

Table 32 debug cot dsp Field Descriptions

Field	Description
DSP (1/1) Allocated	Slot and port of the DSP allocated for the COT operation.
Received DSP Q Event	Indicates the COT subsystem received an event from the DSP.
DSP (1/1) Done	Slot and port of the DSP transitioning to IDLE state.
DSP (1/1) De-allocated	Slot and port of the DSP de-allocated after the completion of the COT operation.

The following is sample output of the **debug cot queue** command.

```
Router# debug cot queue

Router#
00:11:26:COT(0x60EBB48C):Adding new request (0x61123DBC) to In
Progress Q
00:11:26:COT(0x60EBB48C):Adding COT(0x61123DBC) to the Q head
00:11:27:In cot_callback
00:11:27:  returned key 0xFFF1, status = 0
```

Table 33 describes the significant fields in the display.

Table 33 debug cot api Field Descriptions

Field	Description
COT	Internal COT operation request.
Adding new request	Internal COT operation request queue.

The following is sample output of the **debug cot detail** command.

```
Router# debug cot detail

Router#
00:04:57:cot_request_handler():CDB@0x60EBB48C, req(COT_CHECK_TONE_ON):

00:04:57:    shelf 0 slot 0 appl_no 1 ds0 1
00:04:57:    freqTX 1780 freqRX 2010 key 0xFFF1 duration 1000

00:04:57:COT:DSP (1/0) Allocated
00:04:57:COT:Request Transition to COT_WAIT_TD_ON
00:04:57:COT(0x60EBB48C):Adding new request (0x61123DBC) to In
Progress Q
00:04:57:COT(0x60EBB48C):Adding COT(0x61123DBC) to the Q head
00:04:57:COT:Start Duration Timer for Check Tone Request
00:04:58:COT:Received Timer Event
00:04:58:COT:T24 Timer Expired
00:04:58:COT Request@ 0x61123DBC, CDB@ 0x60EBB48C, Params@0x61123E08
00:04:58:  request type = COT_CHECK_TONE_ON
00:04:58:  shelf 0 slot 0 appl_no 1 ds0 1
00:04:58:  duration 1000 key FFF1 freqTx 1780 freqRx 2010
00:04:58:  state COT_WAIT_TD_ON_CT
00:04:58:  event_proc(0x6093B55C)

00:04:58:Invoke NI2 callback to inform COT request status
00:04:58:In cot_callback
00:04:58:  returned key 0xFFF1, status = 0
00:04:58:Return from NI2 callback
00:04:58:COT:Request Transition to IDLE
00:04:58:COT:Received DSP Q Event
00:04:58:COT:DSP (1/0) Done
00:04:58:COT:DSP (1/0) De-allocated
```

Because the **debug cot detail** command is a summary of the **debug cot api**, **debug cot dsp**, and **debug cot queue** commands, the field descriptions are the same.

debug cpp event

To display general Combinet Proprietary Protocol (CPP) events, use the **debug cpp event** privileged EXEC command. The **no** form of this command disables debugging output.

debug cpp event

no debug cpp event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debug cpp event** command displays events such as CPP sequencing, group creation, and keepalives.

Examples

One or more of the messages in Table 34 appear when you use the **debug cpp event** command. Each message begins with the short name of the interface the event occurred on (for example, SERIAL0:1 or BRI0:1) and might contain one or more packet sequence numbers or remote site names.

Table 34 debug cpp event Messages

Message	Description
BRI0:1: negotiation complete	Call was set up on the interface (in this example, BRI0:1).
BRI0:1: negotiation timed out	Call timed out.
BRI0:1: sending negotiation packet	Negotiation packet was sent to set up the call.
BRI0:1: out of sequence packet - got 10, range 1 8	Packet was received that was out of sequence. The first number displayed in the message is the sequence number received, and the following numbers are the range of valid sequence numbers.
BRI0:1: Sequence timer expired - Lost 11 Trying sequence 12	Timer expired before the packet was received. The first number displayed in the message is the sequence number of the packet that was lost, and the second number is the next sequence number.
BRI0:1: Line Integrity Violation	Router fails to maintain keepalives.
BRI0:1: create cpp group ber19 destroyed cpp group ber19	Dialer group is created on the remote site (in this example, ber19).

Related Commands

Command	Description
debug cpp negotiation	Displays CPP negotiation events.
debug cpp packet	Displays CPP packets.

debug cpp negotiation

To display Combinet Proprietary Protocol (CPP) negotiation events, use the **debug cpp negotiation** privileged EXEC command. The **no** form of this command disables debugging output.

debug cpp negotiation

no debug cpp negotiation

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debug cpp negotiation** command displays events such as the type of packet and packet size being sent.

Examples

The following is sample output from the **debug cpp negotiation** command. In this example, a sample connection is shown.

```
Router# debug cpp negotiation

%LINK-3-UPDOWN: Interface BRI0: B-Channel 2, changed state to down
%LINK-3-UPDOWN: Interface BRI0, changed state to up
%SYS-5-CONFIG_I: Configured from console by console
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
BR0:1:(I) NEG packet - len 77
   attempting proto:2
   ether id:0040.f902.c7b4
   port 1 number:5559876
   port 2 number:5559876
   origination port:1
   remote name:berl9
   password is correct
```

The following describes the significant fields in the display.

Table 35 Debug CPP Negotiation Field Descriptions

Field	Description
BR0:1 (I) NEG packet - len 77	Interface name, packet type, and packet size.
attempting proto:	CPP protocol type.
ether id:	Ethernet address of the destination router.
port 1 number:	ISDN phone number of remote B channel #1.
port 2 number:	ISDN phone number of remote B channel #2.
origination port:	B channel 1 or 2 called.
remote name:	Remote site name to which this call is connecting.
password is correct	Password is accepted so the connection is established.

Related Commands

Command	Description
debug cot	Displays information about the COT functionality.
debug cpp packet	Displays CPP packets.

debug cpp packet

To display Combinet Proprietary Protocol (CPP) packets, use the **debug cpp packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug cpp packet

no debug cpp packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debug cpp packet** command displays the hexadecimal values of the packets.

Examples

The following is sample output from the **debug cpp packet** command. This example shows the interface name, packet type, packet size, and the hexadecimal values of the packet.

```
Router# debug cpp packet

BR0:1:input packet - len 60
00 00 00 00 00 00 00 40 F9 02 C7 B4 08 0.!6 00 01
08 00 06 04 00 02 00 40 F9 02 C7 B4 83 6C A1 02!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 64/66/68 ms
BR0:1 output packet - len 116
06 00 00 40 F9 02 C7 B4 00 00 0C 3E 12 3A 08 00
45 00 00 64 00 01 00 00 FF 01 72 BB 83 6C A1 01
```

Related Commands

Command	Description
debug cot	Displays information about the COT functionality.
debug cpp negotiation	Displays CPP negotiation events.

debug crypto engine

To display debug messages about crypto engines, which perform encryption and decryption, use the **debug crypto engine** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug crypto engine

no debug crypto engine

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0	This command was introduced.

Usage Guidelines

Use the **debug crypto engine** command to display information pertaining to the crypto engine, such as when Cisco IOS software is performing encryption or decryption operations.

The crypto engine is the actual mechanism that performs encryption and decryption. A crypto engine can be software or a hardware accelerator. Some platforms can have multiple crypto engines; therefore, the router will have multiple hardware accelerators.

Examples

The following is sample output from the **debug crypto engine** command. The first sample output shows messages from a router that successfully generates RSA keys. The second sample output shows messages from a router that decrypts the RSA key during Internet Key Exchange (IKE) negotiation.

```
Router# debug crypto engine

00:25:13:CryptoEngine0:generate key pair
00:25:13:CryptoEngine0:CRYPTO_GEN_KEY_PAIR
00:25:13:CRYPTO_ENGINE:key process suspended and continued
00:25:14:CRYPTO_ENGINE:key process suspended and continuedcr
```

```
Router# debug crypto engine

00:27:45:%SYS-5-CONFIG_I:Configured from console by console
00:27:51:CryptoEngine0:generate alg parameter
00:27:51:CRYPTO_ENGINE:Dh phase 1 status:0
00:27:51:CRYPTO_ENGINE:Dh phase 1 status:0
00:27:51:CryptoEngine0:generate alg parameter
00:27:52:CryptoEngine0:calculate pkey hmac for conn id 0
00:27:52:CryptoEngine0:create ISAKMP SKEYID for conn id 1
00:27:52:Crypto engine 0:RSA decrypt with public key
00:27:52:CryptoEngine0:CRYPTO_RSA_PUB_DECRYPT
00:27:52:CryptoEngine0:generate hmac context for conn id 1
00:27:52:CryptoEngine0:generate hmac context for conn id 1
00:27:52:Crypto engine 0:RSA encrypt with private key
```

```
00:27:52:CryptoEngine0:CRYPTO_RSA_PRIV_ENCRYPT
00:27:53:CryptoEngine0:clear dh number for conn id 1
00:27:53:CryptoEngine0:generate hmac context for conn id 1
00:27:53:validate proposal 0
00:27:53:validate proposal request 0
00:27:54:CryptoEngine0:generate hmac context for conn id 1
00:27:54:CryptoEngine0:generate hmac context for conn id 1
00:27:54:ipsec allocate flow 0
00:27:54:ipsec allocate flow 0
```

Related Commands

Command	Description
crypto key generate rsa	Generates RSA key pairs.

debug crypto engine accelerator logs

To enable logging of commands and associated parameters sent from the VPN module driver to the VPN module hardware using a debug flag, use the **debug crypto engine accelerator logs** privileged EXEC command.

debug crypto engine accelerator logs

no debug crypto engine accelerator logs

Syntax Description

This command has no arguments or keywords.

Defaults

The logging of commands sent from the VPN module driver to the VPN module hardware is disabled.

Command History

Release	Modification
12.1(1)XC	This command was introduced on the Cisco 1720 and Cisco 1750 routers.

Usage Guidelines

Use the **debug crypto engine accelerator logs** command when encryption traffic is sent to the router and a problem with the encryption module is suspected.

This command is intended only for Cisco TAC personnel to collect debugging information.

Examples

The command **debug crypto engine accelerator logs** uses a debug flag to log commands and associated parameters sent from the VPN module driver to the VPN module hardware as follows:

```
Router# debug crypto engine accelerator logs

encryption module logs debugging is on
```

Related Commands

Command	Description
crypto engine accelerator	Enables or disables the crypto engine accelerator if it exists.
show crypto engine accelerator logs	Prints information about the last 32 CGX Library packet processing commands, and associated parameters sent from the VPN module driver to the VPN module hardware.
show crypto engine accelerator sa-database	Prints active (in-use) entries in the platform-specific VPN module database.
show crypto engine configuration	Displays the Cisco IOS crypto engine of your router.

debug crypto ipsec

To display IPSec events, use the **debug crypto ipsec** privileged EXEC command. The **no** form of this command disables debugging output.

debug crypto ipsec

no debug crypto ipsec

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug crypto ipsec** command. In this example, security associations (SAs) have been successfully established.

```
Router# debug crypto ipsec
```

IPSec requests SAs between 172.21.114.123 and 172.21.114.67, on behalf of the **permit ip host 172.21.114.123 host 172.21.114.67** command. It prefers to use the transform set esp-des w/esp-md5-hmac, but it will also consider ah-sha-hmac.

```
00:24:30: IPSEC(sa_request): ,
  (key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
    src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
    dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 120s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
00:24:30: IPSEC(sa_request): ,
  (key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
    src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
    dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1).,
    protocol= AH, transform= ah-sha-hmac ,
    lifedur= 120s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x0.
```

IKE asks for SPIs from IPSec. For inbound security associations, IPSec controls its own SPI space.

```
00:24:34: IPSEC(key_engine): got a queue event...
00:24:34: IPSEC(spi_response): getting spi 302974012ld for SA
        from 172.21.114.67   to 172.21.114.123  for prot 3
00:24:34: IPSEC(spi_response): getting spi 525075940ld for SA
        from 172.21.114.67   to 172.21.114.123  for prot 2
```

IKE will ask IPSec if it accepts the SA proposal. In this case, it will be the one sent by the local IPSec in the first place:

```
00:24:34: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
    dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
    src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
```

After the proposal is accepted, IKE finishes the negotiations, generates the keying material, and then notifies IPSec of the new security associations (one security association for each direction).

```
00:24:35: IPSEC(key_engine): got a queue event...
```

The following output pertains to the inbound SA. The conn_id value references an entry in the crypto engine connection table.

```
00:24:35: IPSEC(initialize_sas): ,
  (key eng. msg.) dest= 172.21.114.123, src= 172.21.114.67,
    dest_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
    src_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 120s and 4608000 kb,
    spi= 0x120F043C(302974012), conn_id= 29, keysize= 0, flags= 0x4
```

The following output pertains to the outbound SA:

```
00:24:35: IPSEC(initialize_sas): ,
  (key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
    src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
    dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 120s and 4608000kb,
    spi= 0x38914A4(59315364), conn_id= 30, keysize= 0, flags= 0x4
```

IPSec now installs the SA information into its SA database.

```
00:24:35: IPSEC(create_sa): sa created,
  (sa) sa_dest= 172.21.114.123, sa_prot= 50,
    sa_spi= 0x120F043C(302974012),
    sa_trans= esp-des esp-md5-hmac , sa_conn_id= 29
00:24:35: IPSEC(create_sa): sa created,
  (sa) sa_dest= 172.21.114.67, sa_prot= 50,
    sa_spi= 0x38914A4(59315364),
    sa_trans= esp-des esp-md5-hmac , sa_conn_id= 30
```

The following is sample output for the **debug crypto ipsec** command as seen on the peer router. In this example, IKE asks IPSec if it will accept an SA proposal. Although the peer sent two proposals, IPSec accepted the first proposal.

```
00:26:15: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
    dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
    src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
```

IKE asks for SPIs.

```
00:26:15: IPSEC(key_engine): got a queue event...
00:26:15: IPSEC(spi_response): getting spi 59315364ld for SA
        from 172.21.114.123  to 172.21.114.67   for prot 3
```

IKE does the remaining processing, completing the negotiation and generating keys. It then tells IPSec about the new SAs.

```
00:26:15: IPSEC(key_engine): got a queue event...
```

The following output pertains to the inbound SA:

```
00:26:15: IPSEC(initialize_sas): ,
  (key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
    dest_proxy= 172.21.114.67/0.0.0.0/0/0 (type=1),
    src_proxy= 172.21.114.123/0.0.0.0/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 120s and 4608000kb,
```

```
    spi= 0x38914A4(59315364), conn_id= 25, keysize= 0, flags= 0x4
```

The following output pertains to the outbound SA:

```
00:26:15: IPSEC(initialize_sas): ,
  (key eng. msg.) src= 172.21.114.67, dest= 172.21.114.123,
    src_proxy= 172.21.114.67/0.0.0.0/0/0 (type=1),
    dest_proxy= 172.21.114.123/0.0.0.0/0/0 (type=1),
    protocol= ESP, transform= esp-des esp-md5-hmac ,
    lifedur= 120s and 4608000kb,
    spi= 0x120F043C(302974012), conn_id= 26, keysize= 0, flags= 0x4
```

IPSec now installs the SA information into its SA database:

```
00:26:15: IPSEC(create_sa): sa created,
  (sa) sa_dest= 172.21.114.67, sa_prot= 50,
    sa_spi= 0x38914A4(59315364),
    sa_trans= esp-des esp-md5-hmac , sa_conn_id= 25
00:26:15: IPSEC(create_sa): sa created,
  (sa) sa_dest= 172.21.114.123, sa_prot= 50,
    sa_spi= 0x120F043C(302974012),
    sa_trans= esp-des esp-md5-hmac , sa_conn_id= 26
```

debug crypto isakmp

To display messages about IKE events, use the **debug crypto isakmp** privileged EXEC command. The **no** form of this command disables debugging output.

debug crypto isakmp

no debug crypto isakmp

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug crypto isakmp** command for an IKE peer that initiates an IKE negotiation.

First, IKE negotiates its own security association (SA), checking for a matching IKE policy:

```
Router# debug crypto isakmp

20:26:58: ISAKMP (8): beginning Main Mode exchange
20:26:58: ISAKMP (8): processing SA payload. message ID = 0
20:26:58: ISAKMP (8): Checking ISAKMP transform 1 against priority 10 policy
20:26:58: ISAKMP:      encryption DES-CBC
20:26:58: ISAKMP:      hash SHA
20:26:58: ISAKMP:      default group 1
20:26:58: ISAKMP:      auth pre-share
20:26:58: ISAKMP (8): atts are acceptable. Next payload is 0
```

IKE has found a matching policy. Next, the IKE SA is used by each peer to authenticate the other peer:

```
20:26:58: ISAKMP (8): SA is doing pre-shared key authentication
20:26:59: ISAKMP (8): processing KE payload. message ID = 0
20:26:59: ISAKMP (8): processing NONCE payload. message ID = 0
20:26:59: ISAKMP (8): SKEYID state generated
20:26:59: ISAKMP (8): processing ID payload. message ID = 0
20:26:59: ISAKMP (8): processing HASH payload. message ID = 0
20:26:59: ISAKMP (8): SA has been authenticated
```

Next, IKE negotiates to set up the IPSec SA by searching for a matching transform set:

```
20:26:59: ISAKMP (8): beginning Quick Mode exchange, M-ID of 767162845
20:26:59: ISAKMP (8): processing SA payload. message ID = 767162845
20:26:59: ISAKMP (8): Checking IPSec proposal 1
20:26:59: ISAKMP: transform 1, ESP_DES
20:26:59: ISAKMP:   attributes in transform:
20:26:59: ISAKMP:      encaps is 1
20:26:59: ISAKMP:      SA life type in seconds
20:26:59: ISAKMP:      SA life duration (basic) of 600
20:26:59: ISAKMP:      SA life type in kilobytes
20:26:59: ISAKMP:      SA life duration (VPI) of
  0x0 0x46 0x50 0x0
20:26:59: ISAKMP:      authenticator is HMAC-MD5
20:26:59: ISAKMP (8): atts are acceptable.
```

A matching IPSec transform set has been found at the two peers. Now the IPSec SA can be created (one SA is created for each direction):

```
20:26:59: ISAKMP (8): processing NONCE payload. message ID = 767162845
20:26:59: ISAKMP (8): processing ID payload. message ID = 767162845
20:26:59: ISAKMP (8): processing ID payload. message ID = 767162845
```

```
20:26:59: ISAKMP (8): Creating IPSec SAs
20:26:59:        inbound SA from 155.0.0.2 to 155.0.0.1 (proxy 155.0.0.2 to 155.0.0.1
)
20:26:59:         has spi 454886490 and conn_id 9 and flags 4
20:26:59:         lifetime of 600 seconds
20:26:59:         lifetime of 4608000 kilobytes
20:26:59:        outbound SA from 155.0.0.1      to 155.0.0.2      (proxy 155.0.0.1
to 155.0.0.2      )
20:26:59:         has spi 75506225 and conn_id 10 and flags 4
20:26:59:         lifetime of 600 seconds
20:26:59:         lifetime of 4608000 kilobytes
```

debug crypto key-exchange

To show Digital Signature Standard (DSS) public key exchange messages, use the **debug crypto key-exchange** privileged EXEC command. The **no** form of this command disables debugging output.

debug crypto key-exchange

no debug crypto key-exchange

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Encryption and authentication are provided by a software service on the router called a *crypto engine*. The crypto engine performs authentication through DSS public and private keys when a connection is set up. DSS is a means of sending a "signature" at the end of a message that positively identifies the author of the message. The signature cannot be forged or duplicated by others, so whoever received a message with a DSS signature knows exactly who sent the message.

If the process of exchanging DSS public keys with a peer router by means of the **config crypto key-exchange** command is not successful, try to exchange DSS public keys again after enabling the **debug crypto key-exchange** command to help you diagnose the problem.

Examples

The following is sample output from the **debug crypto key-exchange** command. The first shows output from the initiating router in a key exchange. The second shows output from the passive router in a key exchange. The number of bytes received should match the number of bytes sent from the initiating side, although the number of messages can be different.

```
Router# debug crypto key-exchange

CRYPTO-KE: Sent 4 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 64 bytes.

Router# debug crypto key-exchange

CRYPTO-KE: Received 4 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 49 bytes.
CRYPTO-KE: Received 15 bytes.
```

Related Commands

Command	Description
debug crypto sesmgmt	Displays connection setup messages and their flow through the router.

debug crypto pki messages

To display debug messages for the details of the interaction (message dump) between the certification authority (CA) and the router, use the **debug crypto pki messages** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug crypto pki messages

no debug crypto pki messages

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0	This command was introduced.

Usage Guidelines

Use the **debug crypto pki messages** command to display messages about the actual data being sent and received during public key infrastructure (PKI) transactions.

You can also use the **show crypto ca certificates** command to display information about your certificate.

Examples

The following example is sample output for the **debug crypto pki messages** command:

```
Router# debug crypto pki messages

Fingerprint: 2CFC6265 77BA6496 3AEFCB50 29BC2BF2
00:48:23:Write out pkcs#10 content:274
00:48:23:30 82 01 0E 30 81 B9 02 01 00 30 22 31 20 30 1E 06 09 2A 86
00:48:23:48 86 F7 0D 01 09 02 16 11 70 6B 69 2D 33 36 61 2E 63 69 73
00:48:23:63 6F 2E 63 6F 6D 30 5C 30 0D 06 09 2A 86 48 86 F7 0D 01 01
00:48:23:01 05 00 03 4B 00 30 48 02 41 00 DD 2C C6 35 A5 3F 0F 97 6C
00:48:23:11 E2 81 95 01 6A 80 34 25 10 C4 5F 3D 8B 33 1C 19 50 FD 91
00:48:23:6C 2D 65 4C B6 A6 B0 02 1C B2 84 C1 C8 AC A4 28 6E EF 9D 3B
00:48:23:30 98 CB 36 A2 47 4E 7E 6F C9 3E B8 26 BE 15 02 03 01 00 01
00:48:23:A0 32 30 10 06 09 2A 86 48 86 F7 0D 01 09 07 31 03 13 01 63
00:48:23:30 1E 06 09 2A 86 48 86 F7 0D 01 09 0E 31 11 14 0F 30 0D 30
00:48:23:0B 06 03 55 1D 0F 04 04 03 02 05 A0 30 0D 06 09 2A 86 48 86
00:48:23:F7 0D 01 01 04 05 00 03 41 00 2C FD 88 2C 8A 13 B6 81 88 EA
00:48:23:5C FD AE 52 8F 2C 13 95 9E 9D 8B A4 C9 48 32 84 BF 05 03 49
00:48:23:63 27 A3 AC 6D 74 EB 69 E3 06 E9 E4 9F 0A A8 FB 20 F0 02 03
00:48:23:BE 90 57 02 F2 75 8E 0F 16 60 10 6F BE 2B
00:48:23:Enveloped Data ...

00:48:23:30 80 06 09 2A 86 48 86 F7 0D 01 07 03 A0 80 30 80 02 01 00
00:48:23:31 80 30 82 01 0F 02 01 00 30 78 30 6A 31 0B 30 09 06 03 55
00:48:23:04 06 13 02 55 53 31 0B 30 09 06 03 55 04 08 13 02 43 41 31
00:48:23:13 30 11 06 03 55 04 07 13 0A 53 61 6E 74 61 20 43 72 75 7A
00:48:23:31 15 30 13 06 03 55 04 0A 13 0C 43 69 73 63 6F 20 53 79 73
00:48:23:74 65 6D 31 0E 30 0C 06 03 55 04 0B 13 05 49 50 49 53 55 31
```

```
00:48:23:Signed Data 1382 bytes
00:48:23:30 80 06 09 2A 86 48 86 F7 0D 01 07 02 A0 80 30 80 02 01 01
00:48:23:31 0E 30 0C 06 08 2A 86 48 86 F7 0D 02 05 05 00 30 80 06 09
00:48:23:2A 86 48 86 F7 0D 01 07 01 A0 80 24 80 04 82 02 75 30 80 06
00:48:23:02 55 53 31 0B 30 09 06 03 55 04 08 13 02 43 41 31 13 30 11
00:48:23:33 34 5A 17 0D 31 30 31 31 31 35 31 38 35 34 33 34 5A 30 22
00:48:23:31 20 30 1E 06 09 2A 86 48 86 F7 0D 01 09 02 16 11 70 6B 69
00:48:23:2D 33 36 61 2E 63 69 73 63 6F 2E 63 6F 6D 30 5C 30 0D 06 09
00:48:23:2A 86 48 86 F7 0D 01 01 01 05 00 03 4B 00 30 48 02 41 00 DD
00:48:23:2C C6 35 A5 3F 0F 97 6C 11 E2 81 95 01 6A 80 34 25 10 C4 5F
00:48:23:3D 8B 33 1C 19 50 FD 91 6C 2D 65 4C B6 A6 B0 02 1C B2 84 C1
00:48:23:86 F7 0D 01 01 01 05 00 04 40 C6 24 36 D6 D5 A6 92 80 5D E5
00:48:23:15 F7 3E 15 6D 71 E1 D0 13 2B 14 64 1B 0C 0F 96 BF F9 2E 05
00:48:23:EF C2 D6 CB 91 39 19 F8 44 68 0E C5 B5 84 18 8B 2D A4 B1 CD
00:48:23:3F EC C6 04 A5 D9 7C B1 56 47 3F 5B D4 93 00 00 00 00 00 00
00:48:23:00 00
00:48:24:Received pki message:1778 types
00:48:24:30 82 06 EE 06 09 2A 86 48 86 F7 0D 01 07 02 A0 82 06 DF 30
00:48:24:82 06 DB 02 01 01 31 0E 30 0C 06 08 2A 86 48 86 F7 0D 02 05
00:48:24:05 00 30 82 04 C5 06 09 2A 86 48 86 F7 0D 01 07 01 A0 82 04
00:48:24:B6 04 82 04 B2 30 82 04 AE 06 09 2A 86 48 86 F7 0D 01 07 03
00:48:24:0E 61 85 48 B1 DA 3D 73 F1 4B D8 5E 03 6E F3 E5 72 5D D7 17
00:48:24:17 3D 03 19 B3 8F 06 8B FE FB B1 CE D4 4C 4D 1B 81 CF 59 B7
00:48:24:78 DD 27 BA 28 2F 85 09 F0 61 74 0F 0F 92 F0 C8 C7 5B 96 E7
00:48:24:71 AF 87 D2 72 75 B7 F7 89 6F E4 E7 57 84 76 53 0B 50 8A B9
00:48:24:05 54 6F 06 75 72 8A AF 54 A6 EF 70 2D 15 6C B7 30 91 1C 00
00:48:24:CB 26 80 8D DC 89 77 57 1E D5 7A 37 86 BE 44 F8 66 60
00:48:24:Verified signed data 1202 bytes:
00:48:24:30 82 04 AE 06 09 2A 86 48 86 F7 0D 01 07 03 A0 82 04 9F 30
00:48:24:82 04 9B 02 01 00 31 81 9F 30 81 9C 02 01 00 30 46 30 22 31
00:48:24:20 30 1E 06 09 2A 86 48 86 F7 0D 01 09 02 16 11 70 6B 69 2D
00:48:24:33 36 61 2E 63 69 73 63 6F 2E 63 6F 6D 02 20 34 45 45 41 44
00:48:24:E2 55 65 DE DB 23 91 D7 60 53 96 64 BE F2 30 A7 8B 1B D9 EB
00:48:24:2E EB 9B 0D 75 EC 8E AF C0 9C 62 78 29 E0 97 00 EA 84 80 DD
00:48:24:AB 83 32 89 3E 5B A9 9F A9 9A 6D 3A 87 E2 71 16 C9 C1 E4 DB
00:48:24:FA 5A FC F3 31 98 2B 8E 55 71 C4 F6 BF CE 45 CA A5 47 40 9B
00:48:24:19 E3 1A C3 F5 ED 4D 81 1F 6F 34 35 E2 00 B3 93 DD A0 6A 74
00:48:24:EA 2B A8 D4 32 53 A7 86 50 71 5E 2A 64 BE 4B B1 72 AB 8C DA
00:48:24:AB 7A 2A 07 C0 7E C1 A7 12 31 33 AB 94 E0 3B A2 68 17 DE CE
00:48:24:57 70 2D 0B F5 C8 A7 FC FE 40 74 E8 EB 9C 82 77 DE A4 FA 75
00:48:24:FF 6F 7B E6 74 E2 F5 A1 9A C8 3C 23 DB 4A 90 BE 4A 94 EB 8B
00:48:24:ED F3
00:48:24:Decrypted enveloped content:
00:48:24:30 82 03 C8 06 09 2A 86 48 86 F7 0D 01 07 02 A0 82 03 B9 30
00:48:24:82 03 B5 02 01 01 31 00 30 0B 06 09 2A 86 48 86 F7 0D 01 07
00:48:24:01 A0 82 03 9D 30 82 03 99 30 82 03 43 A0 03 02 01 02 02 0A
00:48:24:70 45 B3 F6 00 00 00 00 01 23 30 0D 06 09 2A 86 48 86 F7 0D
000:48:24:35 35 32 32 5A 30 22 31 20 30 1E 06 09 2A 86 48 86 F7 0D 01
00:48:24:09 02 13 11 70 6B 69 2D 33 36 61 2E 63 69 73 63 6F 2E 63 6F
00:48:24:6D 30 5C 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05 00 03 4B
00:48:24:00 30 48 02 41 00 DD 2C C6 35 A5 3F 0F 97 6C 11 E2 81 95 01
00:48:24:6A 80 34 25 10 C4 5F 3D 8B 33 1C 19 50 FD 91 6C 2D 65 4C B6
00:48:24:63 6F 2E 63 6F 6D 2F 43 65 72 74 45 6E 72 6F 6C 6C 2F 6D 73
00:48:24:63 61 2D 72 6F 6F 74 5F 6D 73 63 61 2D 72 6F 6F 74 2E 63 72
00:48:24:74 30 41 06 08 2B 06 01 05 05 07 30 02 86 35 66 69 6C 65 3A
00:48:24:2F 2F 5C 5C 6D 73 63 61 2D 72 6F 6F 74 5C 43 65 72 74 45 6E
00:48:24:72 6F 6C 6C 5C 6D 73 63 61 2D 72 6F 6F 74 5F 6D 73 63 61 2D
00:48:24:72 6F 6F 74 2E 63 72 74 30 0D 06 09 2A 86 48 86 F7 0D 01 01
00:48:24:05 05 00 03 41 00 56 30 AD 99 1F FA 0D 1A C3 3D 71 2A DB A0
00:48:24:48 C5 EB C8 D4 FE 62 49 9C 69 5D E4 80 77 19 3E 07 B8 2B 4F
00:48:24:9A D7 72 A7 26 25 61 AE 5B 1C B5 7B 4C 18 CA 17 C3 D0 76 84
00:48:24:75 41 92 74 5E A4 E8 9E 09 60 31 00
00:48:24:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

Related Commands

Command	Description
crypto ca enroll	Obtains the certificate of your router from the CA.
debug crypto pki transactions	Displays debug messages for the trace of interaction (message type) between the CA and the router.
show crypto ca certificates	Displays information about your certificate, the certificate of the CA, and any RA certificates.

debug crypto sesmgmt

To show connection setup messages and their flow through the router, use the **debug crypto sesmgmt** privileged EXEC command. The **no** form of this command disables debugging output.

debug crypto sesmgmt

no debug crypto sesmgmt

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Encryption and authentication are provided by a software service on the router called a *crypto engine*. The crypto engine performs authentication through DSS public and private keys when a connection is set up. DSS is a means of sending a "signature" at the end of a message that positively identifies the author of the message. The signature cannot be forged or duplicated by others, so whoever receives a message with a DSS signature knows exactly who sent the message.

When connections are not completing, use the **debug crypto sesmgmt** command to follow the progress of connection messages as a first step in diagnosing the problem. You see a record of each connection message as the router discovers it, and can track its progress through the necessary signing, verifying, and encryption session setup operations. Other significant connection setup events, such as the pregeneration of Diffie-Hellman public numbers, are also shown. For information on Diffie-Hellman public numbers, refer to the *Security Configuration Guide*.

Also use the **show crypto connections** command to display additional information on connections.

Examples

The following is sample output from the **debug crypto sesmgmt** command. The first shows messages from a router that initiates a successful connection. The second shows messages from a router that receives a connection.

```
Router# debug crypto sesmgmt

CRYPTO: Dequeued a message: Inititate_Connection
CRYPTO: DH gen phase 1 status for conn_id 2 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: send_nnc_req:   NNC Echo Request sent
CRYPTO: Dequeued a message: CRM
CRYPTO: DH gen phase 2 status for conn_id 2 slot 0:OK
CRYPTO: Verify done. Status=OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: recv_nnc_rpy:   NNC Echo Confirm sent
CRYPTO: Create encryption key for conn_id 2 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 2 (slot 0)

Router# debug crypto sesmgmt

CRYPTO: Dequeued a message: CIM
CRYPTO: Verify done. Status=OK
CRYPTO: DH gen phase 1 status for conn_id 1 slot 0:OK
CRYPTO: DH gen phase 2 status for conn_id 1 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.162, d=172.21.114.163
```

```
CRYPTO-SDU: act_on_nnc_req: NNC Echo Reply sent
CRYPTO: Create encryption key for conn_id 1 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 1 (slot 0)
CRYPTO: Dequeued a message: CCM
CRYPTO: Verify done. Status=OK
```

Related Commands

Command	Description
debug crypto key-exchange	Displays DSS public key exchange messages.

debug crypto pki transactions

To display debug messages for the trace of interaction (message type) between the certification authority (CA) and the router, use the **debug crypto pki transactions** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug crypto pki transactions

no debug crypto pki transactions

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

Release	Modification
12.0	This command was introduced.

Usage Guidelines

Use the **debug crypto pki transactions** command to display debug messages pertaining to public key infrastructure (PKI) certificates. The messages will show status information during certificate enrollment and verification.

You can also use the **show crypto ca certificates** command to display information about your certificate.

Examples

The following example, which authenticates and enrolls a CA, contains sample output for the **debug crypto pki transactions** command:

```
Router(config)# crypto ca authenticate msca
Certificate has the following attributes:
Fingerprint:A5DE3C51 AD8B0207 B60BED6D 9356FB00
% Do you accept this certificate? [yes/no]:y

Router# debug crypto pki transactions

00:44:00:CRYPTO_PKI:Sending CA Certificate Request:
GET /certsrv/mscep/mscep.dll/pkiclient.exe?operation=GetCACert&message=msca HTTP/1.0

00:44:00:CRYPTO_PKI:http connection opened
00:44:01:CRYPTO_PKI:HTTP response header:
 HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:50:59 GMT
Content-Length:2693
Content-Type:application/x-x509-ca-ra-cert

Content-Type indicates we have received CA and RA certificates.

00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status
```

```
00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status

00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:transaction GetCACert completed
00:44:01:CRYPTO_PKI:CA certificate received.
00:44:01:CRYPTO_PKI:CA certificate received.
Router(config)# crypto ca enroll msca
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.

Password:
Re-enter password:

% The subject name in the certificate will be:Router.cisco.com
% Include the router serial number in the subject name? [yes/no]:n
% Include an IP address in the subject name? [yes/no]:n
Request certificate from CA? [yes/no]:y
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

Router(config)#    Fingerprint: 2CFC6265 77BA6496 3AEFCB50 29BC2BF2

00:44:29:CRYPTO_PKI:transaction PKCSReq completed
00:44:29:CRYPTO_PKI:status:
00:44:29:CRYPTO_PKI:http connection opened
00:44:29:CRYPTO_PKI: received msg of 1924 bytes
00:44:29:CRYPTO_PKI:HTTP response header:
 HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:51:28 GMT
Content-Length:1778
Content-Type:application/x-pki-message

00:44:29:CRYPTO_PKI:signed attr:pki-message-type:
00:44:29:13 01 33
00:44:29:CRYPTO_PKI:signed attr:pki-status:
00:44:29:13 01 30
00:44:29:CRYPTO_PKI:signed attr:pki-recipient-nonce:
00:44:29:04 10 B4 C8 2A 12 9C 8A 2A 4A E1 E5 15 DE 22 C2 B4 FD
00:44:29:CRYPTO_PKI:signed attr:pki-transaction-id:
00:44:29:13 20 34 45 45 41 44 42 36 33 38 43 33 42 42 45 44 45 39 46
00:44:29:34 38 44 33 45 36 39 33 45 33 43 37 45 39
00:44:29:CRYPTO_PKI:status = 100:certificate is granted
00:44:29:CRYPTO__PKI:All enrollment requests completed.
00:44:29:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

Related Commands

Command	Description
crypto ca authenticate	Authenticates the CA (by getting the certificate of the CA).
crypto ca enroll	Obtains the certificate of your router from the CA.

Command	Description
debug crypto pki messages	Displays debug messages for details of the interaction (message dump) between the CA and the router.
show crypto ca certificates	Displays information about your certificate, the certificate of the CA, and any RA certificates.

debug csm voice

To turn on debugging for all CSM VoIP calls, use the **debug csm voice** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug csm voice [*slot* | *dspm* | *dsp* | *dsp-channel*]

no debug csm voice [*slot* | *dspm* | *dsp* | *dsp-channel*]

Syntax Description

slot | *dspm* | *dsp* | *dsp-channel*	(Optional) Identifies the location of a particular DSP channel.

Usage Guidelines

The **debug csm voice** command turns on debugging for all CSM Voice-over-IP calls. If this command has no keyword specified, then debugging is enabled for all voice calls. The **no debug cms voice** command turns off debugging information for all voice calls.

If the keyword *slot* | *dspm* | *dsp* | *dsp-channel* argument is specified, then (if the specified DSP channel is engaged in a CSM call) CSM call-related debugging information will be turned on for this channel. The **no** form of this command turns off debugging for that particular channel.

Examples

The following examples show sample output from the **debug csm voice** command. The following shows that CSM has received an event from ISDN.

```
Oct 18 04:05:07.052: EVENT_FROM_ISDN::dchan_idb=0x60D7B6B8, call_id=0xCF, ces=0x1
bchan=0x0, event=0x1, cause=0x0
```

In this example, terms are explained as follows:

- dchan_idb—Indicates the address of the hardware IDB for the D channel
- call_id—Indicates the call ID assigned by ISDN
- bchan—Indicates the number of the B channel assigned for this call
- cause—Indicates the ISDN event cause

The following shows that CSM has allocated the CSM voice control block for the DSP device on slot 1 port 10 for this call.

```
Oct 18 04:05:07.052: VDEV_ALLOCATE: slot 1 and port 10 is allocated.
```

This AS5300 access server might not be actually used to handle this call. CSM must first allocate the CSM voice control block to initiate the state machine. After the voice control block has been allocated, CSM obtains from the DSP Resource Manager the actual DSP channel that will be used for the call. At that point, CSM will switch to the actual logical port number. The slot number refers to the physical slot on the AS5300 access server. The port number is the logical DSP number interpreted as listed in Table 36.

Table 36 Logical DSP Numbers

Logical Port Number	Physical DSP Channel
Port 0	DSPRM 1, DSP 1, DSP channel 1
Port 1	DSPRM 1, DSP 1, DSP channel 2
Port 2	DSPRM 1, DSP 2, DSP channel 1
Port 3	DSPRM 1, DSP 2, DSP channel 2
Port 4	DSPRM 1, DSP 3, DSP channel 1
Port 5	DSPRM 1, DSP 3, DSP channel 2
Port 6	DSPRM 1, DSP 4, DSP channel 1
Port 7	DSPRM 1, DSP 4, DSP channel 2
Port 8	DSPRM 1, DSP 5, DSP channel 1
Port 9	DSPRM 1, DSP 5, DSP channel 2
Port 10	DSPRM 1, DSP 6, DSP channel 1
Port 11	DSPRM 1, DSP 6, DSP channel 2
Port 12	DSPRM 2, DSP 1, DSP channel 1
Port 13	DSPRM 2, DSP 1, DSP channel 2
Port 14	DSPRM 2, DSP 2, DSP channel 1
Port 15	DSPRM 2, DSP 2, DSP channel 2
Port 16	DSPRM 2, DSP 3, DSP channel 1
Port 17	DSPRM 2, DSP 3, DSP channel 2
Port 18	DSPRM 2, DSP 4, DSP channel 1
Port 19	DSPRM 2, DSP 4, DSP channel 2
Port 20	DSPRM 2, DSP 5, DSP channel 1
Port 21	DSPRM 2, DSP 5, DSP channel 2
Port 22	DSPRM 2, DSP 6, DSP channel 1
Port 23	DSPRM 2, DSP 6, DSP channel 2
Port 48	DSPRM 5, DSP 1, DSP channel 1
Port 49	DSPRM 5, DSP 1, DSP channel 2
Port 50	DSPRM 5, DSP 2, DSP channel 1
Port 51	DSPRM 5, DSP 2, DSP channel 2
Port 52	DSPRM 5, DSP 3, DSP channel 1
Port 53	DSPRM 5, DSP 3, DSP channel 2
Port 54	DSPRM 5, DSP 4, DSP channel 1
Port 55	DSPRM 5, DSP 4, DSP channel 2
Port 56	DSPRM 5, DSP 5, DSP channel 1
Port 57	DSPRM 5, DSP 5, DSP channel 2
Port 58	DSPRM 5, DSP 6, DSP channel 1
Port 59	DSPRM 5, DSP 6, DSP channel 2

The following shows that the function csm_vtsp_init_tdm() has been called with a voice control block of address 0x60B8562C. This function will be called only when the call is treated as a voice call.

```
Oct 18 04:05:07.052: csm_vtsp_init_tdm (voice_vdev=0x60B8562C)
```

The following shows that CSM has obtained a DSP channel from the DSP Resource Manager:

```
Oct 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 2, dsp 5,
dsp_channel 1csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 9, bank 0,
bp_channel 10
```

The DSP channel has the following initialized TDM channel information:

- TDM slot 1, dspm 2, dsp 5, dsp_channel 1—Indicates the physical DSP channel that will be used for this call.
- TDM stream 5, channel 9, bank 0, bp_channel 10—Indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 9 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 10 means that the backplane stream 0 and backplane channel #1 are assigned to this DSP.

The following shows that CSM has received an incoming call event from ISDN:

```
Oct 18 04:05:07.052: EVENT_FROM_ISDN:(00CF): DEV_INCALL at slot 1 and port 20
```

Slot 1, port 20 means the logical DSP channel 20 (mapped to DSPRM 2, DSP 5, DSP channel 1).

The following shows that the DEV_INCALL message has been translated into a CSM_EVENT_ISDN_CALL message:

```
Oct 18 04:05:07.052: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 20
```

This message is passed to the CSM central state machine while it is in the CSM_IDLE state and is in the CSM_PROC_IDLE procedure. The logical DSP channel port 20 on slot 1 is used to handle this call.

The following shows that CSM has invoked the vtsp_ic_notify() function with a CSM voice call control block 0x60B8562C.

```
Oct 18 04:05:07.052: vtsp_ic_notify : (voice_vdev= 0x60B8562C)
```

Inside this function, CSM will send a SETUP INDICATION message to the VTSP. This function will be invoked only if the call is a voice call.

The following shows that CSM has received a SETUP INDICATION RESPONSE message from the VTSP as an acknowledgement.

```
Oct 18 04:05:07.056: csm_vtsp_call_setup_resp (vdev_info=0x60B8562C, vtsp_cdb=0x60FCA114)
```

This means that the VTSP has received the CALL SETUP INDICATION message previously sent and has proceeded to process the call.

- vdev_info—Contains the address of the CSM voice data block.
- vtsp_cdb—Contains the address of the VTSP call control block.

The following shows that CSM has received a CALL CONNECT message from the VTSP:

```
Oct 18 04:05:07.596: csm_vtsp_call_connect (vtsp_cdb=0x60FCA114, voice_vdev=0x60B8562C)
```

This indicates that the VTSP has received a CONNECT message for the call leg initiated to the Internet side.

- vtsp_cdb—Contains the address of the VTSP call control block.
- voice_vdev—Contains the address of the CSM voice data block.

The following shows that while CSM is in the CSM_IC2_RING state, it receives a SETUP INDICATION RESPONSE from the VTSP. This message is translated into CSM_EVENT_MODEM_OFFHOOK and passed to the CSM central state machine.

```
Oct 18 04:05:07.596: CSM_PROC_IC2_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 20
```

The following shows that CSM has received a CONNECT message from ISDN for the call using the logical DSP channel on slot 1 and port 20:

```
Oct 18 04:05:07.616: EVENT_FROM_ISDN:(00CF): DEV_CONNECTED at slot 1 and port 20
```

The following shows that CSM has translated the CONNECT event from ISDN into the CSM_EVENT_ISDN_CONNECTED message, which is then passed to the CSM central state machine:

```
Oct 18 04:05:07.616: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 20
```

The following shows that CSM has received a CALL SETUP REQUEST from the VTSP:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request (vtsp_cdb=0x60FCFA20,
vtsp_sdb=0x60DFB608)
```

This represents a request to make an outgoing call to the PSTN.

- vtsp_cdb—Contains the address of the VTSP call control block.
- vtsp_sdb—Contains the address of the signalling data block for the signalling interface to be used to send the outgoing call.

The following shows that the physical DSP channel has been allocated for this outgoing call:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm slot 1, dspm 5, dsp 4, dsp_channel 1
```

The following shows the on-chip and backplane TDM channel assigned to this DSP channel:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm stream 5, channel 25, bank 0,
bp_channel 27
```

In this sample output, tdm stream 5, channel 25, bank 0, bp_channel 27 indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 25 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 27 means that the backplane stream 0 and backplane channel 1 are assigned to this DSP.

The following shows the calling number and the called number for this call.

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: calling number: 10001, called number:
30001
```

The following shows that the CALL SETUP REQUEST from the VTSP has been translated into the ' CSM_EVENT_MODEM_OFFHOOK message and is passed to the CSM central state machine:

```
May 16 12:22:27.580: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 54
```

The logical DSP channel number for the DSP (slot 1, port 54) is now displayed, which maps to the physical DSP channel slot 1, dspm 5, dsp 4, dsp_channel 1.

The following shows that CSM has collected all the digits for dialing out:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS at slot 1, 
port 54
```

For PRI and for applications that do not require digit collection of outdialing digits (for example, voice calls), the intermediate digit collection states are omitted and the CSM state machine moves to this state directly, pretending that the digit collection has been done.

The following shows an information message:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num: (30001) at slot 1, 
port 54
```

The following shows that CSM attempts to find a free signalling D channel to direct the outgoing call:

```
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0, 
dchan_idb=0x60E8ACF0)
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0, 
dchan_idb=0x60D7ACB0)
```

In the case of voice calls, the free signalling D channel must match the voice interface specified inside the signalling data block (vtsp_sdb) passed from the VTSP.

The following shows that CSM has received an event from ISDN:

```
May 16 12:22:27.624: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1 
bchan=0x1E, event=0x3, cause=0x0
```

In this sample output:

- dchan_idb—indicates the address of the hardware IDB for the D channel
- call_id—Indicates the call id assigned by ISDN
- bchan—Indicates the number of the B channel assigned for this call
- cause—Indicates the ISDN event cause

The following shows that CSM has received a CALL PROCEEDING message from ISDN.

```
May 16 12:22:27.624: EVENT_FROM_ISDN:(A121): DEV_CALL_PROC at slot 1 and port 54
```

The following shows that the CALL PROCEEDING event received from ISDN has been interpreted as a CSM_EVENT_ISDN_BCHAN_ASSIGNED message:

```
*May 16 12:22:27.624: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port 
54
```

ISDN has assigned a B channel for this outgoing call. This B channel must be on the same PRI span as the signalling D channel allocated previously.

The following shows that the csm_vtsp_setup_for_oc function is called:

```
May 16 12:22:27.624: csm_vtsp_setup_for_oc (voice_vdev=0x60B8562C)
```

This is invoked when an outgoing call initiated by the VTSP receives a response from the ISDN stack.

The following shows that ISDN has sent a CONNECT message to CSM indicating that the call leg to the PSTN side has been established:

```
May 16 12:22:28.084: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
   bchan=0x1E, event=0x4, cause=0x0
May 16 12:22:28.084: EVENT_FROM_ISDN:(A121): DEV_CONNECTED at slot 1 and port 54
```

The following shows that while CSM is in the OC5_WAIT_FOR_CARRIER state, it has received the 'CONNECT' message from ISDN and has translated it into the CSM_EVENT_ISDN_CONNECTED message, which is passed to the CSM central state machine:

```
May 16 12:22:28.084: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 54
```

The following shows that the function vtsp_confirm_oc() has been called:

```
May 16 12:22:28.084: vtsp_confirm_oc : (voice_vdev= 0x60B8562C)
```

This is invoked after CSM received the CONNECT message from ISDN. CSM sends a confirmation of the CONNECT to the VTSP.

debug ctunnel

To display debug messages for the IP over a CLNS Tunnel feature, use the **debug ctunnel** privileged EXEC command. To disable the debug messages, use the **no** form of this command.

debug ctunnel

no debug ctunnel

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(5)	This command was introduced.

Examples

As packets are sent over the virtual interface, the following type of output will appear on the console when the **debug ctunnel** command is used:

```
4d21h: CTunnel1: IPCLNP encapsulated 49.0001.1111.1111.1111.00->49.0001.2222.2222.2222.00
(linktype=7, len=89)
```

debug custom-queue

To enable custom queueing output, use the **debug custom-queue** EXEC command. Use the **no** form of this command to disable custom queueing output.

debug custom-queue

no debug custom-queue

Syntax Description

This command has no arguments or keywords.

Examples

The following is an example of enabling custom queueing output:

```
Router# debug custom-queue

Custom output queueing debugging is on
```

The following is sample output from the **debug custom-queue** command:

```
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
```

Related Commands

Command	Description
debug priority	Enables priority queueing output.

debug dbconn all

To turn on all debug flags for Database Connection, use the **debug dbconn all** privileged EXEC command. The Database Connection debug flags are **appc**, **config**, **drda**, **event**, and **tcp**. Use the **no** form of this command to disable all debugging output.

debug dbconn all

no debug dbconn all

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled for Database Connection.

Usage Guidelines

The **debug dbconn all** command displays debug output for APPC, Database Connection configuration, DRDA, error messages, event traces, and TCP.

Examples

See the sample output provided for the **debug dbconn appc**, **debug dbconn config**, **debug dbconn drda**, **debug dbconn event**, and **debug dbconn tcp** commands.

Related Commands

Command	Description
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn event	Displays trace or error messages for Database Connection events.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn appc

To display APPC-related trace or error messages, use the **debug dbconn appc** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug dbconn appc

no debug dbconn appc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

In a router with stable Database Connection, the alias_cp_name field in the trace message should not be blank. There should be no other APPC error message. You can use APPN debug commands with this debug command to track APPN-related errors.

Examples

The following is sample output from the **debug dbconn appc** command. In a normal situation, only the following message is displayed:

```
DBCONN-APPC: alias_cp_name is "ASH"
```

The following error messages are displayed if there is a network configuration error or other APPN-related problem:

```
DBCONN-APPC-612C2B28: APPC error: opcode 0x1, primary_rc 0x0003,
secondary_rc 0x00000004
DBCONN-APPC-612C2B28: Verb block =
DBCONN-APPC-612C2B28:   0001 0200 0003 0000  0000 0004 0020 100C
DBCONN-APPC-612C2B28:   610A 828B 0000 0000  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   0000 0000 8014 0003  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   D3E4 F6F2 E2E3 C1D9  C4C2 F240 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  0200 0000 0000 0000
DBCONN-APPC-612C2B28:   0000 0000 D4C5 D9D9  C9C5 4040 4040 D7C5
DBCONN-APPC-612C2B28:   E3C5 D940 4040 4040  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   00E2 E3C1 D9E6 4BE3  D6D9 C3C8 4040 4040
DBCONN-APPC-612C2B28:   4040 0000 0000 0000  0000 0000
DBCONN-APPC-612C2B28: ALLOCATE verb block =
DBCONN-APPC-612C2B28:   0001 0200 0003 0000  0000 0004 0020 100C
DBCONN-APPC-612C2B28:   610A 828B 0000 0000  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   0000 0000 8014 0003  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   D3E4 F6F2 E2E3 C1D9  C4C2 F240 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  0200 0000 0000 0000
```

You can use the **debug appn** command to obtain more information.

The following message is displayed if a database connection is manually cleared and an outstanding APPC verb is pending:

```
DBCONN-APPC-%612C2B28: Canceling pending APPC verb 0x1
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn event	Displays trace or error messages for Database Connection events.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn config

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

debug dbconn config

no debug dbconn config

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Most of the messages for Database Connection and control blocks do not report any errors. If a connection is inactive and cannot be cleared, use this command with the **debug dbconn appc, debug dbconn tcp**, and **debug appn** commands to locate the problem. The alias_cp_name field must match the configured APPN cpname.

Examples

The following is sample output from the **debug dbconn config** command:

```
DBCONN-CONFIG: alias_cp_name is "ASH      "
DBCONN-CONFIG: connection 612BDAAC matching server on 198.147.235.5:0 with
rdbname=STELLA
DBCONN-CONFIG: APPN shutdown; clearing connection 1234abcd
DBCONN-CONFIG: created server 612C2720
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: new connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 accepts connection 612BDAAC
DBCONN-CONFIG: server 60F74614 takes connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 releases connection 612BDAAC
DBCONN-CONFIG: server 60F74614 releases connection 612BDAAC
DBCONN-CONFIG: deleting connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 abandons connection 612BDAAC
DBCONN-CONFIG: server 612C2720 abandons connection 612BDAAC
DBCONN-CONFIG: deleting server 612C2720
DBCONN-CONFIG: daemon 60381738 takes zombie connection 612BDAAC
DBCONN-CONFIG: daemon 60381738  releases zombie connection 612BDAAC
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn event	Displays trace or error messages for Database Connection events.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn drda

To display error messages and stream traces for DRDA, use the **debug dbconn drda** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug dbconn drda

no debug dbconn drda

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.0(5)XN	This command was moved from the CDBC feature to the CTRC feature.

Examples

The following example displays output from the **debug dbconn drda** command:

```
Router# debug dbconn drda

*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: DSS X'006CD0410001', length 108, in chain,
REQDSS, correlator 1
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'00661041', length 98, code point
X'1041'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0020115E' in COLLECTION X'1041',
length 28, code point X'115E'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'000C116D' in COLLECTION X'1041',
length 8, code point X'116D'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0013115A' in COLLECTION X'1041',
length 15, code point X'115A' (skipping...)
```

Related Commands

Command	Description
debug dbconn all	Displays all CTRC debugging information related to communications with DB2.
debug dbconn appc	Displays APPC-related trace or error messages for communications with DB2.
debug dbconn config	Displays trace or error messages for CTRC configuration and control blocks for DB2 communications.
debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
debug dbconn tcp	Displays error messages or traces for TCP/IP communications with DB2.
debug snasw	Displays debugging information related to SNA Switching Services.

debug dbconn event

To display trace or error messages for CTRC events related to DB2 communications, use the **debug dbconn event** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug dbconn event

no debug dbconn event

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.0(5)XN	This command was moved from the CDBC feature to the CTRC feature.

Examples

The following examples display output from the **debug dbconn event** command in a variety of situations. A normal trace for the **debug dbconn event** displays as follows:

```
Router# debug dbconn event

DBCONN-EVENT: Dispatch to 60FD6C00, from 0, msg 60F754CC, msgid 6468 'dh',
buffer 0.
DBCONN-EVENT: [*] Post to 61134240(cn), from 60EC5470(tc), msg 611419E4,
msgid 0x6372 'cr', buffer 612BF68C.
DBCONN-EVENT: Flush events called for pto 61182742, pfrom 61239837.
DBCONN-EVENT: Event discarded: to 61182742 (cn), from 61239837(ap), msg
61339273, msgid 0x6372 'cr' buffer 0.
DBCONN-EVENT: == Send to 1234abcd, from 22938acd, msg 72618394, msgid
0x6372 'cr', buffer 0.
```

If the following messages are displayed, contact Cisco technical support personnel:

```
DBCONN-TCPFSM-1234abcd: Cannot occur in state 2 on input 6363 ('cc')
DBCONN-APPCFSM-1234abcd: Cannot occur in state 3 on input 6363 ('cc')
```

Related Commands

Command	Description
debug dbconn all	Displays all CTRC debugging information related to communications with DB2.
debug dbconn appc	Displays APPC-related trace or error messages for communications with DB2.
debug dbconn config	Displays trace or error messages for CTRC configuration and control blocks for DB2 communications.
debug dbconn drda	Displays error messages or stream traces for DRDA communications with DB2.
debug dbconn tcp	Displays error messages or traces for TCP/IP communications with DB2.
debug snasw	Displays debugging information related to SNA Switching Services.
show debugging	Displays the state of each debugging option.

debug dbconn tcp

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

debug dbconn tcp

no debug dbconn tcp

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled for the dbconn subsystem.

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.0(5)XN	This command was moved from the CDBC feature to the CTRC feature.

Examples

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

```
Router# debug dbconn tcp

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

Related Commands

Command	Description
debug dbconn all	Displays all CTRC debugging information related to communications with DB2.
debug dbconn appc	Displays APPC-related trace or error messages for communications with DB2.
debug dbconn config	Displays trace or error messages for CTRC configuration and control blocks for DB2 communications.
debug dbconn drda	Displays error messages or stream traces for DRDA communications with DB2.

Command	Description
debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
debug ip tcp	Displays debugging information related to TCP/IP.
debug snasw	Displays debugging information related to SNA Switching Services.
show debugging	Displays the state of each debugging option.

debug decnet adj

To display debugging information on DECnet adjacencies, use the **debug decnet adj** privileged EXEC command. The **no** form of this command disables debugging output.

debug decnet adj

no debug decnet adj

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the sample **debug decnet adj** command:

```
Router# debug decnet adj

DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: sending hellos
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: 1.5 adjacency initializing
DNET-ADJ: sending triggered hellos
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
DNET-ADJ: Level 1 hello from 1.3
DNET-ADJ: 1.5 adjacency up
DNET-ADJ: Level 1 hello from 1.5
DNET-ADJ: 1.5 adjacency down, listener timeout
```

The following line indicates that the router is sending hello messages to all routers on this segment, which in this case is Ethernet 0:

```
DNET-ADJ: Sending hellos to all routers on interface Ethernet0, blksize 1498
```

The following line indicates that the router has heard a hello message from address 1.5 and is creating an adjacency entry in its table. The initial state of this adjacency will be *initializing*.

```
DNET-ADJ: 1.5 adjacency initializing
```

The following line indicates that the router is sending an unscheduled (triggered) hello message as a result of some event, such as new adjacency being heard:

```
DNET-ADJ: sending triggered hellos
```

The following line indicates that the adjacency with 1.5 is now up, or active:

```
DNET-ADJ: 1.5 adjacency up
```

The following line indicates that the adjacency with 1.5 has timed out, because no hello message has been heard from adjacency 1.5 in the time interval originally specified in the hello message from 1.5:

```
DNET-ADJ: 1.5 adjacency down, listener timeout
```

The following line indicates that the router is sending an unscheduled hello message, as a result of some event, such as the adjacency state changing:

```
DNET-ADJ: hello update triggered by state changed in dn_add_adjacency
```

debug decnet connects

To display debugging information of all connect packets that are filtered (permitted or denied) by DECnet access lists, use the **debug decnet connects** privileged EXEC command. The **no** form of this command disables debugging output.

debug decnet connects

no debug decnet connects

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

When you use connect packet filtering, it may be helpful to use the **decnet access-group** configuration command to apply the following basic access list:

```
access-list 300 permit 0.0 63.1023 eq any
```

You can then log all connect packets sent on interfaces to which you applied this list, in order to determine those elements on which your connect packets must be filtered.

Note

Packet password and account information is not logged in the **debug decnet connects** message, nor is it displayed by the **show access** EXEC command. If you specify **password** or **account** information in your access list, they can be viewed by anyone with access to the configuration of the router.

Examples

The following is sample output from the **debug decnet connects** command:

```
Router# debug decnet connects

DNET-CON: list 300 item #2 matched src=19.403 dst=19.309 on Ethernet0: permitted
 srcname="RICK" srcuic=[0,017]
 dstobj=42 id="USER"
```

Table 37 describes significant fields in the output.

Table 37 debug decnet connects Field Descriptions

Field	Description
DNET-CON:	Indicates that this is a **debug decnet connects** packet.
list 300 item #2 matched	Indicates that a packet matched the second item in access list 300.
src=19.403	Indicates the source DECnet address for the packet.
dst=19.309	Indicates the destination DECnet address for the packet.
on Ethernet0:	Indicates the router interface on which the access list filtering the packet was applied.
permitted	Indicates that the access list permitted the packet.
srcname = “RICK”	Indicates the originator user of the packet.
srcuic=[0,017]	Indicates the source UIC of the packet.

Table 37 debug decnet connects Field Descriptions (continued)

Field	Description
dstobj=42	Indicates that DECnet object 42 is the destination.
id="USER"	Indicates the access user.

debug decnet events

To display debugging information on DECnet events, use the **debug decnet events** privileged EXEC command. The **no** form of this command disables debugging output.

debug decnet events

no debug decnet events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug decnet events

DNET: Hello from area 50 rejected - exceeded ‘max area' parameter (45)
DNET: Hello from area 50 rejected - exceeded ‘max area' parameter (45)
```

The following line indicates that the router received a hello message from a router whose area was greater than the max-area parameter with which this router was configured:

```
DNET: Hello from area 50 rejected - exceeded'max area' parameter (45)
```

The following line indicates that the router received a hello message from a router whose node ID was greater than the max-node parameter with which this router was configured:

```
DNET: Hello from node 1002 rejected - exceeded'max node' parameter (1000)
```

debug decnet packet

To display debugging information on DECnet packet events, use the **debug decnet packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug decnet packet

no debug decnet packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug decnet packet

DNET-PKT: src 1.4 dst 1.5 sending to PHASEV
DNET-PKT: Packet fwded from 1.4 to 1.5, via 1.5, snpa 0000.3080.cf90, TokenRing0
```

The following line indicates that the router is sending a converted packet addressed to node 1.5 to Phase V:

```
DNET-PKT: src 1.4 dst 1.5 sending to PHASEV
```

The following line indicates that the router forwarded a packet from node 1.4 to node 1.5. The packet is being sent to the next hop of 1.5 whose subnetwork point of attachment (MAC address) on that interface is 0000.3080.cf90.

```
DNET-PKT: Packet fwded from 1.4 to 1.5, via 1.5, snpa 0000.3080.cf90, TokenRing0
```

debug decnet routing

To display all DECnet routing-related events occurring at the router, use the **debug decnet routing** privileged EXEC command. The **no** form of this command disables debugging output.

debug decnet routing

no debug decnet routing

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug decnet routing

DNET-RT: Received level 1 routing from 1.3 on Ethernet0 at 1:16:34
DNET-RT: Sending routes
DNET-RT: Sending normal routing updates on Ethernet0
DNET-RT: Sending level 1 routing updates on interface Ethernet0
DNET-RT: Level1 routes from 1.5 on Ethernet0: entry for node 5 created
DNET-RT: route update triggered by after split route pointers in dn_rt_input
DNET-RT: Received level 1 routing from 1.5 on Ethernet 0 at 1:18:35
DNET-RT: Sending L1 triggered routes
DNET-RT: Sending L1 triggered routing updates on Ethernet0
DNET-RT: removing route to node 5
```

The following line indicates that the router has received a level 1 update on Ethernet interface 0:

```
DNET-RT: Received level 1 routing from 1.3 on Ethernet0 at 1:16:34
```

The following line indicates that the router is sending its scheduled updates on Ethernet interface 0:

```
DNET-RT: Sending normal routing updates on Ethernet0
```

The following line indicates that the route will send an unscheduled update on this interface as a result of some event. In this case, the unscheduled update is a result of a new entry created in the routing table of the interface.

```
DNET-RT: route update triggered by after split route pointers in dn_rt_input
```

The following line indicates that the router sent the unscheduled update on Ethernet 0:

```
DNET-RT: Sending L1 triggered routes
DNET-RT: Sending L1 triggered routing updates on Ethernet0
```

The following line indicates that the router removed the entry for node 5 because the adjacency with node 5 timed out, or the route to node 5 through a next-hop router was disconnected:

```
DNET-RT: removing route to node 5
```

debug dhcp

To display debugging information about the Dynamic Host Configuration Protocol (DHCP) client activities and to monitor the status of DHCP packets, use the **debug dhcp** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug dhcp [**detail**]

no debug dhcp [**detail**]

Syntax Description

detail	(Optional) Displays additional debug information.

Usage Guidelines

You can also use the **debug dhcp** command to monitor the subnet allocation and releasing for on-demand address pools.

For debugging purposes, the **debug dhcp detail** command provides the most useful information such as the lease entry structure of the client and the state transitions of the lease entry. The debug output shows the scanned option values from received DHCP messages that are replies to a router request. The values of the op, htype, hlen, hops, server identifier option, xid, secs, flags, ciaddr, yiaddr, siaddr, and giaddr fields of the DHCP packet are shown in addition to the length of the options field.

Examples

The following examples show and explain some of the typical debug messages you might see when using the **debug dhcp detail** command.

The following example shows the debug output when a DHCP client sends out a DHCPDISCOVER broadcast message to find its local DHCP server:

```
Router# debug dhcp detail
00:07:16:DHCP:DHCP client process started:10
00:07:16:RAC:Starting DHCP discover on Ethernet2
00:07:16:DHCP:Try 1 to acquire address for Ethernet2
00:07:16:%SYS-5-CONFIG_I:Configured from console by console
00:07:19:DHCP:Shutting down from get_netinfo()
00:07:19:DHCP:Attempting to shutdown DHCP Client
00:07:21:DHCP:allocate request
00:07:21:DHCP:new entry. add to queue
00:07:21:DHCP:SDiscover attempt # 1 for entry:
```

The first seven lines of the following output show the current values stored in the lease entry structure for the client:

```
00:07:21:Temp IP addr:0.0.0.0  for peer on Interface:Ethernet2
00:07:21:Temp  sub net mask:0.0.0.0
00:07:21: DHCP Lease server:0.0.0.0, state:1 Selecting
00:07:21: DHCP transaction id:582
00:07:21: Lease:0 secs,  Renewal:0 secs,  Rebind:0 secs
00:07:21: Next timer fires after:00:00:03
00:07:21: Retry count:1   Client-ID:cisco-0010.7b6e.afd8-Et2
00:07:21:DHCP:SDiscover:sending 308 byte length DHCP packet
00:07:21:DHCP:SDiscover 308 bytes
00:07:21: B'cast on Ethernet2 interface from 0.0.0.0
```

The following example shows the offered addresses and parameters sent to the DHCP client by the DHCP server via a DHCPOFFER message. The messages containing the field "Scan" indicate the options that were scanned from the received BOOTP packet and the corresponding values.

```
00:07:23:DHCP:Received a BOOTREP pkt
00:07:23:DHCP:Scan:Message type:DHCP Offer
00:07:23:DHCP:Scan:Server ID Option:10.1.1.1 = A010101
00:07:23:DHCP:Scan:Lease Time:180
00:07:23:DHCP:Scan:Renewal time:90
00:07:23:DHCP:Scan:Rebind time:157
00:07:23:DHCP:Scan:Subnet Address Option:255.255.255.0
```

The following debug output shows selected fields in the received BOOTP packet:

```
00:07:23:DHCP:rcvd pkt source:10.1.1.1,  destination: 255.255.255.255
00:07:23: UDP  sport:43,  dport:44,  length:308
00:07:23: DHCP op:2, htype:1, hlen:6, hops:0
00:07:23: DHCP server identifier:10.1.1.1
00:07:23:   xid:582, secs:0, flags:8000
00:07:23:   client:0.0.0.0, your:10.1.1.2
00:07:23:   srvr:  0.0.0.0, gw:0.0.0.0
00:07:23:   options block length:60

00:07:23:DHCP Offer Message Offered Address:10.1.1.2
00:07:23:DHCP:Lease Seconds:180 Renewal secs: 90 Rebind secs:157
00:07:23:DHCP:Server ID Option:10.1.1.1
00:07:23:DHCP:offer received from 10.1.1.1
```

The following example shows the debug output when the DHCP client sends out a DHCPREQUEST broadcast message to the DHCP server to accept the offered parameters:

```
00:07:23:DHCP:SRequest attempt # 1 for entry:
00:07:23:Temp IP addr:10.1.1.2  for peer on Interface:Ethernet2
00:07:23:Temp  sub net mask:255.255.255.0
00:07:23:   DHCP Lease server:10.1.1.1, state:2 Requesting
00:07:23:   DHCP transaction id:582
00:07:23:   Lease:180 secs,  Renewal:0 secs,  Rebind:0 secs
00:07:23:   Next timer fires after:00:00:02
00:07:23:   Retry count:1   Client-ID:cisco-0010.7b6e.afd8-Et2
00:07:23:DHCP:SRequest- Server ID option:10.1.1.1
00:07:23:DHCP:SRequest- Requested IP addr option:10.1.1.2
00:07:23:DHCP:SRequest placed lease len option:180
00:07:23:DHCP:SRequest:326 bytes
00:07:23:DHCP:SRequest:326 bytes
00:07:23:  B'cast on Ethernet2 interface from 0.0.0.0
```

The following example shows the debug output when the DHCP server sends a DHCPACK message to the client with the full set of configuration parameters:

```
00:07:23:DHCP:Received a BOOTREP pkt
00:07:23:DHCP:Scan:Message type:DHCP Ack
00:07:23:DHCP:Scan:Server ID Option:10.1.1.1 = A010101
00:07:23:DHCP:Scan:Lease Time:180
00:07:23:DHCP:Scan:Renewal time:90
00:07:23:DHCP:Scan:Rebind time:157
00:07:23:DHCP:Scan:Subnet Address Option:255.255.255.0
00:07:23:DHCP:rcvd pkt source:10.1.1.1,  destination: 255.255.255.255
00:07:23: UDP  sport:43,  dport:44,  length:308
00:07:23: DHCP op:2, htype:1, hlen:6, hops:0
00:07:23: DHCP server identifier:10.1.1.1
00:07:23:   xid:582, secs:0, flags:8000
00:07:23:   client:0.0.0.0, your:10.1.1.2
00:07:23:   srvr:  0.0.0.0, gw:0.0.0.0
00:07:23:   options block length:60
```

```
00:07:23:DHCP Ack Messag
00:07:23:DHCP:Lease Seconds:180 Renewal secs: 90 Rebind secs:157
00:07:23:DHCP:Server ID Option:10.1.1.1Interface Ethernet2 assigned DHCP address 10.1.1.2,
mask 255.255.255.0

00:07:26:DHCP Client Pooling:***Allocated IP address:10.1.1.2
00:07:26:Allocated IP address = 10.1.1.2  255.255.255.0
```

Most fields are self-explanatory; however, fields that may need further explanation are described in Table 38.

Table 38 debug dhcp Command Field Descriptions

Fields	Description
DHCP:Scan:Subnet Address Option:255.255.255.0	Subnet mask option (option 1).
DHCP server identifier:1.1.1.1	Value of the DHCP server id option (option 54). Note that this is not the same as the siaddr field, which is the server IP address.
srvr:0.0.0.0, gw:0.0.0.0	srvr is the value of the siaddr field. gw is the value of the giaddr field.

Related Commands

Command	Description
debug ip dhcp server	Enables DHCP server debugging.
show dhcp lease	Displays DHCP addresses leased from a server.

debug dialer events

To display debugging information about the packets received on a dialer interface, use the **debug dialer events** privileged EXEC command. The **no** form of this command disables debugging output.

debug dialer events

no debug dialer events

Syntax Description

This command has no arguments or keywords.

Examples

When DDR is enabled on the interface, information concerning the cause of any call (called the *Dialing cause*) is displayed. The following line of output for an IP packet lists the name of the DDR interface and the source and destination addresses of the packet:

```
Dialing cause: Serial0: ip (s=172.16.1.111 d=172.16.2.22)
```

The following line of output for a bridged packet lists the DDR interface and the type of packet (in hexadecimal). For information on these packet types, see the "Ethernet Type Codes" appendix of the *Cisco IOS Bridging and IBM Networking Command Reference* publication.

```
Dialing cause: Serial1: Bridge (0x6005)
```

Most messages are self-explanatory; however, messages that may need some explanation are described in Table 39.

Table 39 General debug dialer events Message Descriptions

Message	Description
Dialer0: Already *xxx* call(s) in progress on Dialer0, dialing not allowed	Number of calls in progress (*xxx*) exceeds the maximum number of calls set on the interface.
Dialer0: No free dialer - starting fast idle timer	All the lines in the interface or rotary group are busy, and a packet is waiting to be sent to the destination.
BRI0: rotary group to *xxx* overloaded (*yyy*)	Number dialer (*xxx*) exceeds the load set on the interface (*yyy*).
BRI0: authenticated host *xxx* with no matching dialer profile	No dialer profile matches *xxx*, the CHAP name or remote name of the remote host.
BRI0: authenticated host *xxx* with no matching dialer map	No dialer map matches *xxx*, the CHAP name or remote name of the remote host.
BRI0: Can't place call, verify configuration	Dialer string or dialer pool on an interface not set.

Table 40 describes the messages that the **debug dialer events** command can generate for a serial interface used as a V.25*bis* dialer for DDR.

Table 40 debug dialer events Message Descriptions for DDR

Message	Description
Serial 0: Dialer result = *xxxxxxxxxx*	Result returned from the V.25*bis* dialer. It is useful in debugging if calls are failing. On some hardware platforms, this message cannot be displayed due to hardware limitations. Possible values for the *xxxxxxxxxx* variable depend on the V.25*bis* device with which the router is communicating.
Serial 0: No dialer string defined. Dialing cannot occur.	Packet is received that should cause a call to be placed. However, no dialer string is configured, so dialing cannot occur. This message usually indicates a configuration problem.
Serial 0: Attempting to dial *xxxxxxxxxx*	Packet has been received that passes the dial-on-demand access lists. That packet causes phone number *xxxxxxxxxx* to be dialed.
Serial 0: Unable to dial *xxxxxxxxxx*	Phone call to *xxxxxxxxxx* cannot be placed. This failure might be due to a lack of memory, full output queues, or other problems.
Serial 0: disconnecting call	Router hangs up a call.
Serial 0: idle timeout Serial 0: re-enable timeout Serial 0: wait for carrier timeout	One of these three messages is displayed when a dialer timer expires. These messages are mostly informational, but are useful for debugging a disconnected call or call failure.

Related Commands

Command	Description
debug dialer packets	Displays debugging information about the packets received on a dialer interface.

debug dialer forwarding

To display debugging information about the control plane at the home gateway (HGW) for Layer 2 Tunneling Protocol (L2TP) dialout, use the **debug dialer forwarding** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug dialer forwarding

no debug dialer forwarding

Syntax Description

This command has no keywords or arguments.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2 T	This command was introduced.

Usage Guidelines

Use the **debug dialer forwarding** command to configure a virtual private dialout network (VPDN) on the HGW and a network access server (NAS) to dial from the HGW to the client.

An L2TP tunnel is created between the HGW and the NAS and the packets are forwarded transparently at the NAS.

Examples

The following is sample output from the **debug dialer forwarding** command for dialing from the HGW to the client.

Note DDR-FWD is **debug dialer forwarding** information. (DDR= dial-on-demand routing.)

```
Router# debug dialer forwarding

Dialer forwarding events debugging is on

Router# ping

Protocol [ip]:
Target IP address:1.1.1.3
Repeat count [5]:1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 1.1.1.3, timeout is 2 seconds:
```

```
1d00h:Vi3 DDR-FWD 83093A60:event [REQUEST]  state before [IDLE]
1d00h:Vi3 DDR-FWD 83093A60:VPN Authorization started
1d00h:Vi3 DDR-FWD 83093A60:VPN author result 1
1d00h:Vi3 DDR-FWD 83093A60:event [AUTHOR FOUND]  state before [AUTHORIZING]
1d00h:Vi3 DDR-FWD 83093A60:event [FORWARDED]  state before [FORWARDING]
1d00h:Vi3 DDR-FWD 83093A60:Connection is up, start LCP now
*Mar  2 00:31:33:%LINK-3-UPDOWN:Interface Virtual-Access3, changed state to up.
Success rate is 0 percent (0/1)
R2604#
*Mar  2 00:31:35:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access3, changed
state to up
Router#
```

Outgoing call disconnected:

```
Router#
1d00h:Vi3 DDR-FWD 83093A60:event [VPDN DISC]  state before [FORWARDED]
*Mar  2 00:33:33:%LINK-3-UPDOWN:Interface Virtual-Access3, changed state to down
*Mar  2 00:33:34:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access3, changed
state to down
```

Related Commands

Command	Description
debug dialer events	Displays debugging information about events on a dialer interface.
debug dialer packets	Displays debugging information about packets received on a dialer interface.

debug dialer map

To display debugging information about the creation and deletion of dynamic dialer maps, use the **debug dialer map** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug dialer map

no debug dialer map

Syntax Description

This command has no keywords or arguments.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(5.1)	This command was introduced.

Usage Guidelines

Use the **debug dialer map** command to track large-scale dialout (LSDO) and incoming calls that use dynamic dialer maps. This command shows the whole trace including when the map is created and removed.

If an interface is configured for dial-on-demand routing (DDR), and a map to a specified address does not exist, then a dynamic dialer map is created and when the call disconnects, the dialer map is removed.

Note Do not configure a dialer string or a dialer map on the incoming interface.

Examples

In the following sample output from the **debug dialer map** command, a dialer map is created when an incoming call is connected and removed when that call is disconnected:

```
Router# debug dialer map

Dial on demand dynamic dialer maps debugging is on
```

Incoming call connected:

```
Router#
*Mar 22 12:19:15.597:%LINK-3-UPDOWN:Interface BRI0/0:1, changed state to up
*Mar 22 12:19:17.748:BR0/0:1 DDR:dialer_create_dynamic_map map created for 11.0.0.1
*Mar 22 12:19:18.734:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRI0/0:1, changed
state to up
*Mar 22 12:19:21.598:%ISDN-6-CONNECT:Interface BRI0/0:1 is now connected to unknown R2604
```

Incoming call disconnected:

```
Router#
*Mar 22 12:21:15.597:%ISDN-6-DISCONNECT:Interface BRI0/0:1  disconnected from  R2604, call
lasted 120 seconds
*Mar 22 12:21:15.645:%LINK-3-UPDOWN:Interface BRI0/0:1, changed state to down
*Mar 22 12:21:15.649:BR0/0:1 DDR:dialer_remove_dynamic_map map 11.0.0.1 removed
*Mar 22 12:21:16.647:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRI0/0:1, changed
state to down
```

Related Commands

Command	Description
debug dialer events	Displays debugging information about events on a dialer interface.
debug dialer packets	Displays debugging information about packets received on a dialer interface.

debug dlsw

To enable debugging of DLSw+, use the **debug dlsw** privileged EXEC command. The **no** form of this command disables debugging output.

debug dlsw [**border-peers** [**interface** *interface* | **ip address** *ip-address*] | **core** [**flow-control messages** | **state** | **xid**] [*circuit-number*] | **local-circuit** *circuit-number* | **peers** [**interface** *interface* [**fast-errors** | **fast-paks**] | **ip address** *ip-address* [**fast-errors** | **fast-paks** | **fst-seq** | **udp**]] | **reachability** [**error** | **verbose**] [**sna** | **netbios**]

no debug dlsw [**border-peers** [**interface** *interface* | **ip address** *ip-address*] | **core** [**flow-control messages** | **state** | **xid**] [*circuit-number*] | **local-circuit** *circuit-number* | **peers** [**interface** *interface* [**fast-errors** | **fast-paks**] | **ip address** *ip-address* [**fast-errors** | **fast-paks** | **fst-seq** | **udp**]] | **reachability** [**error** | **verbose**] [**sna** | **netbios**]

Syntax Description

border-peers	(Optional) Enables debugging output for border peer events.
interface *interface*	(Optional) Specifies a remote peer to debug by a direct interface.
ip address *ip-address*	(Optional) Specifies a remote peer to debug by its IP address.
core	(Optional) Enables debugging output for DLSw core events.
flow-control	(Optional) Enables debugging output for congestion in the WAN or at the remote end station.
messages	(Optional) Enables debugging output of core messages—specific packets received by DLSw either from one of its peers or from a local medium via the Cisco link services interface.
state	(Optional) Enables debugging output for state changes on the circuit.
xid	(Optional) Enables debugging output for the exchange identification state machine.
*circuit-numbe*r	(Optional) Specifies the circuit for which you want core debugging output to reduce the of output.
local-circuit *circuit-number*	(Optional) Enables debugging output for circuits performing local conversion. Local conversion occurs when both the input and output data-link connections are on the same local peer and no remote peer exists.
peers	(Optional) Enables debugging output for peer events.
fast-errors	(Optional) Debugs errors for fast-switched packets.
fast-paks	(Optional) Debugs fast-switched packets.
fst-seq	(Optional) Debugs FST sequence numbers on fast switched packets.
udp	(Optional) Debugs UDP packets.
reachability	(Optional) Enables debugging output for reachability events (explorer traffic). If no options are specified, event-level information is displayed for all protocols.

error | **verbose**	(Optional) Specifies how much reachability information you want displayed. The **verbose** keyword displays everything, including errors and events. The **error** keyword displays error information only. If no option is specified, event-level information is displayed.
sna | **netbios**	(Optional) Specifies that reachability information be displayed for only SNA or NetBIOS protocols. If no option is specified, information for all protocols is displayed.

Usage Guidelines

When you specify no optional keywords, the **debug dlsw** command enables all available DLSw debugging output.

Normally you need to use only the **error** or **verbose** option of the **debug dlsw reachability** command to help identify problems. The **error** option is recommended for use by customers and provides a subset of the messages from the normal event-level debugging. The **verbose** option provides a very detailed view of events, and is typically used only by service personnel.

To reduce the amount of debug information displayed, use the **sna** or **netbios** option with the **debug dlsw reachability** command if you know that you have an SNA or NetBIOS problem.

The DLSw core is the engine that is responsible for the establishment and maintenance of remote circuits. If possible, specifying the index of the specific circuit you want to debug reduces the amount of output displayed. However, if you want to watch a circuit initially come up, do not use the *circuit-number* option with the **core** keyword.

The **core flow-control** option provides information about congestion in the WAN or at the remote end station. In these cases, DLSw sends Receiver Not Ready (RNR) frames on its local circuits, slowing data traffic on established sessions and giving the congestion an opportunity to clear.

The **core state** option allows you to see when the circuit changes state. This capability is especially useful for determining why a session cannot be established or why a session is being disconnected.

The **core XID** option allows you to track the XID-state machine. The router tracks XID commands and responses used in negotiations between end stations before establishing a session.

Examples

The following examples show and explain some of the typical DLSw debug messages you might see when using the **debug dlsw** command.

The following example enables UDP packet debugging for a specific remote peer:

```
Router# debug dlsw peers ip-address 1.1.1.6 udp
```

The following message is sample output from the **debug dlsw border-peers** command:

```
*Mar 10 17:39:56: CSM: delete group mac cache for group 0
*Mar 10 17:39:56: CSM: delete group name cache for group 0
*Mar 10 17:40:19: CSM: update group cache for mac 0000.3072.1070, group 10
*Mar 10 17:40:22: DLSw: send_to_group_members(): copy to peer 10.19.32.5
```

The following message is from a router that initiated a TCP connection:

```
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLSw: dtp_action_a() attempting to connect peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:DISCONN->WAIT_WR
DLSw: Async Open Callback 10.3.8.7(2065) -> 11002
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-WR PIPE OPENED state:WAIT_WR
DLSw: dtp_action_f() start read open timer for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_WR->WAIT_RD
DLSw: passive open 10.3.8.7(11004) -> 2065
```

```
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-RD PIPE OPENED state:WAIT_RD
DLSw: dtp_action_g() read pipe opened for peer 10.3.8.7(2065)
DLSw: CapExId Msg sent to peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_RD->WAIT_CAP
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.3.8.7(2065)
DLSw: Recv CapExId Msg from peer 10.3.8.7(2065)
DLSw: Pos CapExResp sent to peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.3.8.7(2065)
DLSw: Recv CapExPosRsp Msg from peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dtp_action_k() cap xchged for peer 10.3.8.7(2065)
DLSw: closing read pipe tcp connection for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:WAIT_CAP->PCONN_WT
DLSw: Processing delayed event:TCP-PEER CONNECTED - prev state:PCONN_WT
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:TCP-PEER CONNECTED state:PCONN_WT
DLSw: dtp_action_m() peer connected for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:PCONN_WT->CONNECT
DLSw: START-TPFSM (peer 10.3.8.7(2065)): event:CORE-ADD CIRCUIT state:CONNECT
DLSw: dtp_action_u(), peer add circuit for peer 10.3.8.7(2065)
DLSw: END-TPFSM (peer 10.3.8.7(2065)): state:CONNECT->CONNECT
```

The following message is from a router that received a TCP connection:

```
DLSw: passive open 10.10.10.4(11002) -> 2065
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-RD PIPE OPENED state:DISCONN
DLSw: dtp_action_c() opening write pipe for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:DISCONN->WWR_RDOP
DLSw: Async Open Callback 10.10.10.4(2065) -> 11004
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-WR PIPE OPENED state:WWR_RDOP
DLSw: dtp_action_i() write pipe opened for peer 10.10.10.4(2065)
DLSw: CapExId Msg sent to peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WWR_RDOP->WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLSw: Recv CapExId Msg from peer 10.10.10.4(2065)
DLSw: Pos CapExResp sent to peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dtp_action_j() cap msg rcvd from peer 10.10.10.4(2065)
DLSw: Recv CapExPosRsp Msg from peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dtp_action_k() cap xchged for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:WAIT_CAP->PCONN_WT
DLSw: dlsw_tcpd_fini() for peer 10.10.10.4(2065)
DLSw: dlsw_tcpd_fini() closing write pipe for peer 10.10.10.4
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-CLOSE WR PIPE state:PCONN_WT
DLSw: dtp_action_l() close write pipe for peer 10.10.10.4(2065)
DLSw: closing write pipe tcp connection for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:PCONN_WT->PCONN_WT
DLSw: Processing delayed event:TCP-PEER CONNECTED - prev state:PCONN_WT
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:TCP-PEER CONNECTED state:PCONN_WT
DLSw: dtp_action_m() peer connected for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:PCONN_WT->CONNECT
DLSw: START-TPFSM (peer 10.10.10.4(2065)): event:CORE-ADD CIRCUIT state:CONNECT
DLSw: dtp_action_u(), peer add circuit for peer 10.10.10.4(2065)
DLSw: END-TPFSM (peer 10.10.10.4(2065)): state:CONNECT->CONNECT
```

The following message is from a router that initiated an FST connection:

```
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:ADMIN-OPEN CONNECTION state:DISCONN
DLSw: dfstp_action_a() attempting to connect peer 10.10.10.4(0)
DLSw: Connection opened for peer 10.10.10.4(0)
DLSw: CapExId Msg sent to peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:DISCONN->WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dfstp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLSw: Recv CapExPosRsp Msg from peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dfstp_action_e() cap msg rcvd for peer 10.10.10.4(0)
DLSw: Recv CapExId Msg from peer 10.10.10.4(0)
DLSw: Pos CapExResp sent to peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-FSTPFSM (peer 10.10.10.4(0)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dfstp_action_f() cap xchged for peer 10.10.10.4(0)
DLSw: END-FSTPFSM (peer 10.10.10.4(0)): state:WAIT_CAP->CONNECT
```

The following message is from a router that received an FST connection:

```
DLSw: START-FSTPFSM (peer 10.3.8.7(0)): event:SSP-CAP MSG RCVD state:DISCONN
DLSw: dfstp_action_c() cap msg rcvd for peer 10.3.8.7(0)
DLSw: Recv CapExId Msg from peer 10.3.8.7(0)
DLSw: Pos CapExResp sent to peer 10.3.8.7(0)
DLSw: CapExId Msg sent to peer 10.3.8.7(0)
DLSw: END-FSTPFSM (peer 10.3.8.7(0)): state:DISCONN->WAIT_CAP
DLSw: START-FSTPFSM (peer 10.3.8.7(0)): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dfstp_action_e() cap msg rcvd for peer 10.3.8.7(0)
DLSw: Recv CapExPosRsp Msg from peer 10.3.8.7(0)
DLSw: END-FSTPFSM (peer 10.3.8.7(0)): state:WAIT_CAP->WAIT_CAP
DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-FSTPFSM (peer 10.3.8.7(0)): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dfstp_action_f() cap xchged for peer 10.3.8.7(0)
DLSw: END-FSTPFSM (peer 10.3.8.7(0)): state:WAIT_CAP->CONNECT
```

The following message is from a router that initiated an LLC2 connection:

```
DLSw-LLC2: Sending enable port ; port no : 0
           PEER-DISP Sent : CLSI Msg : ENABLE.Req   dlen: 20
DLSw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20
DLSw-LLC2 : Sending activate sap for Serial1 - port_id = 887C3C
            port_type = 7 dgra(UsapID) = 952458
            PEER-DISP Sent : CLSI Msg : ACTIVATE_SAP.Req   dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLSw Got ActSapcnf back for Serial1 - port_id = 8978204, port_type = 7, psap_id = 0

DLSw: START-LLC2PFSM (peer on interface Serial1): event:ADMIN-OPEN CONNECTION
state:DISCONN
DLSw: dllc2p_action_a() attempting to connect peer on interface Serial1
 PEER-DISP Sent : CLSI Msg : REQ_OPNSTN.Req   dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial1): state:DISCONN->ROS_SENT

DLSw: Peer Received : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLSw: START-LLC2PFSM (peer on interface Serial1): event:CLS-REQOPNSTN.CNF state:ROS_SENT
DLSw: dllc2p_action_c()
 PEER-DISP Sent : CLSI Msg : CONNECT.Req   dlen: 16
DLSw: END-LLC2PFSM (peer on interface Serial1): state:ROS_SENT->CON_PEND

DLSw: Peer Received : CLSI Msg : CONNECT.Cfm CLS_OK dlen: 28
DLSw: START-LLC2PFSM (peer on interface Serial1): event:CLS-CONNECT.CNF state:CON_PEND
DLSw: dllc2p_action_e() send capabilities to peer on interface Serial1
 PEER-DISP Sent : CLSI Msg : SIGNAL_STN.Req   dlen: 8
```

```
 PEER-DISP Sent : CLSI Msg : DATA.Req   dlen: 418
DLSw: CapExId Msg sent to peer on interface Serial1
DLSw: END-LLC2PFSM (peer on interface Serial1): state:CON_PEND->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind   dlen: 418
DLSw: START-LLC2PFSM (peer on interface Serial1): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial1
DLSw: Recv CapExId Msg from peer on interface Serial1
 PEER-DISP Sent : CLSI Msg : DATA.Req   dlen: 96
DLSw: Pos CapExResp sent to peer on interface Serial1
DLSw: END-LLC2PFSM (peer on interface Serial1): state:WAIT_CAP->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind   dlen: 96
DLSw: START-LLC2PFSM (peer on interface Serial1): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial1
DLSw: Recv CapExPosRsp Msg from peer on interface Serial1
DLSw: END-LLC2PFSM (peer on interface Serial1): state:WAIT_CAP->WAIT_CAP

DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-LLC2PFSM (peer on interface Serial1): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dllc2p_action_l() cap xchged for peer on interface Serial1
DLSw: END-LLC2PFSM (peer on interface Serial1): state:WAIT_CAP->CONNECT
```

The following message is from a router that received an LLC2 connection:

```
DLSw-LLC2: Sending enable port ; port no : 0
 PEER-DISP Sent : CLSI Msg : ENABLE.Req   dlen: 20
DLSw: Peer Received : CLSI Msg : ENABLE.Cfm CLS_OK dlen: 20
DLSw-LLC2 : Sending activate sap for Serial0 - port_id = 887C3C
 port_type = 7 dgra(UsapID) = 93AB34
 PEER-DISP Sent : CLSI Msg : ACTIVATE_SAP.Req   dlen: 60
DLSw: Peer Received : CLSI Msg : ACTIVATE_SAP.Cfm CLS_OK dlen: 60
DLSw Got ActSapcnf back for Serial0 - port_id = 8944700, port_type = 7, psap_id = 0

DLSw: Peer Received : CLSI Msg : CONECT_STN.Ind   dlen: 39
DLSw: START-LLC2PFSM (peer on interface Serial0): event:CLS-CONNECT_STN.IND state:DISCONN
DLSw: dllc2p_action_s() conn_stn for peer on interface Serial0
 PEER-DISP Sent : CLSI Msg : REQ_OPNSTN.Req   dlen: 106
DLSw: END-LLC2PFSM (peer on interface Serial0): state:DISCONN->CONS_PEND

DLSw: Peer Received : CLSI Msg : REQ_OPNSTN.Cfm CLS_OK dlen: 106
DLSw: START-LLC2PFSM (peer on interface Serial0): event:CLS-REQOPNSTN.CNF state:CONS_PEND
DLSw: dllc2p_action_h() send capabilities to peer on interface Serial0
 PEER-DISP Sent : CLSI Msg : CONNECT.Rsp   dlen: 20
 PEER-DISP Sent : CLSI Msg : DATA.Req   dlen: 418
DLSw: CapExId Msg sent to peer on interface Serial0
DLSw: END-LLC2PFSM (peer on interface Serial0): state:CONS_PEND->WAIT_CAP

DLSw: Peer Received : CLSI Msg : CONNECTED.Ind   dlen: 8
DLSw: START-LLC2PFSM (peer on interface Serial0): event:CLS-CONNECTED.IND state:WAIT_CAP
DLSw: END-LLC2PFSM (peer on interface Serial0): state:WAIT_CAP->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind   dlen: 418
DLSw: START-LLC2PFSM (peer on interface Serial0): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial0
DLSw: Recv CapExId Msg from peer on interface Serial0
 PEER-DISP Sent : CLSI Msg : DATA.Req   dlen: 96
DLSw: Pos CapExResp sent to peer on interface Serial0
DLSw: END-LLC2PFSM (peer on interface Serial0): state:WAIT_CAP->WAIT_CAP

DLSw: Peer Received : CLSI Msg : DATA.Ind   dlen: 96
DLSw: START-LLC2PFSM (peer on interface Serial0): event:SSP-CAP MSG RCVD state:WAIT_CAP
DLSw: dllc2p_action_k() cap msg rcvd for peer on interface Serial0
DLSw: Recv CapExPosRsp Msg from peer on interface Serial0
```

```
DLSw: END-LLC2PFSM (peer on interface Serial0): state:WAIT_CAP->WAIT_CAP

DLSw: Processing delayed event:SSP-CAP EXCHANGED - prev state:WAIT_CAP
DLSw: START-LLC2PFSM (peer on interface Serial0): event:SSP-CAP EXCHANGED state:WAIT_CAP
DLSw: dllc2p_action_l() cap xchged for peer on interface Serial0
DLSw: END-LLC2PFSM (peer on interface Serial0): state:WAIT_CAP->CONNECT
```

The following messages occur when a CUR_ex (CANUREACH explorer) frame is received from other peers, and the peer statements or the **promiscuous** keyword have not been enabled so that the router is not configured correctly:

```
22:42:44: DLSw: Not promiscuous - Rej conn from 172.20.96.1(2065)
22:42:51: DLSw: Not promiscuous - Rej conn from 172.20.99.1(2065)
```

In the following messages, the router sends a keepalive message every 30 seconds to keep the peer connected. If three keepalive messages are missed, the peer is torn down. These messages are displayed only if keepalives are enabled (by default, keepalives are disabled):

```
22:44:03: DLSw: Keepalive Request sent to peer 172.20.98.1(2065) (168243148)
22:44:03: DLSw: Keepalive Response from peer 172.20.98.1(2065) (168243176)
22:44:34: DLSw: Keepalive Request sent to peer 172.20.98.1(2065) (168274148)
22:44:34: DLSw: Keepalive Response from peer 172.20.98.1(2065) (168274172)
```

The following peer debug messages indicate that the local peer is disconnecting from the specified remote peer because of missed peer keepalives:

```
0:03:24: DLSw: keepalive failure for peer on interface Serial0
0:03:24: DLSw: action_d(): for peer on interface Serial0
0:03:24: DLSW: DIRECT aborting connection for peer on interface Serial0
0:03:24: DLSw: peer on interface Serial0, old state CONNECT, new state DISCONN
```

The following peer debug messages result from an attempt to connect to an IP address that does not have DLSw enabled. The local router attempts to connect in 30-second intervals:

```
23:13:22: action_a() attempting to connect peer 172.20.100.1(2065)
23:13:22: DLSw: CONN: peer 172.20.100.1 open failed, rejected [9]
23:13:22: action_a() retries: 8 next conn time: 861232504
23:13:52: action_a() attempting to connect peer 172.20.100.1(2065)
23:13:52: DLSw: CONN: peer 172.20.100.1 open failed, rejected [9]
23:13:52: action_a() retries: 9 next conn time: 861292536
```

The following peer debug messages that indicates a remote peer statement is missing on the router (address 172.20.100.1) to which the connection attempt is sent:

```
23:14:52: action_a() attempting to connect peer 172.20.100.1(2065)
23:14:52: DLSw: action_a(): Write pipe opened for peer 172.20.100.1(2065)
23:14:52: DLSw: peer 172.20.100.1(2065), old state DISCONN, new state WAIT_RD
23:14:52: DLSw: dlsw_tcpd_fini() closing connection for peer 172.20.100.1
23:14:52: DLSw: action_d(): for peer 172.20.100.1(2065)
23:14:52: DLSw: aborting tcp connection for peer 172.20.100.1(2065)
23:14:52: DLSw: peer 172.20.100.1(2065), old state WAIT_RD, new state DISCONN
```

The following messages show a peer connection opening with no errors or abnormal events:

```
23:16:37: action_a() attempting to connect peer 172.20.100.1(2065)
23:16:37: DLSw: action_a(): Write pipe opened for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state DISCONN, new state WAIT_RD
23:16:37: DLSW: passive open 172.20.100.1(17762) -> 2065
23:16:37: DLSw: action_c(): for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state WAIT_RD, new state CAP_EXG
23:16:37: DLSw: peer 172.20.100.1(2065) conn_start_time set to 861397784
23:16:37: DLSw: CapExId Msg sent to peer 172.20.100.1(2065)
23:16:37: DLSw: Recv CapExId Msg from peer 172.20.100.1(2065)
23:16:37: DLSw: Pos CapExResp sent to peer 172.20.100.1(2065)
```

```
23:16:37: DLSw: action_e(): for peer 172.20.100.1(2065)
23:16:37: DLSw: Recv CapExPosRsp Msg from peer 172.20.100.1(2065)
23:16:37: DLSw: action_e(): for peer 172.20.100.1(2065)
23:16:37: DLSw: peer 172.20.100.1(2065), old state CAP_EXG, new state CONNECT
23:16:37: DLSw: dlsw_tcpd_fini() closing write pipe for peer 172.20.100.1
23:16:37: DLSw: action_g(): for peer 172.20.100.1(2065)
23:16:37: DLSw: closing write pipe tcp connection for peer 172.20.100.1(2065)
23:16:38: DLSw: peer_act_on_capabilities() for peer 172.20.100.1(2065)
```

The following two messages show that an information frame is passing through the router:

```
DLSw: dlsw_tr2fct() lmac:c000.a400.0000 rmac:0800.5a29.75fe ls:5 rs:4 i:34
DLSw: dlsw_tr2fct() lmac:c000.a400.0000 rmac:0800.5a29.75fe ls:4 rs:4 i:34
```

Sample Debug DLSw Reachability Messages

The messages in this section are based on the following criteria:

- Reachability is stored in cache. DLSw+ maintains two reachability caches: one for MAC addresses and one for NetBIOS names. Depending on how long entries have been in the cache, they are either fresh or stale.
- If a router has a fresh entry in the cache for a certain resource, it answers a locate request for that resource without verifying that it is still available. A locate request is typically a TEST frame for MAC addresses or a FIND_NAME_QUERY for NetBIOS.
- If a router has a stale entry in the cache for a certain resource, it verifies that the entry is still valid before answering a locate request for the resource by sending a frame to the last known location of the resource and waits for a resource. If the entry is a REMOTE entry, the router sends a CUR_ex frame to the remote peer to verify. If the entry is a LOCAL entry, it sends either a TEST frame or a NetBIOS FIND_NAME_QUERY on the appropriate local port.
- By default, all reachability cache entries remain fresh for 4 minutes after they are learned. For MAC addresses, you can change this time with the **dlsw timer sna-verify-interval** command. For NetBIOS names, you can change this time with the **dlsw timer netbios-verify-interval** command.
- By default, all reachability cache entries age out of the cache 16 minutes after they are learned. For MAC addresses, you can change this time with the **dlsw timer sna-cache-timeout** command. For NetBIOS names, you can change the time with the **dlsw timer netbios-cache-timeout** command.

Table 41 describes the debug output indicating that the DLSW router received an SSP message that is flow controlled and should be counted against the window of the sender.

```
Dec  6 11:26:49: CSM: Received SSP  CUR   csex flags = 80, mac 4000.90b1.26cf,
The csex flags = 80 means that this is an CUR_ex (explorer).
Dec  5 10:48:33: DLSw: 1620175180 decr r - s:27 so:0 r:27 ro:0
```

Table 41 Debug Output Command Descriptions

Field	Description
decr r	Decrement received count.
s	This DLSW router's granted units for the circuit.
so	0=This DLSW router does not owe a flow control acknowledgment. 1=This router owes a flow control acknowledgment.
r	Partner's number of granted units for the circuit.
ro	Indicates whether the partner owes flow control acknowledgment.

The following message shows that DLSw is sending an I frame to a LAN:

```
Dec  5 10:48:33:  DISP Sent : CLSI Msg : DATA.Req   dlen: 1086
```

The following message shows that DLSw received the I frame from the LAN:

```
Dec  5 10:48:35:  DLSW Received-disp : CLSI Msg : DATA.Ind   dlen: 4
```

The following messages show that the reachability cache is cleared:

```
Router# clear dlsw rea

23:44:11: CSM: Clearing CSM cache
23:44:11: CSM: delete local mac cache for port 0
23:44:11: CSM: delete local name cache for port 0
23:44:11: CSM: delete remote mac cache for peer 0
23:44:11: CSM: delete remote name cash dlsw rea
```

The next group of messages show that the DLSw reachability cache is added, and that a name query is perform from the router Marian:

```
23:45:11: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:11: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:11: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:11: CSM: update local cache for name MARIAN         , port 5EFBB4
23:45:11: CSM: Received CLS_UDATA_STN from Core
23:45:11: CSM: Received netbios frame type A
23:45:11: CSM: Processing Name Query
23:45:11: CSM: Netbios Name Query: ws_status = 6
23:45:11: CSM: Write to peer 0 ok.
23:45:11: CSM: Freeing clsi message
23:45:11: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:11: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:11: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:11: CSM: update local cache for name MARIAN         , port 658AB4
23:45:11: CSM: Received CLS_UDATA_STN from Core
23:45:11: CSM: Received netbios frame type A
23:45:11: CSM: Processing Name Query
23:45:11: CSM: Netbios Name Query: ws_status = 5
23:45:11: CSM: DLXNR_PEND match found.... drop name query
23:45:11: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:12: CSM: update local cache for name MARIAN         , port 5EFBB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:12: CSM: update local cache for name MARIAN         , port 658AB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:12: CSM: update local cache for name MARIAN         , port 5EFBB4
```

```
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:12: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:12: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:12: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:12: CSM: update local cache for name MARIAN         , port 658AB4
23:45:12: CSM: Received CLS_UDATA_STN from Core
23:45:12: CSM: Received netbios frame type A
23:45:12: CSM: Processing Name Query
23:45:12: CSM: Netbios Name Query: ws_status = 5
23:45:12: CSM: DLXNR_PEND match found.... drop name query
23:45:12: CSM: Freeing clsi message
23:45:18: CSM: Deleting Reachability cache
23:45:18: CSM: Deleting DLX NR pending record....
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:38: CSM: update local cache for name MARIAN         , port 5EFBB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:38: CSM: update local cache for name MARIAN         , port 658AB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message
```

The following messages show that the router named Marian is added to the network:

```
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 5EFBB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 5EFBB4
23:45:38: CSM: update local cache for name MARIAN         , port 5EFBB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message
23:45:38: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
23:45:38: CSM: 0800.5a30.7a9b passes local mac excl. filter
23:45:38: CSM: update local cache for mac 0800.5a30.7a9b, port 658AB4
23:45:38: CSM: update local cache for name MARIAN         , port 658AB4
23:45:38: CSM: Received CLS_UDATA_STN from Core
23:45:38: CSM: Received netbios frame type 8
23:45:38: CSM: Write to peer 0 ok.
23:45:38: CSM: Freeing clsi message
```

In the next group of messages, an attempt is made to add the router named Ginger on the Ethernet interface:

```
0:07:44: CSM: core_to_csm CLSI_MSG_PROC - port_id 658AB4
0:07:44: CSM: 0004.f545.24e6 passes local mac excl. filter
0:07:44: CSM: update local cache for mac 0004.f545.24e6, port 658AB4
0:07:44: CSM: update local cache for name GINGER         , port 658AB4
0:07:44: CSM: Received CLS_UDATA_STN from Core
0:07:44: CSM: Received netbios frame type 8
0:07:44: CSM: Write to peer 0 ok.
```

In the following example, the output from the **show dlsw reachability** command indicates that Ginger is on the Ethernet interface and Marian is on the Token Ring interface:

```
Router# show dlsw reachability

DLSw MAC address reachability cache list
Mac Addr        status     Loc.    peer/port           rif
0004.f545.24e6  FOUND      LOCAL   P007-S000    --no rif--
0800.5a30.7a9b  FOUND      LOCAL   P000-S000    06C0.0621.7D00
                                   P007-S000    F0F8.0006.A6FC.005F.F100.0000.0000.0000

DLSw NetBIOS Name reachability cache list
NetBIOS Name    status     Loc.    peer/port           rif
GINGER          FOUND      LOCAL   P007-S000     --no rif--
MARIAN          FOUND      LOCAL   P000-S000     06C0.0621.7D00
                                   P007-S000     --no rif--
```

debug dmsp doc-to-fax

To display debug messages for the doc Media Service Provider TIFF or text2Fax engine, use the **debug dmsp doc-to-fax** EXEC command. To disable the debug messages, use the **no** form of this command.

debug dmsp doc-to-fax [**text-to-fax** | **tiff-reader**]

no debug dmsp doc-to-fax [**text-to-fax** | **tiff-reader**]

Syntax Description

text-to-fax	(Optional) Displays debug messages that occur while the DocMSP Component is receiving text packets and producing T4 fax data.
tiff-reader	(Optional) Displays debug messages that occur while the DocMSP Component is receiving TIFF packets and producing T4 fax data.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug dmsp doc-to-fax** command.

```
Router# debug dmsp doc-to-fax

Jan  1 04:58:39.898: docmsp_call_setup_request: callid=18
Jan  1 04:58:39.902: docmsp_call_setup_request():  ramp data dir=OFFRAMP, conf dir=SRC
Jan  1 04:58:39.902: docmsp_caps_ind: call id=18, src=17
Jan  1 04:58:39.902: docmsp_bridge cfid=5, srccid=18, dstcid=17

Jan  1 04:58:39.902: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC, encode out=2
Jan  1 04:58:39.902: docmsp_rcv_msp_ev: call id =18, evID = 42
Jan  1 04:58:39.902: docmsp_bridge cfid=6, srccid=18, dstcid=15

Jan  1 04:58:39.902: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=DEST, encode out=2
Jan  1 04:58:39.902: docmsp_process_rcv_data: call id src=0, dst=18
Jan  1 04:58:39.902: docmsp_generate_page:
Jan  1 04:58:39.902: docmsp_generate_page: new context for Call 18
Jan  1 04:58:39.922: docmsp_get_msp_event_buffer:
Jan  1 04:58:42.082: docmsp_xmit: call id src=15, dst=18
Jan  1 04:58:42.082: docmsp_process_rcv_data: call id src=15, dst=18
Jan  1 04:58:42.082: offramp_data_process:
Jan  1 04:58:42.102: docmsp_xmit: call id src=15, dst=18
Jan  1 04:58:42.106: docmsp_process_rcv_data: call id src=15, dst=18
Jan  1 04:58:42.106: offramp_data_process:
Jan  1 04:58:42.122: docmsp_xmit: call id src=15, dst=18
Jan  1 04:58:42.126: docmsp_process_rcv_data: call id src=15, dst=18
Jan  1 04:58:42.126: offramp_data_process:
Jan  1 04:58:42.142: docmsp_xmit: call id src=15, dst=18
Jan  1 04:58:42.146: docmsp_xmit: call id src=15, dst=18
```

Related Commands

Command	Description
debug dmsp fax-to-doc	Displays debug messages for the doc Media Service Provider fax-to-doc TIFF engine.

debug dmsp fax-to-doc

To display debug messages for doc MSP fax-to-doc, use the **debug dmsp fax-to-doc** EXEC command. To disable the debug messages, use the **no** form of this command.

debug dmsp fax-to-doc [**tiff-writer**]

no debug dmsp fax-to-doc [**tiff-writer**]

Syntax Description

tiff-writer	(Optional) Displays debug messages that occur while the DocMSP Component is receiving T4 fax data and producing TIFF packets.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug dmsp fax-to-doc** command.

```
Router# debug dmsp fax-to-doc

*Oct 16 08:29:54.487: docmsp_call_setup_request: callid=22
*Oct 16 08:29:54.487: docmsp_call_setup_request():  ramp data dir=OFFRAMP, conf dir=SRC
*Oct 16 08:29:54.487: docmsp_caps_ind: call id=22, src=21
*Oct 16 08:29:54.487: docmsp_bridge cfid=15, srccid=22, dstcid=21

*Oct 16 08:29:54.487: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=SRC, encode out=2
*Oct 16 08:29:54.487: docmsp_bridge cfid=16, srccid=22, dstcid=17

*Oct 16 08:29:54.487: docmsp_bridge(): ramp data dir=OFFRAMP, conf dir=DEST, encode out=2
*Oct 16 08:29:54.487: docmsp_xmit: call id src=17, dst=22
*Oct 16 08:29:54.487: docmsp_process_rcv_data: call id src=17, dst=22
*Oct 16 08:29:54.487: offramp_data_process:
*Oct 16 08:29:54.515: docmsp_get_msp_event_buffer:
*Oct 16 08:29:56.115: docmsp_call_setup_request: callid=24
*Oct 16 08:29:56.115: docmsp_call_setup_request():  ramp data dir=ONRAMP, conf dir=DEST
*Oct 16 08:29:56.115: docmsp_caps_ind: call id=24, src=20
*Oct 16 08:29:56.115: docmsp_bridge cfid=17, srccid=24, dstcid=20
```

Related Commands

Command	Description
debug dmsp doc-to-fax	Displays debug messages for the doc Media Service Provider TIFF or text2Fax engine.

debug drip event

To display debug messages for Duplicate Ring Protocol (DRiP) events, use the **debug drip event** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug drip event

no debug drip event

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is disabled for DRiP events.

Command History

Release	Modification
11.3(4)T	This command was introduced.

Usage Guidelines

When a TrBRF interface is configured on the RSM, the DRiP protocol is activated. The DRiP protocol adds the VLAN ID specified in the router command to its database and recognizes the VLAN as a locally configured, active VLAN.

Examples

The following examples show output for the **debug drip event** command.

DRiP gets a packet from the network:

```
612B92C0: 01000C00 00000000 0C501900 0000AAAA  .........P....**
612B92D0: 0300000C 00020000 00000100 0CCCCCCC  .............LLL
612B92E0: 00000C50 19000020 AAAA0300 000C0102  ...P... **......
612B92F0: 01010114 00000002 00000002 00000C50  ...............P
612B9300: 19000001 04C00064 04                 .....@.d.
```

DRiP gets a packet from the network:

```
Recvd. pak
```

DRiP recognizes that the VLAN ID it is getting is a new one from the network:

```
6116C840:                          0100 0CCCCCCC             ...LLL
6116C850: 00102F72 CBFB0024 AAAA0300 000C0102  ../rK{.$**......
6116C860: 01FF0214 0002E254 00015003 00102F72  ......bT..P.../r
6116C870: C8000010 04C00014 044003EB 14        H....@...@.k.
DRIP : remote update - Never heard of this vlan
```

DRiP attempts to resolve any conflicts when it discovers a new VLAN. The value action = 1 means to notify the local platform of change in state.

```
DRIP : resolve remote for vlan 20 in VLAN0
DRIP : resolve remote - action = 1
```

The local platform is notified of change in state:

```
DRIP Change notification active vlan 20
```

Another new VLAN ID was received in the packet:

```
DRIP : resolve remote for vlan 1003 in Vlan0
```

No action is required:

```
DRIP : resolve remote - action = 0
```

Thirty seconds have expired, and DRiP sends its local database entries to all its trunk ports:

```
DRIP : local timer expired
DRIP : transmit on 0000.0c50.1900, length = 24
612B92C0: 01000C00 00000000 0C501900 0000AAAA  .........P....**
612B92D0: 0300000C 00020000 00000100 0CCCCCCC  .............LLL
612B92E0: 00000C50 19000020 AAAA0300 000C0102  ...P... **......
612B92F0: 01FF0114 00000003 00000002 00000C50  ...............P
612B9300: 19000001 04C00064 04                 .....@.d.
```

debug drip packet

To display debug messages for DRiP packets, use the **debug drip packet** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug drip packet

no debug drip packet

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled for DRiP packets.

Command History

Release	Modification
11.3(4)T	This command was introduced.

Usage Guidelines

Before you use this command, you can optionally use the **clear drip** command first. As a result the DRiP counters are reset to 0. If the DRiP counters begin to increment, the router is receiving packets.

Examples

Following is sample output for the **debug drip packet** command.

The following type of output is displayed when a packet is entering the router and you use the **show debug** command:

```
039E5FC0:      0100 0CCCCCCC 00E0A39B 3FFB0028    ...LLL.`#.?{.(
039E5FD0: AAAA0300 000C0102 01FF0314 0000A5F6  **............%v
039E5FE0: 00008805 00E0A39B 3C000000 04C00028  .....`#.<....@.(
039E5FF0: 04C00032 044003EB 0F                 .@.2.@.k.
039FBD20:                   01000C00 00000010          ........
```

The following type of output is displayed when a packet is sent by the router:

```
039FBD30: A6AEB450 0000AAAA 0300000C 00020000  &.4P..**........
039FBD40: 00000100 0CCCCCCC 0010A6AE B4500020  .....LLL..&.4P.
039FBD50: AAAA0300 000C0102 01FF0114 00000003  **..............
039FBD60: 00000002 0010A6AE B4500001 04C00064  ......&.4P...@.d
039FBD70: 04                                   .
```

Related Commands

Command	Description
debug drip event	Displays debug messages for DRIP events.

debug dsc clock

To display output for the time-division multiplexing (TDM) clock switching events on the dial shelf controller, use the **debug dsc clock** privileged EXEC command. To turn off output, use the **no** form of this command.

debug dsc clock

no debug dsc clock

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)AA	This command was introduced.

Usage Guidelines

The **debug dsc clock** command displays TDM clock switching events on the dial shelf controller. The information displayed includes the following:

- Clock configuration messages received from trunks via the bus
- Dial shelf controller clock configuration messages from the router shelf over the dial shelf interface link
- Clock switchover algorithm events

Examples

The following example shows that the **debug dsc clock** command has been enabled, that trunk messages are received, and that the configuration message has been received:

```
Router# debug dsc clock

Dial Shelf Controller Clock debugging is on
Router#
00:02:55: Clock Addition msg of len 12 priority 8 from slot 1 port 1 on line 0
00:02:55: Trunk 1 has reloaded
```

Related Commands

Command	Description
show dsc clock	Displays information about the dial shelf controller clock.

debug dsip

To display output for the distributed system interconnect protocol (DSIP) used between the router shelf and the dial shelf, use the **debug dsip** privileged EXEC command. To disable the output, use the **no** form of this command.

debug dsip {**all** | **api** | **boot** | **console** | **trace** | **transport**}

no debug dsip {**all** | **api** | **boot** | **console** | **trace** | **transport**}

Syntax Description

all	Displays all DSIP messages.
api	Displays DSIP client interface (API) messages.
boot	Displays DSIP booting messages that are generated when a download of the feature board image is occurring properly.
console	Displays DSIP console operation.
trace	Enables logging of header information concerning DSIP packets entering the system in a trace buffer.
transport	Debugs the DSIP transport layer, the module that interacts with the underlying physical media driver.

Command History

Release	Modification
11.3(2)AA	This command was introduced.

Usage Guidelines

The **debug dsip** command is used to display messages for DSIP between the router shelf and the dial shelf. Using this command, you can display booting messages generated when the download of an image occurs, view console operation, trace logging of MAC header information, and view DSIP transport-layer information as modules interact with the underlying physical media driver. This command can be applied to a single modem or a group of modems.

Once the **debug dsip trace** command is enabled, you can read the information captured in the trace buffer using the **show dsip tracing** command.

Examples

The following example shows the available **debug dsip** command options:

```
Router# debug dsip ?

  all        All DSIP debugging messages
  api        DSIP API debugging
  boot       DSIP booting
  console    DSIP console
  trace      DSIP tracing
  transport  DSIP transport
```

The following example indicates that the **debug dsip trace** command logs MAC headers of the various classes of DSIP packets. View the logged information using the **show dsip tracing** command.

```
Router# debug dsip trace

NIP tracing debugging is on
Router# show dsip tracing
```

```
NIP Control Packet Trace
------------------------------------------------------------
Dest:00e0.b093.2238 Src:0007.4c72.0058 Type:200B SrcShelf:1 SrcSlot:11
MsgType:0 MsgLen:82 Timestamp: 00:49:14
------------------------------------------------------------
Dest:00e0.b093.2238 Src:0007.4c72.0028 Type:200B SrcShelf:1 SrcSlot:5
MsgType:0 MsgLen:82 Timestamp: 00:49:14
------------------------------------------------------------
```

Related Commands

Command	Description
debug modem dsip	Displays output for modem control messages that are received or sent to the router.

debug dspu activation

To display information on downstream physical unit (DSPU) activation, use the **debug dspu activation** privileged EXEC command. The **no** form of this command disables debugging output.

debug dspu activation [*name*]

no debug dspu activation [*name*]

Syntax Description

name	(Optional) The host or physical unit (PU) name designation.

Usage Guidelines

The **debug dspu activation** command displays all DSPU activation traffic. To restrict the output to a specific host or PU, include the host or PU *name* argument. You cannot turn off debugging output for an individual PU if that PU has not been named in the **debug dspu activation** command.

Examples

The following is sample output from the **debug dspu activation** command. Not all intermediate numbers are shown for the "activated" and "deactivated" logical unit (LU) address ranges.

```
Router# debug dspu activation

DSPU: LS HOST3745 connected
DSPU: PU HOST3745 activated
DSPU: LU HOST3745-2 activated
DSPU: LU HOST3745-3 activated
.
.
.
DSPU: LU HOST3745-253 activated
DSPU: LU HOST3745-254 activated

DSPU: LU HOST3745-2 deactivated
DSPU: LU HOST3745-3 deactivated
.
.
.
DSPU: LU HOST3745-253 deactivated
DSPU: LU HOST3745-254 deactivated
DSPU: LS HOST3745 disconnected
DSPU: PU HOST3745 deactivated
```

Table 42 describes the significant fields shown in the display.

Table 42 debug dspu activation Field Descriptions

Field	Description
DSPU	Downstream PU debug message.
LS	Link station (LS) event triggered the message.
PU	PU event triggered the message.
LU	LU event triggered the message.
HOST3745	Host name or PU name.

Table 42 debug dspu activation Field Descriptions (continued)

Field	Description
HOST3745-253	Host name or PU name and the LU address, separated by a dash.
connected activated disconnected deactivated	Event that occurred to trigger the message.

Related Commands

Command	Description
debug dspu packet	Displays information on a DSPU packet.
debug dspu state	Displays information on DSPU FSM state changes.
debug dspu trace	Displays information on DSPU trace activity.

debug dspu packet

To display information on a downstream physical unit (DSPU) packet, use the **debug dspu packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug dspu packet [*name*]

no debug dspu packet [*name*]

Syntax Description

name	(Optional) The host or PU name designation.

Usage Guidelines

The **debug dspu packet** command displays all DSPU packet data flowing through the router. To restrict the output to a specific host or PU, include the host or PU *name* argument. You cannot turn off debugging output for an individual PU if that PU has not been named in the **debug dspu packet** command.

Examples

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

```
Router# debug dspu packet

DSPU: Rx: PU HOST3745 data length 12 data:
    2D0003002BE16B80 000D0201
DSPU: Tx: PU HOST3745 data length 25 data:
    2D0000032BE1EB80 000D020100850000 000C060000010000 00
DSPU: Rx: PU HOST3745 data length 12 data:
    2D0004002BE26B80 000D0201
DSPU: Tx: PU HOST3745 data length 25 data:
    2D0000042BE2EB80 000D020100850000 000C060000010000 00
```

Table 43 describes the significant fields shown in the display.

Table 43 debug dspu packet Field Descriptions

Field	Description
DSPU: Rx:	Received frame (packet) from the remote PU to the router PU.
DSPU: Tx:	Transmitted frame (packet) from the router PU to the remote PU.
PU HOST3745	Host name or PU associated with the transmit or receive.
data length 12 data:	Number of bytes of data, followed by up to 128 bytes of displayed data.

Related Commands

Command	Description
debug drip event	Displays debug messages for DRiP packets.
debug dspu state	Displays information on DSPU FSM state changes.
debug dspu trace	Displays information on DSPU trace activity.

debug dspu state

To display information on downstream physical unit (DSPU) finite state machine (FSM) state changes, use the **debug dspu state** privileged EXEC command. The **no** form of this command disables debugging output.

debug dspu state [*name*]

no debug dspu state [*name*]

Syntax Description

name	(Optional) The host or PU name designation.

Usage Guidelines

Use the **debug dspu state** command to display only the FSM state changes. To see all FSM activity, use the debug **dspu trace** command. You cannot turn off debugging output for an individual PU if that PU has not been named in the **debug dspu state** command.

Examples

The following is sample output from the **debug dspu state** command. Not all intermediate numbers are shown for the "activated" and "deactivated" logical unit (LU) address ranges.

```
Router# debug dspu state

DSPU: LS HOST3745: input=StartLs, Reset -> PendConOut
DSPU: LS HOST3745: input=ReqOpn.Cnf, PendConOut -> Xid
DSPU: LS HOST3745: input=Connect.Ind, Xid -> ConnIn
DSPU: LS HOST3745: input=Connected.Ind, ConnIn -> Connected
DSPU: PU HOST3745: input=Actpu, Reset -> Active
DSPU: LU HOST3745-2: input=uActlu, Reset -> upLuActive
DSPU: LU HOST3745-3: input=uActlu, Reset -> upLuActive
.
.
.
DSPU: LU HOST3745-253: input=uActlu, Reset -> upLuActive
DSPU: LU HOST3745-254: input=uActlu, Reset -> upLuActive

DSPU: LS HOST3745: input=PuStopped, Connected -> PendDisc
DSPU: LS HOST3745: input=Disc.Cnf, PendDisc -> PendClose
DSPU: LS HOST3745: input=Close.Cnf, PendClose -> Reset
DSPU: PU HOST3745: input=T2ResetPu, Active -> Reset
DSPU: LU HOST3745-2: input=uStopLu, upLuActive -> Reset
DSPU: LU HOST3745-3: input=uStopLu, upLuActive -> Reset
.
.
.
DSPU: LU HOST3745-253: input=uStopLu, upLuActive -> Reset
DSPU: LU HOST3745-254: input=uStopLu, upLuActive -> Reset
```

Table 44 describes the significant fields shown in the display.

Table 44 debug dspu state Coomand Field Descriptions

Field	Description
DSPU	Downstream PU debug message.
LS	Link station (LS) event triggered the message.
PU	PU event triggered the message.
LU	LU event triggered the message.
HOST3745-253	Host name or PU name and LU address.
input=input,	Input received by the FSM.
previous-state, -> current-state	Previous state and current new state as seen by the FSM.

Related Commands

Command	Description
debug drip event	Displays debug messages for DRiP packets.
debug drip packet	Displays information on DSPU packet.
debug dspu trace	Displays information on DSPU trace activity.

debug dspu trace

To display information on downstream physical unit (DSPU) trace activity, which includes all finite state machine (FSM) activity, use the **debug dspu trace** privileged EXEC command. The **no** form of this command disables debugging output.

debug dspu trace [*name*]

no debug dspu trace [*name*]

Syntax Description

name	(Optional) The host or PU name designation.

Usage Guidelines

Use the **debug dspu trace** command to display all FSM state changes. To see FSM state changes only, use the **debug dspu state** command. You cannot turn off debugging output for an individual PU if that PU has not been named in the **debug dspu trace** command.

Examples

The following is sample output from the **debug dspu trace** command:

```
Router# debug dspu trace

DSPU: LS HOST3745 input = 0 ->(1,a1)
DSPU: LS HOST3745 input = 5 ->(5,a6)
DSPU: LS HOST3745 input = 7 ->(5,a9)
DSPU: LS HOST3745 input = 9 ->(5,a28)
DSPU: LU HOST3745-2 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-3 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-252 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-253 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
DSPU: LS HOST3745 input = 18 ->(8,a17)
DSPU: LU HOST3745-254 in:0 s:0->(2,a1)
DSPU: LS HOST3745 input = 19 ->(8,a20)
```

Table 45 describes significant fields in the output.

Table 45 debug dspu trace Field Descriptions

Field	Description
7:23:57	Time stamp.
DSPU	Downstream PU debug message.
LS	Link station (LS) event triggered the message.
PU	A PU event triggered the message.
LU	LU event triggered the message.
HOST3745-253	Host name or PU name and LU address.

Table 45 debug dspu trace Field Descriptions (continued)

Field	Description
in:input s:state ->(new-state, action)	String describing the following: • input—LU FSM input • state—Current FSM state • new-state—New FSM state • action—FSM action
input=input -> (new-state, action)	String describing the following: • input—PU or LS FSM input • new-state—New PU or LS FSM state • action—PU or LS FSM action

Related Commands

Command	Description
debug drip event	Displays debug messages for DRiP packets.
debug drip packet	Displays information on DSPU packet.
debug dspu state	Displays information on DSPU FSM state changes.

debug dss ipx event

To display debug messages for route change events that affect IPX Multilayer Switching (MLS), use the **debug dss ipx event** privileged EXEC command. To disable debugging output, use the **no** form of the command.

debug dss ipx event

no debug dss ipx event

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled.

Command History

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

Examples

The following displays sample output from the **debug dss ipx event** command:

```
Router# debug dss ipx event

    DSS IPX events debugging is on
Router# configure terminal
    Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# interface vlan 22
Router(config-if)# ipx access-group 800 out
    05:51:36:DSS-feature:dss_ipxcache_version():idb:NULL, reason:42,
    prefix:0, mask:FFFFFFFF
    05:51:36:DSS-feature:dss_ipx_access_group():idb:Vlan22
    05:51:36:DSS-feature:dss_ipx_access_list()
    05:51:36:DSS-base 05:51:33.834 dss_ipx_invalidate_interface Vl22
    05:51:36:DSS-base 05:51:33.834 dss_set_ipx_flowmask_reg 2
    05:51:36:%IPX mls flowmask transition from 1 to 2 due to new status of
    simple IPX access list on interfaces
```

Related Commands

Command	Description
debug mls rp	Displays various MLS debugging elements.

debug eigrp fsm

To display debugging information about Enhanced Interior Gateway Routing Protocol (EIGRP) feasible successor metrics (FSM), use the **debug eigrp fsm** privileged EXEC command. The **no** form of this command disables debugging output.

debug eigrp fsm

no debug eigrp fsm

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command helps you observe EIGRP feasible successor activity and to determine whether route updates are being installed and deleted by the routing process.

Examples

The following is sample output from the **debug eigrp fsm** command:

```
Router# debug eigrp fsm

DUAL: dual_rcvupdate(): 172.25.166.0 255.255.255.0 via 0.0.0.0 metric 750080/0
DUAL: Find FS for dest 172.25.166.0 255.255.255.0. FD is 4294967295, RD is 42949
67295 found
DUAL: RT installed 172.25.166.0 255.255.255.0 via 0.0.0.0
DUAL: dual_rcvupdate(): 192.168.4.0 255.255.255.0 via 0.0.0.0 metric 4294967295/
4294967295
DUAL: Find FS for dest 192.168.4.0 255.255.255.0. FD is 2249216, RD is 2249216
DUAL:   0.0.0.0 metric 4294967295/4294967295not found Dmin is 4294967295
DUAL: Dest 192.168.4.0 255.255.255.0 not entering active state.
DUAL: Removing dest 192.168.4.0 255.255.255.0, nexthop 0.0.0.0
DUAL: No routes. Flushing dest 192.168.4.0 255.255.255.0
```

In the first line, DUAL stands for diffusing update algorithm. It is the basic mechanism within EIGRP that makes the routing decisions. The next three fields are the Internet address and mask of the destination network and the address through which the update was received. The metric field shows the metric stored in the routing table and the metric advertised by the neighbor sending the information. If shown, the term “Metric... inaccessible” usually means that the neighbor router no longer has a route to the destination, or the destination is in a hold-down state.

In the following output, EIGRP is attempting to find a feasible successor for the destination. Feasible successors are part of the DUAL loop avoidance methods. The FD field contains more loop avoidance state information. The RD field is the reported distance, which is the metric used in update, query, or reply packets.

The indented line with the “not found” message means a feasible successor (FS) was not found for 192.168.4.0 and EIGRP must start a diffusing computation. This means it begins to actively probe (sends query packets about destination 192.168.4.0) the network looking for alternate paths to 192.164.4.0.

```
DUAL: Find FS for dest 192.168.4.0 255.255.255.0. FD is 2249216, RD is 2249216
DUAL:   0.0.0.0 metric 4294967295/4294967295not found Dmin is 4294967295
```

The following output indicates the route DUAL successfully installed into the routing table:

```
DUAL: RT installed 172.25.166.0 255.255.255.0 via 0.0.0.0
```

The following output shows that no routes to the destination were discovered and that the route information is being removed from the topology table:

```
DUAL: Dest 192.168.4.0 255.255.255.0 not entering active state.
DUAL: Removing dest 192.168.4.0 255.255.255.0, nexthop 0.0.0.0
DUAL: No routes. Flushing dest 192.168.4.0 255.255.255.0
```

debug eigrp neighbor

To display neighbors discovered by the Enhanced Interior Gateway Routing Protocol (EIGRP), use the **debug eigrp neighbor** command in privileged EXEC mode. To disable **debug eigrp neighbor**, use the **no** form of this command.

debug eigrp neighbor [**siatimer**] [**static**]

no debug eigrp neighbor [**siatimer**] [**static**]

Syntax Description

siatimer	(Optional) Stuck-in-active (SIA) timer messages.
static	(Optional) Static routes.

Defaults

Debugging for EIGRP neighbors is not enabled.

Command Modes

Privileged EXEC

Command History

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

Examples

The following is sample output from the **debug eigrp neighbor** command.

```
Router# debug eigrp neighbor static

EIGRP Static Neighbors debugging is on

Router#configure terminal

Router(config)#router eigrp 100

Router(config-router)#neighbor 10.1.1.1 e3/1

Router(config-router)#
22:40:07:EIGRP:Multicast Hello is disabled on Ethernet3/1!
22:40:07:EIGRP:Add new static nbr 10.1.1.1 to AS 100 Ethernet3/1

Router(config-router)#no neighbor 10.1.1.1 e3/1

Router(config-router)#
22:41:23:EIGRP:Static nbr 10.1.1.1 not in AS 100 Ethernet3/1 dynamic list
22:41:23:EIGRP:Delete static nbr 10.1.1.1 from AS 100 Ethernet3/1
22:41:23:EIGRP:Multicast Hello is enabled on Ethernet3/1!
```

Related Commands

Command	Description
show ip eigrp neighbors	Displays EIGRP neighbors.
neighbor	Defines a neighboring router with which to exchange routing information.

debug eigrp packet

To display general debugging information, use the **debug eigrp packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug eigrp packet

no debug eigrp packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

If a communication session is closing when it should not be, an end-to-end connection problem can be the cause. The **debug eigrp packet** command is useful for analyzing the messages traveling between the local and remote hosts.

Examples

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

```
Router# debug eigrp packet

EIGRP: Sending HELLO on Ethernet0/1
       AS 109, Flags 0x0, Seq 0, Ack 0
EIGRP: Sending HELLO on Ethernet0/1
       AS 109, Flags 0x0, Seq 0, Ack 0
EIGRP: Sending HELLO on Ethernet0/1
       AS 109, Flags 0x0, Seq 0, Ack 0
EIGRP: Received UPDATE on Ethernet0/1 from 192.195.78.24,
       AS 109, Flags 0x1, Seq 1, Ack 0
EIGRP: Sending HELLO/ACK on Ethernet0/1 to 192.195.78.24,
       AS 109, Flags 0x0, Seq 0, Ack 1
EIGRP: Sending HELLO/ACK on Ethernet0/1 to 192.195.78.24,
       AS 109, Flags 0x0, Seq 0, Ack 1
EIGRP: Received UPDATE on Ethernet0/1 from 192.195.78.24,
       AS 109, Flags 0x0, Seq 2, Ack 0
```

The output shows transmission and receipt of Enhanced Interior Gateway Routing Protocol (EIGRP) packets. These packet types may be hello, update, request, query, or reply packets. The sequence and acknowledgment numbers used by the EIGRP reliable transport algorithm are shown in the output. Where applicable, the network-layer address of the neighboring router is also included.

Table 46 describes the significant fields shown in the display.

Table 46 debug eigrp packet Field Descriptions

Field	Description
EIGRP:	EIGRP packet information.
AS n	Autonomous system number.

Table 46 debug eigrp packet Field Descriptions (continued)

Field	Description
Flags *nxn*	A flag of 1 means the sending router is indicating to the receiving router that this is the first packet it has sent to the receiver. A flag of 2 is a multicast that should be conditionally received by routers that have the conditionally receive (CR) bit set. This bit gets set when the sender of the multicast has previously sent a sequence packet explicitly telling it to set the CR bit.
HELLO	Hello packets are the neighbor discovery packets. They are used to determine whether neighbors are still alive. As long as neighbors receive the hello packets the router is sending, the neighbors validate the router and any routing information sent. If neighbors lose the hello packets, the receiving neighbors invalidate any routing information previously sent. Neighbors also send hello packets.

debug eigrp transmit

To display transmittal messages sent by the Enhanced Interior Gateway Routing Protocol (EIGRP), use the **debug eigrp transmit** command in privileged EXEC mode. To disable **debug eigrp transmit**, use the **no** form of this command.

debug eigrp transmit [**ack**] [**build**] [**detail**] [**link**] [**packetize**] [**peerdown**] [**startup**] [**strange**]

no debug eigrp transmit [**ack**] [**build**] [**detail**] [**link**] [**packetize**] [**peerdown**] [**sia**] [**startup**] [**strange**]

Syntax Description

ack	(Optional) Information for acknowledgment (ACK) messages sent by the system.
build	(Optional) Build information messages (messages that indicate that a topology table was either successfully built or could not be built).
detail	(Optional) Additional detail for debug output.
link	(Optional) Information regarding topology table linked-list management.
packetize	(Optional) Information regarding topology table linked-list management.
peerdown	(Optional) Information regarding the impact on packet generation when a peer is down.
sia	(Optional) Stuck-in-active (SIA) messages.
startup	(Optional) Information regarding peer startup and initialization packets that have been transmitted.
strange	(Optional) Unusual events relating to packet processing.

Defaults

Debugging for EIGRP transmittal messages is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.

Examples

The following is sample output from the **debug eigrp transmit** command.

```
Router# debug eigrp transmit

EIGRP Transmission Events debugging is on
    (ACK, PACKETIZE, STARTUP, PEERDOWN, LINK, BUILD, STRANGE, SIA, DETAIL)

Router#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.
Router#(config)#router eigrp 100
Router#(config-router)#network 10.4.9.0 0.0.0.255
Router#(config-router)#
5d22h: DNDB UPDATE 10.0.0.0/8, serno 0 to 1, refcount 0
Router#(config-router)#
```

debug errors

To display errors, use the **debug errors** privileged EXEC command. The **no** form of this command disables debugging output.

debug errors

no debug errors

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug errors

(2/0): Encapsulation error, link=7, host=836CA86D.
(4/0): VCD#7 failed to echo OAM. 4 tries
```

The first line of output indicates that a packet was routed to the interface, but no static map was set up to route that packet to the proper virtual circuit.

The second line of output shows that an OAM F5 (virtual circuit) cell error occurred.

debug events

To display events, use the **debug events** privileged EXEC command. The **no** form of this command disables debugging output.

debug events

no debug events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command displays events that occur on the interface processor and is useful for diagnosing problems in an network. It provides an overall picture of the stability of the network. In a stable network, the **debug events** command does not return any information. If the command generates numerous messages, the messages can indicate the possible source of problems.

When configuring or making changes to a router or interface for, enable the **debug events** command. Doing so alerts you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

Examples

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

```
Router# debug events

RESET(4/0): PLIM type is 1, Rate is 100Mbps
aip_disable(4/0): state=1
config(4/0)
aip_love_note(4/0): asr=0x201
aip_enable(4/0)
aip_love_note(4/0): asr=0x4000
aip_enable(4/0): restarting VCs: 7
aip_setup_vc(4/0): vc:1 vpi:1 vci:1
aip_love_note(4/0): asr=0x200
aip_setup_vc(4/0): vc:2 vpi:2 vci:2
aip_love_note(4/0): asr=0x200
aip_setup_vc(4/0): vc:3 vpi:3 vci:3
aip_love_note(4/0): asr=0x200
aip_setup_vc(4/0): vc:4 vpi:4 vci:4
aip_love_note(4/0): asr=0x200
aip_setup_vc(4/0): vc:6 vpi:6 vci:6
aip_love_note(4/0): asr=0x200
aip_setup_vc(4/0): vc:7 vpi:7 vci:7
aip_love_note(4/0): asr=0x200
aip_setup_vc(4/0): vc:11 vpi:11 vci:11
aip_love_note(4/0): asr=0x200
```

Table 47 describes the significant fields in the display.

Table 47 debug events Field Descriptions

Field	Description
PLIM type	Indicates the interface rate in Mbps. Possible values are: • 1 = TAXI(4B5B) 100 Mbps • 2 = SONET 155 Mbps • 3 = E3 34 Mbps
state	Indicates current state of the AIP. Possible values are: • 1 = An ENABLE will be issued soon. • 0 = The AIP will remain shut down.
asr	Defines a bitmask, which indicates actions or completions to commands. Valid bitmask values are: • 0x0800 = AIP crashed, reload may be required. • 0x0400 = AIP detected a carrier state change. • 0x0n00 = Command completion status. Command completion status codes are: – n = 8 Invalid PLIM detected – n = 4 Command failed – n = 2 Command completed successfully – n = 1 CONFIG request failed – n = 0 Invalid value

The following line indicates that the AIP was reset. The PLIM detected was 1, so the maximum rate is set to 100 Mbps.

```
RESET(4/0): PLIM type is 1, Rate is 100Mbps
```

The following line indicates that the AIP was given a **shutdown** command, but the current configuration indicates that the AIP should be up:

```
aip_disable(4/0): state=1
```

The following line indicates that a configuration command has been completed by the AIP:

```
aip_love_note(4/0): asr=0x201
```

The following line indicates that the AIP was given a **no shutdown** command to take it out of the shutdown state:

```
aip_enable(4/0)
```

The following line indicates that the AIP detected a carrier state change. It does not indicate that the carrier is down or up, only that it has changed.

```
aip_love_note(4/0): asr=0x4000
```

The following line of output indicates that the AIP enable function is restarting all PVCs automatically:

```
aip_enable(4/0): restarting VCs: 7
```

The following lines of output indicate that PVC 1 was set up and a successful completion code was returned:

```
aip_setup_vc(4/0): vc:1 vpi:1 vci:1
aip_love_note(4/0): asr=0x200
```

debug fddi smt-packets

To display information about Station Management (SMT) frames received by the router, use the **debug fddi smt-packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug fddi smt-packets

no debug fddi smt-packets

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug fddi smt-packets** command. In this example, an SMT frame has been output by FDDI 1/0. The SMT frame is a next station addressing (NSA) neighbor information frame (NIF) request frame with the parameters as shown.

```
Router# debug fddi smt-packets

SMT O: Fddi1/0, FC=NSA, DA=ffff.ffff.ffff, SA=00c0.eeee.be04,
 class=NIF, type=Request, vers=1, station_id=00c0.eeee.be04, len=40
 - code 1, len 8 -- 000000016850043F
 - code 2, len 4 -- 00010200
 - code 3, len 4 -- 00003100
 - code 200B, len 8 -- 0000000100000000
```

Table 48 describes the significant fields shown in the display.

Table 48 debug fddi smt-packets Field Descriptions

Field	Description
SMT O	SMT frame was sent from FDDI interface 1/0. Also, SMT I indicates that an SMT frame was received on the FDDI interface 1/0.
Fddi1/0	Interface associated with the frame.
FC	Frame control byte in the MAC header.
DA, SA	Destination and source addresses in FDDI form.
class	Frame class. Values can be echo frame (ECF), neighbor information frame (NIF), parameter management frame (PMF), request denied frame (RDF), status information frame (SIF), and status report frame (SRF).
type	Frame type. Values can be Request, Response, and Announce.
vers	Version identification. Values can be 1 or 2.
station_id	Station identification.
len	Packet size.
code 1, len 8 -- 000000016850043F	Parameter type X’0001—upstream neighbor address (UNA), parameter length in bytes, and parameter value. SMT parameters are described in the SMT specification ANSI X3T9.

debug fmsp receive

To display debug messages for FMSP receive, use the **debug fmsp receive** EXEC command. To disable the debug messages, use the **no** form of this command.

debug fmsp receive [**t30** | **t38**]

no debug fmsp receive [**t30** | **t38**]

Syntax Description

t30	(Optional) Specifies the T.30 fax protocol.
t38	(Optional) Specifies the T.38 fax protocol.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug fmsp receive** command.

```
Router# debug fmsp receive

*Oct 16 08:31:33.243: faxmsp_call_setup_request: call id=28
*Oct 16 08:31:33.243: faxmsp_call_setup_request: ramp data dir=ONRAMP, conf dir=DEST
*Oct 16 08:31:33.243: faxmsp_bridge(): cfid=19, srccid=28, dstcid=27

*Oct 16 08:31:33.243: faxmsp_bridge():  ramp data dir=ONRAMP, conf dir=DEST
*Oct 16 08:31:33.243: faxmsp_bridge(): Explicit caps ind. done; will wait for registry
caps ind
*Oct 16 08:31:33.243: faxmsp_caps_ind: call id=28, src=27
*Oct 16 08:31:33.243: faxmsp_caps_ack: call id src=27
*Oct 16 08:31:33.279: faxmsp_call_setup_request: call id=29
*Oct 16 08:31:33.279: faxmsp_call_setup_request: ramp data dir=OFFRAMP, conf dir=SRC
*Oct 16 08:31:33.283: faxmsp_bridge(): cfid=20, srccid=29, dstcid=26

*Oct 16 08:31:33.283: faxmsp_bridge():  ramp data dir=OFFRAMP, conf dir=SRC
*Oct 16 08:31:33.283: faxmsp_bridge(): Explicit caps ind. done; will wait for registry
caps ind
*Oct 16 08:31:33.283: faxmsp_caps_ind: call id=29, src=26
*Oct 16 08:31:33.283: faxmsp_caps_ack: call id src=26
*Oct 16 08:31:33.635: faxmsp_codec_download_done: call id=29
*Oct 16 08:31:33.635: faxmsp_codec_download_done: call id=28
*Oct 16 08:31:33.643: faxmsp_xmit: callid src=26, dst=29
*Oct 16 08:31:33.643: faxmsp_xmit: callid src=27, dst=28
*Oct 16 08:31:33.643: faxmsp_process_rcv_data: call id src=26, dst=29
```

Related Commands

Command	Description
debug fmsp send	Displays debug messages for FMSP send.

debug fmsp send

To display debug messages for FMSP send, use the **debug fmsp send** EXEC command. To disable the debug messages, use the **no** form of this command.

debug fmsp send [**t30** | **t38**]

no debug fmsp send [**t30** | **t38**]

Syntax Description

t30	(Optional) Specifies the T.30 fax protocol.
t38	(Optional) Specifies the T.38 fax protocol.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug fmsp send** command.

```
Router# debug fmsp send

Jan  1 05:02:56.782: faxmsp_call_setup_request: call id=21
Jan  1 05:02:56.782: faxmsp_call_setup_request: ramp data dir=OFFRAMP, conf dir=SRC
Jan  1 05:02:56.782: faxmsp_bridge(): cfid=7, srccid=21, dstcid=20

Jan  1 05:02:56.782: faxmsp_bridge():  ramp data dir=OFFRAMP, conf dir=SRC
Jan  1 05:02:56.782: faxmsp_bridge(): Explicit caps ind. done; will wait for registry caps
ind
Jan  1 05:02:56.782: faxmsp_caps_ind: call id=21, src=20
Jan  1 05:02:56.782: faxmsp_caps_ack: call id src=20
Jan  1 05:02:57.174: faxmsp_codec_download_done: call id=21
Jan  1 05:02:57.174: faxMsp_tx_buffer callID=21
Jan  1 05:02:57.178: faxMsp_tx_buffer callID=21
Jan  1 05:02:57.178: faxMsp_tx_buffer callID=21
Jan  1 05:02:57.178: faxMsp_tx_buffer callID=21
Jan  1 05:02:57.182: faxmsp_xmit: callid src=20, dst=21
Jan  1 05:02:57.182: faxmsp_process_rcv_data: call id src=20, dst=21
Jan  1 05:03:01.814: faxmsp_xmit: callid src=20, dst=21
Jan  1 05:03:01.814: faxmsp_process_rcv_data: call id src=20, dst=21
Jan  1 05:03:01.814: faxMsp_tx_buffer callID=21
Jan  1 05:03:02.802: faxmsp_xmit: callid src=20, dst=21
Jan  1 05:03:02.802: faxmsp_process_rcv_data: call id src=20, dst=21
Jan  1 05:03:02.822: faxmsp_xmit: callid src=20, dst=21
Jan  1 05:03:02.822: faxmsp_process_rcv_data: call id src=20, dst=21
Jan  1 05:03:02.854: faxmsp_xmit: callid src=20, dst=21
Jan  1 05:03:02.854: faxmsp_process_rcv_data: call id src=20, dst=21
```

Related Commands

Command	Description
debug fmsp receive	Displays debug messages for FMSP receive.

debug foip off-ramp

To display debug messages for off-ramp faxmail, use the **debug foip off-ramp** EXEC command. To disable the debug messages, use the **no** form of this command.

debug foip off-ramp

no debug foip off-ramp

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug foip off-ramp** command.

```
Router# debug foip off-ramp

Jan  1 02:31:17.539: lapp off: CC_EV_CALL_HANDOFF, cid(0xB)
Jan  1 02:31:17.539: loffHandoff: called number=5271714, callid=0xB
Jan  1 02:31:17.543: loffSetupPeer: cid1(0xB)
Jan  1 02:31:17.543:  destPat(5271714),matched(1),pref(5),tag(20),encap(1)
Jan  1 02:31:22.867: lapp off: CC_EV_CALL_CONNECTED, cid(0xC)
Jan  1 02:31:22.867:  st=CALL_SETTING cid(0xB,0x0,0x0,0xC),cfid(0x0,0x0,0x0)
Jan  1 02:31:22.867: loffConnected
Jan  1 02:31:22.867: loffFlushPeerTagQueue cid(11) peer list: (empty)
Jan  1 02:31:22.867: lapp off: CC_EV_CONF_CREATE_DONE, cid(0xC), cid2(0xD), cfid(0x1)
Jan  1 02:31:22.867:  st=CONFERENCING3 cid(0xB,0x0,0xD,0xC),cfid(0x0,0x0,0x1)
Jan  1 02:31:22.867: loffConfDone3
Jan  1 02:31:30.931: lapp off: CC_EV_FROM_FMSP_ON_CALL_DETAIL, cid(0xD)
Jan  1 02:31:30.931:  st=WAIT_SESS_INFO cid(0xB,0x0,0xD,0xC),cfid(0x0,0x0,0x1)
Jan  1 02:31:30.931: loffSessionInfo
Jan  1 02:31:30.931:  encd=2, resl=2, spd=26, min_scan_len=0, csid=          4085271714
Jan  1 02:31:30.931: lapp off: CC_EV_CONF_CREATE_DONE, cid(0xD), cid2(0xE), cfid(0x2)
Jan  1 02:31:30.931:  st=CONFERENCING2 cid(0xB,0xE,0xD,0xC),cfid(0x0,0x2,0x1)
Jan  1 02:31:30.931: loffConfDone2
```

Related Commands

Command	Description
debug foip on-ramp	Displays debug messages for on-ramp faxmail.

debug foip on-ramp

To display debug messages for on-ramp faxmail, use the **debug foip on-ramp** EXEC command. To disable the debug messages, use the **no** form of this command.

debug foip on-ramp

no debug foip on-ramp

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug foip on-ramp** command.

```
Router# debug foip on-ramp

*Oct 16 08:07:01.947: lapp_on_application: Incoming Event: (15 = CC_EV_CALL_HANDOFF),
CID(11), DISP(0)
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication enabled = FALSE
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication ID = 0
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication ID source = IVR or unknown
*Oct 16 08:07:01.947: lapp_on_call_handoff: Authentication status = SUCCESS
*Oct 16 08:07:01.947: lapp_on_call_handoff: Accounting enabled  = FALSE
*Oct 16 08:07:01.947: lapp_on_call_handoff: Accounting method list  = fax
*Oct 16 08:07:01.947: lapp_on_conference_vtsp_fmsp: Begin conferencing VTSP and FMSP...
*Oct 16 08:07:01.951: lapp_on_change_state: old state(0) new state(1)
*Oct 16 08:07:01.951: lapp_on_application: Incoming Event: (29 = CC_EV_CONF_CREATE_DONE),
CID(11), DISP(0)
*Oct 16 08:07:01.951: lapp_on_application: Current call state = 1
*Oct 16 08:07:01.951: lapp_on_conference_created: The VTSP and the FMSP are conferenced
*Oct 16 08:07:01.951: lapp_on_conference_created: Wait for FMSP call detail event
```

Related Commands

Command	Description
debug foip off-ramp	Displays debug messages for off-ramp faxmail.

debug frame-relay

To display debugging information about the packets received on a Frame Relay interface, use the **debug frame-relay** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay

no debug frame-relay

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command helps you analyze the packets that have been received. However, because the **debug frame-relay** command generates a substantial amount of output, only use it when traffic on the Frame Relay network is fewer than 25 packets per second.

To analyze the packets that have been *sent* on a Frame Relay interface, use the **debug frame-relay packet** command.

Examples

The following is sample output from the **debug frame-relay** command:

```
Router# debug frame-relay

Serial0(i): dlci 500(0x7C41), pkt type 0x809B, datagramsize 24
Serial1(i): dlci 1023(0xFCF1), pkt type 0x309, datagramsize 13
Serial0(i): dlci 500(0x7C41), pkt type 0x809B, datagramsize 24
Serial1(i): dlci 1023(0xFCF1), pkt type 0x309, datagramsize 13
Serial0(i): dlci 500(0x7C41), pkt type 0x809B, datagramsize 24
```

Table 49 describes the significant fields shown in the display.

Table 49 debug frame-relay Field Descriptions

Field	Description
Serial0(i):	Indicates that serial interface 0 has received this Frame Relay datagram as input.
dlci 500(0x7C41)	Indicates the value of the data-link connection identifier (DLCI) for this packet in decimal (and q922). In this case, 500 has been configured as the multicast DLCI.

Table 49 debug frame-relay Field Descriptions (continued)

Field	Description
pkt type 0x809B	Indicates the packet type code. Possible supported signalling message codes are as follows: • 0x308—Signalling message; valid only with a DLCI of 0 • 0x309—LMI message; valid only with a DLCI of 1023Possible supported Ethernet type codes are: • 0x0201—IP on a 3 MB net • 0x0201—Xerox ARP on 10 MB networks • 0xCC—RFC 1294 (only for IP) • 0x0600—XNS • 0x0800—IP on a 10 MB network • 0x0806—IP ARP • 0x0808—Frame Relay ARP • 0x0BAD—VINES IP • 0x0BAE—VINES loopback protocol • 0x0BAF—VINES Echo Possible HDLC type codes are as follows: • 0x6001—DEC MOP booting protocol • 0x6002—DEC MOP console protocol • 0x6003—DECnet Phase IV on Ethernet • 0x6004—DEC LAT on Ethernet • 0x8005—HP Probe • 0x8035—RARP • 0x8038—DEC spanning tree • 0x809b—Apple EtherTalk • 0x80f3—AppleTalk ARP • 0x8019—Apollo domain • 0x80C4—VINES IP • 0x80C5—VINES ECHO • 0x8137—IPX • 0x9000—Ethernet loopback packet IP • 0x1A58—IPX, standard form • 0xFEFE—CLNS • 0xEFEF—ES-IS • 0x1998—Uncompressed TCP • 0x1999—Compressed TCP • 0x6558—Serial line bridging

Table 49 debug frame-relay Field Descriptions (continued)

Field	Description
datagramsize 24	Indicates size of this datagram (in bytes).

debug frame-relay callcontrol

To display Frame Relay Layer 3 (network layer) call control information, use the **debug frame-relay callcontrol** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay callcontrol

no debug frame-relay callcontrol

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug frame-relay callcontrol** command is used specifically for observing FRF.4/Q.933 signalling messages and related state changes. The FRF.4/Q.933 specification describes a state machine for call control. The signalling code implements the state machine. The debug statements display the actual event and state combinations.

The Frame Relay switched virtual circuit (SVC) signalling subsystem is an independent software module. When used with the **debug frame-relay networklayerinterface** command, the **debug frame-relay callcontrol** command provides a better understanding of the call setup and teardown sequence. The **debug frame-relay networklayerinterface** command provides the details of the interactions between the signalling subsystem on the router and the Frame Relay subsystem.

Examples

State changes can be observed during a call setup on the calling party side. The **debug frame-relay networklayerinterface** command shows the following state changes or transitions:

```
STATE_NULL -> STATE_CALL_INITIATED -> STATE_CALL_PROCEEDING->STATE_ACTIVE
```

The following messages are samples of output generated during a call setup on the calling side:

```
6d20h: U0_SetupRequest: Serial0
6d20h: L3SDL: Ref: 1, Init: STATE_NULL, Rcvd: SETUP_REQUEST, Next: STATE_CALL_INITIATED
6d20h: U1_CallProceeding: Serial0
6d20h: L3SDL: Ref: 1, Init: STATE_CALL_INITIATED, Rcvd: MSG_CALL_PROCEEDING, Next:
     STATE_CALL_PROCEEDING
6d20h: U3_Connect: Serial0
6d20h: L3SDL: Ref: 1, Init: STATE_CALL_PROCEEDING, Rcvd: MSG_CONNECT, Next: STATE_ACTIVE
6d20h:
```

The following messages are samples of output generated during a call setup on the called party side. Note the state transitions as the call goes to the active state:

```
STATE_NULL -> STATE_CALL_PRESENT-> STATE_INCOMING_CALL_PROCEEDING->STATE_ACTIVE

1w4d: U0_Setup: Serial2/3
1w4d: L3SDL: Ref: 32769, Init: STATE_NULL, Rcvd: MSG_SETUP, Next: STATE_CALL_PRESENT 1w4d:
L3SDL: Ref: 32769, Init: STATE_CALL_PRESENT, Rcvd: MSG_SETUP, Next:
     STATE_INCOMING_CALL_PROC 1w4d: L3SDL: Ref: 32769, Init: STATE_INCOMING_CALL_PROC,
     Rcvd: MSG_SETUP, Next: STATE_ACTIVE
```

Table 50 explains the possible call states.

Table 50 Frame Relay Switched Virtual Circuit Call States

Call State	Description
Null	No call exists.
Call Initiated	User has requested the network to establish a call.
Outgoing Call Proceeding	User has received confirmation from the network that the network has received all call information necessary to establish the call.
Call Present	User has received a request to establish a call but has not yet responded.
Incoming Call Proceeding	User has sent acknowledgment that all call information necessary to establish the call has been received (for an incoming call).
Active	On the called side, the network has indicated that the calling user has been awarded the call. On the calling side, the remote user has answered the call.
Disconnect Request	User has requested that the network clear the end-to-end call and is waiting for a response.
Disconnect Indication	User has received an invitation to disconnect the call because the network has disconnected the call.
Release Request	User has requested that the network release the call and is waiting for a response.

Related Commands

Command	Description
debug fmsp receive	Displays debugging information about the packets that are received on a Frame Relay interface.
debug frame-relay networklayerinterface	Displays NLI information.

debug frame-relay end-to-end keepalive

To display debug messages for the Frame Relay End-to-End Keepalive feature, use the **debug frame-relay end-to-end keepalive** command. Use the **no** form of this command to disable the display of debug messages.

debug frame-relay end-to-end keepalive {**events** | **packet**}

no debug frame-relay end-to-end keepalive {**events** | **packet**}

Syntax Description

events	Displays keepalive events.
packet	Displays keepalive packets sent and received.

Command History

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

Usage Guidelines

We recommend that both commands be enabled.

Examples

The following examples show typical output from the **debug frame-relay end-to-end keepalive packet** command. The following example shows output for an outgoing request packet:

```
EEK (o, Serial0.1 DLCI 200): 1 1 1 3 2 4 3
```

The seven number fields that follow the colon signify the following:

Field	Description
first (example value = 1)	Information Element (IE) type.
second (example value = 1)	IE length.
third (example value = 1)	Report ID. 1 = request, 2 = reply.
fourth (example value = 3)	Next IE type. 3 = LIV ID (Keepalive ID).
fifth (example value = 2)	IE length. (This IE is a Keepalive IE.)
sixth (example value = 4)	Send sequence number.
seventh (example value = 3)	Receive sequence number.

The following example shows output for an incoming reply packet:

```
EEK (i, Serial0.1 DLCI 200): 1 1 2 3 2 4 4
```

The seven number fields that follow the colon signify the following:

Field	Description
first (example value = 1)	Information Element (IE) type.
second (example value = 1)	IE length.
third (example value = 2)	Report ID. 1 = request, 2 = reply.

Field	Description
fourth (example value = 3)	Next IE type. 3 = LIV ID (Keepalive ID).
fifth (example value = 2)	IE length. (This IE is a Keepalive IE.)
sixth (example value = 4)	Send sequence number.
seventh (example value = 4)	Receive sequence number.

The following example shows typical output from the **debug frame-relay end-to-end keepalive events** command:

```
EEK SUCCESS (request, Serial0.2 DLCI 400)
EEK SUCCESS (reply, Serial0.1 DLCI 200)
EEK sender timeout (Serial0.1 DLCI 200)
```

debug frame-relay events

To display debugging information about Frame Relay ARP replies on networks that support a multicast channel and use dynamic addressing, use the **debug frame-relay events** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay events

no debug frame-relay events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is useful for identifying the cause of end-to-end connection problems during the installation of a Frame Relay network or node.

Note Because the **debug frame-relay events** command does not generate much output, you can use it at any time, even during periods of heavy traffic, without adversely affecting other users on the system.

Examples

The following is sample output from the **debug frame-relay events** command:

```
Router# debug frame-relay events

Serial2(i): reply rcvd 172.16.170.26 126
Serial2(i): reply rcvd 172.16.170.28 128
Serial2(i): reply rcvd 172.16.170.34 134
Serial2(i): reply rcvd 172.16.170.38 144
Serial2(i): reply rcvd 172.16.170.41 228
Serial2(i): reply rcvd 172.16.170.65 325
```

As the output shows, the **debug frame-relay events** command returns one specific message type. The first line, for example, indicates that IP address 172.16.170.26 sent a Frame Relay ARP reply; this packet was received as input on serial interface 2. The last field (126) is the data-link connection identifier (DLCI) to use when communicating with the responding router.

debug frame-relay fragment

To display information related to Frame Relay fragmentation on a PVC, use the **debug frame-relay fragment** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug frame-relay fragment [**event** | **interface** *type number dlci*]

no debug frame-relay fragment [**event** | **interface** *type number dlci*]

Syntax Description

event	(Optional) Displays event or error messages related to Frame Relay fragmentation.
interface	(Optional) Displays fragments received or sent on the specified interface.
type	(Optional) The interface type for which you wish to display fragments received or sent.
number	(Optional) The Interface number.
dlci	(Optional) The DLCI value of the PVC for which you wish to display fragments received or sent.

Command History

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

Usage Guidelines

This command will display event or error messages related to Frame Relay fragmentation; it is only enabled at the PVC level on the selected interface.

This command is not supported on the Cisco MC3810 networking device for fragments received by a PVC configured via the **voice-encap** command.

Examples

The following example shows sample output from the **debug frame-relay fragment** command:

```
Router# debug frame-relay fragment interface serial 0/0 109

This may severely impact network performance.
You are advised to enable 'no logging console debug'. Continue?[confirm]
Frame Relay fragment/packet debugging is on
Displaying fragments/packets on interface Serial0/0  dlci 109 only


Serial0/0(i): dlci 109, rx-seq-num 126, exp_seq-num 126, BE bits set, frag_hdr 04 C0 7E

Serial0/0(o): dlci 109, tx-seq-num 82, BE bits set, frag_hdr 04 C0 52
```

The following example shows sample output from the **debug frame-relay fragment event** command:

```
Router# debug frame-relay fragment event

This may severely impact network performance.
You are advised to enable 'no logging console debug'. Continue?[confirm]
Frame Relay fragment event/errors debugging is on

Frame-relay reassembled packet is greater than MTU size, packet dropped on serial 0/0
      dlci 109
```

```
Unexpected B bit  frame rx on serial0/0 dlci 109, dropping pending segments

Rx an out-of-sequence packet on serial 0/0 dlci 109, seq_num_received 17
      seq_num_expected 19
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccsip all	Displays the ccswvoice function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug voice vofr	Displays Cisco trunk and FRF.11 trunk call setup attempts; shows which dial peer is used in the call setup.
debug vpm error	Displays the behavior of the Holst state machine.
debug vtsp port	Displays the behavior of the VTSP state machine.
debug vtsp vofr subframe	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug frame-relay foresight

To observe Frame Relay traces relating to traffic shaping with router ForeSight enabled, use the **debug frame-relay foresight** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay foresight

no debug frame-relay foresight

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output that shows the display message returned in response to the **debug frame-relay foresight** command:

```
Router# debug frame-relay foresight

FR rate control for DLCI 17 due to ForeSight msg
```

This message indicates the router learned from the ForeSight message that DLCI 17 is now experiencing congestion. The output rate for this circuit should be slowed down, and in the router this DLCI is configured to adapt traffic shaping in response to foresight messages.

Related Commands

Command	Description
show frame-relay pvc	Displays statistics about PVCs for Frame Relay interfaces.

debug frame-relay informationelements

To display information about Frame Relay Layer 3 (network layer) information element parsing and construction, use the **debug frame-relay informationelements** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay informationelements

no debug frame-relay informationelements

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Within the FRF.4/Q.933 signalling specification, messages are divided into subunits called information elements. Each information element defines parameters specific to the call. These parameters can be values configured on the router, or values requested from the network.

The **debug frame-relay informationelements** command shows the signalling message in hexadecimal format. Use this command to determine parameters being requested and granted for a call.

Note Use the **debug frame-relay informationelements** command when the **debug frame-relay callcontrol** command does not explain why calls are not being set up.

Caution The **debug frame-relay informationelements** command displays a substantial amount of information in bytes. You must be familiar with FRF.4/Q.933 to decode the information contained within the debug output.

Examples

The following is sample output from the **debug frame-relay informationelements** command. In this example, each information element has a length associated with it. For those with odd-numbered lengths, only the specified bytes are valid, and the extra byte is invalid. For example, in the message "Call Ref, length: 3, 0x0200 0x0100," only "02 00 01" is valid; the last "00" is invalid.

```
1w0d# debug frame-relay informationelements

Router: Outgoing MSG_SETUP

Router: Dir: U --> N, Type: Prot Disc, length: 1, 0x0800
Router: Dir: U --> N, Type: Call Ref, length: 3, 0x0200 0x0100
Router: Dir: U --> N, Type: Message type, length: 1, 0x0500
Router: Dir: U --> N, Type: Bearer Capability, length: 5, 0x0403 0x88A0 0xCF00
Router: Dir: U --> N, Type: DLCI, length: 4, 0x1902 0x46A0
Router: Dir: U --> N, Type: Link Lyr Core, length: 27, 0x4819 0x090B 0x5C0B 0xDC0A
Router:                  0x3140 0x31C0 0x0B21 0x4021
Router:                  0xC00D 0x7518 0x7598 0x0E09
Router:                  0x307D 0x8000
Router: Dir: U --> N, Type: Calling Party, length: 12, 0x6C0A 0x1380 0x3837 0x3635
Router:                  0x3433 0x3231
Router: Dir: U --> N, Type: Calling Party Subaddr, length: 4, 0x6D02 0xA000
Router: Dir: U --> N, Type: Called Party, length: 11, 0x7009 0x9331 0x3233 0x3435
Router:                  0x3637 0x386E
Router: Dir: U --> N, Type: Called Party Subaddr, length: 4, 0x7102 0xA000
```

```
Router: Dir: U --> N, Type: Low Lyr Comp, length: 5, 0x7C03 0x88A0 0xCE65
Router: Dir: U --> N, Type: User to User, length: 4, 0x7E02 0x0000
```

Table 51 explains the information elements in the example shown.

Table 51 Information Elements in a Setup Message

Information Element	Description
Prot Disc	Protocol discriminator.
Call Ref	Call reference.
Message type	Message type such as *setup*, *connect*, and *call proceeding*.
Bearer Capability	Coding format such as data type, and Layer 2 and Layer 3 protocols.
DLCI	Data-link connection identifier.
Link Lyr Core	Link-layer core quality of service (QoS) requirements.
Calling Party	Type of source number (X121/E164) and the number.
Calling Party Subaddr	Subaddress that originated the call.
Called Party	Type of destination number (X121/E164) and the number.
Called Party Subaddr	Subaddress of the called party.
Low Lyr Comp	Coding format, data type, and Layer 2 and Layer 3 protocols intended for the end user.
User to User	Information between end users.

Related Commands

Command	Description
debug frame-relay callcontrol	Displays Frame Relay Layer 3 (network layer) call control information.

debug frame-relay lapf

To display Frame Relay switched virtual circuit (SVC) Layer 2 information, use the **debug frame-relay lapf** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay lapf

no debug frame-relay lapf

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug frame-relay lapf** command to troubleshoot the data-link control portion of Layer 2 that runs over data-link connection identifier (DLCI) 0. Use this command only if you have a problem bringing up Layer 2. You can use the **show interface serial** command to determine the status of Layer 2. If it shows a Link Access Procedure, Frame Relay (LAPF) state of down, Layer 2 has a problem.

Examples

The following is sample output from the **debug frame-relay lapf** command. In this example, a line being brought up indicates an exchange of set asynchronous balanced mode extended (SABME) and unnumbered acknowledgment (UA) commands. A SABME is initiated by both sides, and a UA is the response. Until the SABME gets a UA response, the line is not declared to be up. The p/f value indicates the poll/final bit setting. TX means send, and RX means receive.

```
Router# debug frame-relay lapf

Router: *LAPF Serial0 TX -> SABME Cmd p/f=1
Router: *LAPF Serial0 Enter state 5
Router: *LAPF Serial0 RX <- UA Rsp p/f=1
Router: *LAPF Serial0 lapf_ua_5
Router: *LAPF Serial0 Link up!
Router: *LAPF Serial0 RX <- SABME Cmd p/f=1
Router: *LAPF Serial0 lapf_sabme_78
Router: *LAPF Serial0 TX -> UA Rsp p/f=1
```

In the following example, a line in an up LAPF state should see a steady exchange of RR (receiver ready) messages. TX means send, RX means receive, and N(R) indicates the receive sequence number.

```
Router# debug frame-relay lapf

Router: *LAPF Serial0 T203 expired, state = 7
Router: *LAPF Serial0 lapf_rr_7
Router: *LAPF Serial0 TX -> RR Rsp p/f=1, N(R)= 3
Router: *LAPF Serial0 RX <- RR Cmd p/f=1, N(R)= 3
Router: *LAPF Serial0 lapf_rr_7
Router: *LAPF Serial0 TX -> RR Rsp p/f=1, N(R)= 3
Router: *LAPF Serial0 RX <- RR Cmd p/f=1, N(R)= 3
Router: *LAPF Serial0 lapf_rr_7
```

debug frame-relay lmi

To display information on the local management interface (LMI) packets exchanged by the router and the Frame Relay service provider, use the **debug frame-relay lmi** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay lmi [**interface** *name*]

no debug frame-relay lmi [**interface** *name*]

Syntax Description

interface *name*	(Optional) The name of interface.

Usage Guidelines

You can use this command to determine whether the router and the Frame Relay switch are sending and receiving LMI packets properly.

Note Because the **debug frame-relay lmi** command does not generate much output, you can use it at any time, even during periods of heavy traffic, without adversely affecting other users on the system.

Examples

The following is sample output from the **debug frame-relay lmi** command:

```
router# debug frame-relay lmi
```

LMI exchange

```
Serial1(out): StEnq, clock 20212760, myseq 206, mineseen 205, yourseen 136, DTE up
Serial1(in): Status, clock 20212764, myseq 206
RT IE 1, length 1, type 1
KA IE 3, length 2, yourseq 138, myseq 206
Serial1(out): StEnq, clock 20222760, myseq 207, mineseen 206, yourseen 138, DTE up
Serial1(in): Status, clock 20222764, myseq 207
RT IE 1, length 1, type 1
KA IE 3, length 2, yourseq 140, myseq 207
Serial1(out): clock 20232760, myseq 208, mineseen 207, yourseen 140, line up
RT IE 1, length 1, type 1
KA IE 3, length 2, yourseq 142, myseq 208
```

Full LMI status message

```
Serial1(out): StEnq, clock 20252760, myseq 210, mineseen 209, yourseen 144, DTE up
Serial1(in): Status, clock 20252764,
RT IE 1, length 1, type 0
KA IE 3, length 2, yourseq 146, myseq 210
PVC IE 0x7, length 0x6, dlci 400, status 0, bw 56000
PVC IE 0x7, length 0x6, dlci 401, status 0, bw 56000
```

S2546

The first four lines describe an LMI exchange. The first line describes the LMI request the router has sent to the switch. The second line describes the LMI reply the router has received from the switch. The third and fourth lines describe the response to this request from the switch. This LMI exchange is followed by two similar LMI exchanges. The last six lines consist of a full LMI status message that includes a description of the two permanent virtual circuits (PVCs) of the router.

Table 52 describes significant fields shown in the first line of the display.

Table 52 debug frame-relay lmi Field Descriptions

Field	Description
Serial1(out)	Indicates that the LMI request was sent out on serial interface 1.
StEnq	Command mode of message, as follows: • StEnq—Status inquiry • Status—Status reply
clock 20212760	System clock (in milliseconds). Useful for determining whether an appropriate amount of time has transpired between events.
myseq 206	Myseq counter maps to the CURRENT SEQ counter of the router.
yourseen 136	Yourseen counter maps to the LAST RCVD SEQ counter of the switch.
DTE up	Line protocol up/down state for the DTE (user) port.

Table 53 describes the significant fields shown in the third and fourth lines of the display.

Table 53 debug frame-relay lmi Field Descriptions

Field	Description
RT IE 1	Value of the report type information element.
length 1	Length of the report type information element (in bytes).
type 1	Report type in RT IE.
KA IE 3	Value of the keepalive information element.
length 2	Length of the keepalive information element (in bytes).
yourseq 138	Yourseq counter maps to the CURRENT SEQ counter of the switch.
myseq 206	Myseq counter maps to the CURRENT SEQ counter of the router.

Table 54 describes the significant fields shown in the last line of the display.

Table 54 debug frame-relay lmi Field Descriptions

Field	Description
PVC IE 0x7	Value of the PVC information element type.
length 0x6	Length of the PVC IE (in bytes).

Table 54 debug frame-relay lmi Field Descriptions (continued)

Field	Description
dlci 401	DLCI decimal value for this PVC.
status 0	Status value. Possible values include the following: • 0x00—Added/inactive • 0x02—Added/active • 0x04—Deleted • 0x08—New/inactive • 0x0a—New/active
bw 56000	Committed information rate (in decimal) for the DLCI.

debug frame-relay networklayerinterface

To display Network Layer Interface (NLI) information, use the **debug frame-relay networklayerinterface** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay networklayerinterface

no debug frame-relay networklayerinterface

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The Frame Relay SVC signalling subsystem is decoupled from the rest of the router code by means of the NLI intermediate software layer.

The **debug frame-relay networklayerinterface** command shows activity within the network-layer interface when a call is set up or torn down. All output that contains an *NL* relates to the interaction between the Q.933 signalling subsystem and the NLI.

Note The **debug frame-relay networklayerinterface** command has no significance to anyone not familiar with the inner workings of the Cisco IOS software. This command is typically used by service personnel to debug problem situations.

Examples

The following is sample output from the **debug frame-relay networklayerinterface** command. This example displays the output generated when a call is set up. The second example shows the output generated when a call is torn down.

```
Router# debug frame-relay networklayerinterface

Router: NLI STATE: L3_CALL_REQ, Call ID 1 state 0
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: Walking the event table 4
Router: NLI: Walking the event table 5
Router: NLI: Walking the event table 6
Router: NLI: Walking the event table 7
Router: NLI: Walking the event table 8
Router: NLI: Walking the event table 9
Router: NLI: NL0_L3CallReq
Router: NLI: State: STATE_NL_NULL, Event: L3_CALL_REQ, Next: STATE_L3_CALL_REQ
Router: NLI: Enqueued outgoing packet on holdq
Router: NLI: Map-list search: Found maplist bermuda
Router: daddr.subaddr 0, saddr.subaddr 0, saddr.subaddr 0
Router: saddr.subaddr 0, daddr.subaddr 0, daddr.subaddr 0
Router: nli_parameter_negotiation
Router: NLI STATE: NL_CALL_CNF, Call ID 1 state 10
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: NLx_CallCnf
Router: NLI: State: STATE_L3_CALL_REQ, Event: NL_CALL_CNF, Next: STATE_NL_CALL_CNF
```

```
Router: Checking maplist "junk"
Router: working with maplist "bermuda"
```

```
Router: Checking maplist "bermuda"
Router: working with maplist "bermuda"
Router: NLI: Emptying holdQ, link 7, dlci 100, size 104

Router# debug frame-relay networklayerinterface

Router: NLI: L3 Call Release Req for Call ID 1
Router: NLI STATE: L3_CALL_REL_REQ, Call ID 1 state 3
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: Walking the event table 4
Router: NLI: Walking the event table 5
Router: NLI: Walking the event table 6
Router: NLI: Walking the event table 7
Router: NLI: Walking the event table 8
Router: NLI: Walking the event table 9
Router: NLI: Walking the event table 10
Router: NLI: NLx_L3CallRej
Router: NLI: State: STATE_NL_CALL_CNF, Event: L3_CALL_REL_REQ, Next: STATE_L3_CALL_REL_REQ
Router: NLI: junk: State: STATE_NL_NULL, Event: L3_CALL_REL_REQ, Next: STATE_NL_NULL
Router: NLI: Map-list search: Found maplist junk
Router: daddr.subaddr 0, saddr.subaddr 0, saddr.subaddr 0
Router: saddr.subaddr 0, daddr.subaddr 0, daddr.subaddr 0
Router: nli_parameter_negotiation
Router: NLI STATE: NL_REL_CNF, Call ID 1 state 0
Router: NLI: Walking the event table 1
Router: NLI: Walking the event table 2
Router: NLI: Walking the event table 3
Router: NLI: Walking the event table 4
Router: NLI: Walking the event table 5
Router: NLI: Walking the event table 6
Router: NLI: Walking the event table 7
Router: NLI: NLx_RelCnf
Router: NLI: State: STATE_NL_NULL, Event: NL_REL_CNF, Next: STATE_NL_NULL
```

Table 55 describes the significant states and events shown in the display.

Table 55 NLI State and Event Descriptions

State and Event	Description
L3_CALL_REQ	Internal call setup request. Network layer indicates that a switched virtual circuit (SVC) is required.
STATE_NL_NULL	Call in initial state—no call exists.
STATE_L3_CALL_REQ	Setup message sent out and waiting for a reply. This is the state the network-layer state machine changes to when a call request is received from Layer 3 but no confirmation has been received from the network.
NL_CALL_CNF	Message sent from the Q.933 signalling subsystem to the NLI asking that internal resources be allocated for the call.
STATE_L3_CALL_CNF	Q.933 state indicating that the call is active. After the network confirms a call request using a connect message, the Q.933 state machine changes to this state.
STATE_NL_CALL_CNF	Internal software state indicating that software resources are assigned and the call is up. After Q.933 changes to the STATE_L3_CALL_CNF state, it sends an NL_CALL_CNF message to the network-layer state machine, which then changes to the STATE_NL_CALL_CNF state.

Table 55 NLI State and Event Descriptions (continued)

State and Event	Description
L3_CALL_REL_REQ	Internal request to release the call.
STATE_L3_CALL_REL_REQ	Internal software state indicating the call is in the process of being released. At this point, the Q.933 subsystem is told that the call is being released and a disconnect message goes out for the Q.933 subsystem.
NL_REL_CNF	Indication from the Q.933 signalling subsystem that the signalling subsystem is releasing the call. After receiving a release complete message from the network indicating that the release process is complete, the Q.933 subsystem sends an NL_REL_CNF event to the network-layer subsystem.

Related Commands

Command	Description
debug frame-relay callcontrol	Displays Frame Relay Layer 3 (network layer) call control information.

debug frame-relay packet

To display information on packets that have been sent on a Frame Relay interface, use the **debug frame-relay packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug frame-relay packet [**interface** *name* [**dlci** *value*]]

no debug frame-relay packet [**interface** *name* [**dlci** *value*]]

Syntax Description

interface *name*	(Optional) Name of interface or subinterface.
dlci *value*	(Optional) Data-link connection indentifier (DLCI) decimal value.

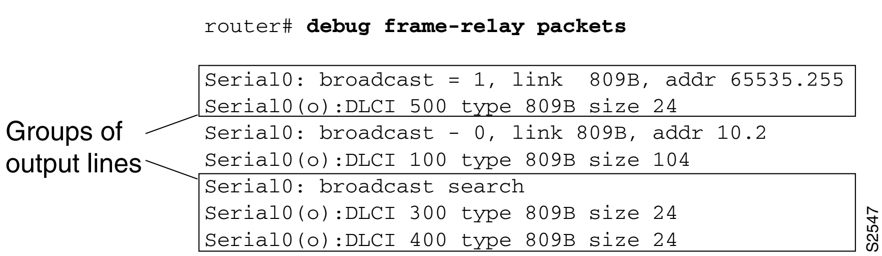
Usage Guidelines

This command helps you analyze the packets that are sent on a Frame Relay interface. Because the **debug frame-relay packet** command generates a substantial amount of output, only use it when traffic on the Frame Relay network is fewer than 25 packets per second. Use the options to limit the debugging output to a specific DLCI or interface.

To analyze the packets *received* on a Frame Relay interface, use the **debug frame-relay** command.

Examples

The following is sample output from the **debug frame-relay packet** command:



The **debug frame-relay packet** output consists of groups of output lines; each group describes a Frame Relay packet that has been sent. The number of lines in the group can vary, depending on the number of DLCIs on which the packet was sent. For example, the first two pairs of output lines describe two different packets, both of which were sent out on a single DLCI. The last three lines describe a single Frame Relay packet that was sent out on two DLCIs.

Table 56 describes the significant fields shown in the display.

Table 56 debug frame-relay packet Field Descriptions

Field	Description
Serial0:	Interface that has sent the Frame Relay packet.
broadcast = 1	Destination of the packet. Possible values include the following: • broadcast = 1—Broadcast address • broadcast = 0—Particular destination • broadcast search—Searches all Frame Relay map entries for this particular protocol that include the **broadcast** keyword.
link 809B	Link type, as documented in the **debug frame-relay** command.
addr 65535.255	Destination protocol address for this packet. In this case, it is an AppleTalk address.
Serial0(o):	(o) indicates that this is an output event.
DLCI 500	Decimal value of the DLCI.
type 809B	Packet type, as documented under the **debug frame-relay** command.
size 24	Size of this packet (in bytes).

The following lines describe a Frame Relay packet sent to a particular address; in this case AppleTalk address 10.2:

```
Serial0: broadcast - 0, link 809B, addr 10.2
Serial0(o):DLCI 100 type 809B size 104
```

The following lines describe a Frame Relay packet that went out on two different DLCIs, because two Frame Relay map entries were found:

```
Serial0: broadcast search
Serial0(o):DLCI 300 type 809B size 24
Serial0(o):DLCI 400 type 809B size 24
```

The following lines do not appear. They describe a Frame Relay packet sent to a true broadcast address.

```
Serial1: broadcast search
Serial1(o):DLCI 400 type 800 size 288
```

debug frame-relay ppp

To display debugging information, use the **debug frame-relay ppp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug frame-relay ppp

no debug frame-relay ppp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command displays error messages for link states and LMI status changes for PPP over Frame Relay sessions.

To debug process-switched packets, use the **debug frame-relay packet** or **debug ppp packet** commands. To analyze the packets that have been *sent* on a Frame Relay interface, use the **debug frame-relay packet** command.

The **debug frame-relay ppp** command is generated from process-level switching only and is not CPU intensive.

Examples

The following shows output from the **debug frame-relay ppp** command where the encapsulation failed for VC 100.

```
Router# debug frame-relay ppp
```

```
FR-PPP: encaps failed for FR VC 100 on Serial0 down
FR-PPP: input- Serial0 vc or va down, pak dropped
```

The following shows the output from the **debug frame relay ppp** and **debug frame-relay packet** commands**.** This example shows a virtual interface (virtual interface 1) establishing a PPP connection over PPP.

```
Router# debug frame-relay ppp
```

```
Router# debug frame-relay packet
```

```
Vi1 LCP: O CONFREQ [Closed] id 1 len 10
Vi1 LCP:    MagicNumber 0xE0638565 (0x0506E0638565)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0xC021, datagramsize 14
Vi1 LCP: I CONFACK [REQsent] id 1 len 10
Vi1 LCP:    MagicNumber 0xE0638565 (0x0506E0638565)
Vi1 PPP: I pkt type 0xC021, datagramsize 14
Vi1 LCP: I CONFREQ [ACKrcvd] id 6 len 10
Vi1 LCP:    MagicNumber 0x000EAD99 (0x0506000EAD99)
Vi1 LCP: O CONFACK [ACKrcvd] id 6 len 10
Vi1 LCP:    MagicNumber 0x000EAD99 (0x0506000EAD99)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 IPCP: O CONFREQ [Closed] id 1 len 10
Vi1 IPCP:    Address 170.100.9.10 (0x0306AA64090A)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0x8021, datagramsize 14
Vi1 IPCP: I CONFREQ [REQsent] id 1 len 10
```

```
Vi1 IPCP:    Address 170.100.9.20 (0x0306AA640914)
Vi1 IPCP: O CONFACK [REQsent] id 1 len 10
Vi1 IPCP:    Address 170.100.9.20 (0x0306AA640914)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0x8021, datagramsize 14
Vi1 IPCP: I CONFACK [ACKsent] id 1 len 10
Vi1 IPCP:    Address 170.100.9.10 (0x0306AA64090A)
Vi1 PPP: I pkt type 0xC021, datagramsize 16
Vi1 LCP: I ECHOREQ [Open] id 1 len 12 magic 0x000EAD99
Vi1 LCP: O ECHOREP [Open] id 1 len 12 magic 0xE0638565
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 18
Vi1 LCP: O ECHOREQ [Open] id 1 len 12 magic 0xE0638565
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 18
Vi1 LCP: echo_cnt 4, sent id 1, line up
```

The following shows the output for the **debug frame-relay ppp** and **debug frame-relay packet** commands that report a failed PPP over Frame Relay session. The problem is due to a challenge handshake authentication protocol (CHAP) failure.

```
Router# debug frame-relay ppp

Router# debug frame-relay packet

Vi1 LCP: O CONFREQ [Listen] id 24 len 10
Vi1 LCP:    MagicNumber 0xE068EC78 (0x0506E068EC78)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 16
Vi1 PPP: I pkt type 0xC021, datagramsize 19
Vi1 LCP: I CONFREQ [REQsent] id 18 len 15
Vi1 LCP:    AuthProto CHAP (0x0305C22305)
Vi1 LCP:    MagicNumber 0x0014387E (0x05060014387E)
Vi1 LCP: O CONFACK [REQsent] id 18 len 15
Vi1 LCP:    AuthProto CHAP (0x0305C22305)
Vi1 LCP:    MagicNumber 0x0014387E (0x05060014387E)
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 21
Vi1 PPP: I pkt type 0xC021, datagramsize 14
Vi1 LCP: I CONFACK [ACKsent] id 24 len 10
Vi1 LCP:    MagicNumber 0xE068EC78 (0x0506E068EC78)
Vi1 PPP: I pkt type 0xC223, datagramsize 32
Vi1 CHAP: I CHALLENGE id 12 len 28 from "krishna"
Vi1 LCP: O TERMREQ [Open] id 25 len 4
Serial2/1(o): dlci 201(0x3091), NLPID 0x3CF(PPP), datagramsize 10
Vi1 PPP: I pkt type 0xC021, datagramsize 8
Vi1 LCP: I TERMACK [TERMsent] id 25 len 4
Serial2/1(i): dlci 201(0x3091), pkt type 0x2000, datagramsize 303
%SYS-5-CONFIG_I: Configured from console by console
Vi1 LCP: TIMEout: Time 0x199580 State Listen
```

debug frame-relay switching

To display debug messages for switched Frame Relay PVCs, use the **debug frame-relay switching** EXEC command. To disable Frame Relay switching debugging, use the **no** form of this command.

debug frame-relay switching interface *interface dlci* [**interval** *interval*]

no debug frame-relay switching

Syntax Description

interface *interface*	The name of the Frame Relay interface.
dlci	The DLCI number of the switched PVC to be debugged.
interval *interval*	(Optional) Interval in seconds at which debugging messages will be updated.

Defaults

The default interval is 1 second.

Command History

Release	Modification
12.0(12)S	This command was introduced.
12.1(5)T	This command was implemented in Cisco IOS Release 12.1(5)T.

Usage Guidelines

The **debug frame-relay switching** command can be used only on switched Frame Relay PVCs, not terminated PVCs.

Debug statistics are displayed only if they have changed.

Note Although statistics are displayed at configured intervals, there may be a delay between the occurrence of a debug event (such as a packet drop) and the display of that event. The delay may be as much as the configured interval plus 10 seconds.

Examples

The following example shows sample output for the **debug frame-relay switching** command:

```
Router# debug frame-relay switching interface s2/1 1000 interval 2

 Frame Relay switching debugging is on
 Display frame switching debug on interface Serial2/1 dlci 1000
 1d02h: Serial2/1 dlci 1000: 32 packets switched to Serial2/0 dlci 1002
 1d02h: Serial2/1 dlci 1000: 1800 packets output
 1d02h: Serial2/1 dlci 1000: 4 packets dropped - outgoing PVC inactive
 1d02h: Serial2/1 dlci 1000: Incoming PVC status changed to ACTIVE
 1d02h: Serial2/1 dlci 1000: Outgoing PVC status changed to ACTIVE
 1d02h: Serial2/1 dlci 1000: Incoming interface hardware module state changed to UP
 1d02h: Serial2/1 dlci 1000: Outgoing interface hardware module state changed to UP
```

debug fras error

To display information about Frame Relay access support (FRAS) protocol errors, use the **debug fras error** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras error

no debug fras error

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For complete information on the FRAS process, use the **debug fras message** along with the **debug fras error** command.

Examples

The following is sample output from the **debug fras error** command. This example shows that no logical connection exists between the local station and remote station in the current setup:

```
Router# debug fras error

FRAS: No route, lmac 1000.5acc.7fb1 rmac 4fff.0000.0000, lSap=0x4, rSap=0x4
FRAS: Can not find the Setup
```

Related Commands

Command	Description
debug cls message	Displays information about CLS messages.
debug fras message	Displays general information about FRAS messages.
debug fras state	Displays information about FRAS data-link control state changes.

debug fras-host activation

To display the LLC2 session activation and deactivation frames (such as XID, SABME, DISC, UA) that are being handled by the FRAS host, use the **debug fras-host activation** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras-host activation

no debug fras-host activation

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

If many LLC2 sessions are being activated or deactivated at any time, this command may generate a substantial amount of output to the console.

Examples

The following is sample output from the **debug fras-host activation** command:

```
Router# debug fras-host activation

FRHOST: Snd      TST C to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x00 SSAP =
0x04
FRHOST: Fwd  BNN   XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd HOST   XID to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd  BNN   XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd HOST SABME to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN    UA to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x05
```

The first line indicates that the FRAS Host sent a TEST Command to the host. In the second line, the FRAS Host forwards an XID frame from a BNN device to the host. In the third line, the FRAS Host forwards an XID from the host to the BNN device.

Table 57 describes the significant fields shown in the display.

Table 57 debug fras-host activation Field Descriptions

Field	Description
DA	Destination MAC address of the frame.
SA	Source MAC address of the frame.
DSAP	Destination SAP of the frame.
SSAP	Source SAP of the frame.

debug fras-host error

To enable the FRAS Host to send error messages to the console, use the **debug fras-host error** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras-host error

no debug fras-host error

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug fras-host error** command when the I-field in a TEST Response frame from a host does not match the I-field of the TEST Command sent by the FRAS Host:

```
Router# debug fras-host error

FRHOST: SRB TST R Protocol Violation - LLC I-field not maintained.
```

debug fras-host packet

To see which LLC2 session frames are being handled by the FRAS Host, use the **debug fras-host packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras-host packet

no debug fras-host packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command with great care. If many LLC2 sessions are active and passing data, this command may generate a substantial amount of output to the console and impact device performance.

Examples

The following is sample output from the **debug fras-host packet** command:

```
Router# debug fras-host packet

FRHOST: Snd      TST C to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x00 SSAP =
0x04
FRHOST: Fwd  BNN   XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd HOST   XID to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd  BNN   XID to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd HOST SABME to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN    UA to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd HOST LLC-2 to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN LLC-2 to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x05
FRHOST: Fwd HOST LLC-2 to  BNN, DA = 400f.dddd.001e SA = 4001.3745.1088 DSAP = 0x04 SSAP =
0x04
FRHOST: Fwd  BNN LLC-2 to HOST, DA = 4001.3745.1088 SA = 400f.dddd.001e DSAP = 0x04 SSAP =
0x04
```

The **debug fras-host packet** output contains all of the output from the **debug fras-host activation** command and additional information. The first six lines of this sample display are the same as the output from the **debug fras-host activation** command. The last lines show LLC-2 frames being sent between the BNN device and the host.

The following describes the significant fields shown in the display.

Table 58 debug fras-host packet Field Descriptions

Field	Description
DA	Destination MAC address of the frame.
SA	Source MAC address of the frame.

Table 58 debug fras-host packet Field Descriptions (continued)

Field	Description
DSAP	Destination SAP of the frame.
SSAP	Source SAP of the frame.

debug fras-host snmp

To display messages to the console describing SNMP requests to the FRAS Host MIB, use the **debug fras-host snmp** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras-host snmp

no debug fras-host snmp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use of this command may result in a substantial amount of output to the screen. Only use this command for problem determination.

Examples

The following is sample output from the **debug fras-host snmp** command. In this example, the MIB variable k_frasHostConnEntry_get() is providing SNMP information for the FRAS host.

```
Router# debug fras-host snmp

k_frasHostConnEntry_get(): serNum = -1, vRingIfIdx = 31, frIfIdx = 12
Hmac = 4001.3745.1088, frLocSap = 4, Rmac = 400f.dddd.001e, frRemSap = 4
```

Table 59 describes the significant fields shown in the display.

Table 59 debug fras-host snmp Field Descriptions

Field	Description
serNum	Serial number of the SNMP request.
vRingIfIdx	Interface index of a virtual Token Ring.
frIfIdx	Interface index of a Frame Relay serial interface.
Hmac	MAC address associated with the host for this connection.
frLocSap	SAP associated with the host for this connection.
Rmac	MAC address associated with the FRAD for this connection.
frRemSap	LLC 2 SAP associated with the FRAD for this connection.

debug fras message

To display general information about Frame Relay access support (FRAS) messages, use the **debug fras message** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras message

no debug fras message

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For complete information on the FRAS process, use the **debug fras error** command along with the **debug fras message** command.

Examples

The following is sample output from the **debug fras message** command. This example shows incoming Cisco Link Services (CLS) primitives:

```
Router# debug fras message

FRAS: receive 4C23
FRAS: receive CC09
```

Related Commands

Command	Description
debug cls message	Limits output for some debugging commands based on the interfaces.
debug fras error	Displays information about FRAS protocol errors.
debug fras state	Displays information about FRAS data-link control state changes.

debug fras state

To display information about Frame Relay access support (FRAS) data-link control link-state changes, use the **debug fras state** privileged EXEC command. The **no** form of this command disables debugging output.

debug fras state

no debug fras state

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug fras state** command. This example shows the state changing from a *request open station is sent* state to an *exchange XID* state.

Possible states are the following: reset, request open station is sent, exchange xid, connection request is sent, signal station wait, connection response wait, connection response sent, connection established, disconnect wait, and number of link states.

```
Router# debug fras state

FRAS: TR0 (04/04) oldstate=LS_RQOPNSTNSENT, input=RQ_OPNSTN_CNF
FRAS: newstate=LS_EXCHGXID
```

Related Commands

Command	Description
debug cls message	Limits output for some debugging commands based on the interfaces.
debug fras error	Displays information about FRAS protocol errors.
debug fras state	Displays general information about FRAS messages.

debug ftpserver

To display information about the FTP server process, use the **debug ftpserver** privileged EXEC command. The **no** form of this command disables debugging output.

debug ftpserver

no debug ftpserver

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ftpserver

Mar  3 10:21:10: %FTPSERVER-6-NEWCONN: FTP Server - new connection made.
-Process= "TCP/FTP Server", ipl= 0, pid= 53
Mar  3 10:21:10: FTPSRV_DEBUG:FTP Server file path: 'disk0:'
Mar  3 10:21:10: FTPSRV_DEBUG:(REPLY)  220
Mar  3 10:21:10: FTPSRV_DEBUG:FTProuter IOS-FTP server (version 1.00) ready.
Mar  3 10:21:10: FTPSRV_DEBUG:FTP Server Command received: 'USER aa'
Mar  3 10:21:20: FTPSRV_DEBUG:(REPLY)  331
Mar  3 10:21:20: FTPSRV_DEBUG:Password required for 'aa'.
Mar  3 10:21:20: FTPSRV_DEBUG:FTP Server Command received: 'PASS aa'
Mar  3 10:21:21: FTPSRV_DEBUG:(REPLY)  230
Mar  3 10:21:21: FTPSRV_DEBUG:Logged in.
Mar  3 10:21:21: FTPSRV_DEBUG:FTP Server Command received: 'SYST'
Mar  3 10:21:21: FTPSRV_DEBUG:(REPLY)  215
Mar  3 10:21:21: FTPSRV_DEBUG:Cisco IOS Type: L8 Version: IOS/FTP 1.00
Mar  3 10:21:21: FTPSRV_DEBUG:FTP Server Command received: 'PWD'
Mar  3 10:21:35: FTPSRV_DEBUG:(REPLY)  257
Mar  3 10:21:39: FTPSRV_DEBUG:FTP Server Command received: 'CWD disk0:/syslogd.d'r/'
Mar  3 10:21:45: FTPSRV_DEBUG:FTP Server file path: 'disk0:/syslogd.dir'
Mar  3 10:21:45: FTPSRV_DEBUG:(REPLY)  250
Mar  3 10:21:45: FTPSRV_DEBUG:CWD command successful.
Mar  3 10:21:45: FTPSRV_DEBUG:FTP Server Command received: 'PORT 171,69,30,20,22',32
Mar  3 10:21:46: FTPSRV_DEBUG:(REPLY)  200
Mar  3 10:21:46: FTPSRV_DEBUG:PORT command successful.
Mar  3 10:21:46: FTPSRV_DEBUG:FTP Server Command received: 'LIST'
Mar  3 10:21:47: FTPSRV_DEBUG:FTP Server file path: 'disk0:/syslogd.dir/.'
Mar  3 10:21:47: FTPSRV_DEBUG:(REPLY)  220
Mar  3 10:23:11: FTPSRV_DEBUG:Opening ASCII mode data connection for file list.
Mar  3 10:23:11: FTPSRV_DEBUG:(REPLY)  226
Mar  3 10:23:12: FTPSRV_DEBUG:Transfer complete.
Mar  3 10:23:12: FTPSRV_DEBUG:FTP Server Command received: 'TYPE I'
Mar  3 10:23:14: FTPSRV_DEBUG:(REPLY)  200
Mar  3 10:23:14: FTPSRV_DEBUG:Type set to I.
Mar  3 10:23:14: FTPSRV_DEBUG:FTP Server Command received: 'PORT 171,69,30,20,22',51
Mar  3 10:23:20: FTPSRV_DEBUG:(REPLY)  200
Mar  3 10:23:20: FTPSRV_DEBUG:PORT command successful.
Mar  3 10:23:20: FTPSRV_DEBUG:FTP Server Command received: 'RETR syslogd.1'
Mar  3 10:23:21: FTPSRV_DEBUG:FTP Server file path: 'disk0:/syslogd.dir/syslogd.1'
Mar  3 10:23:21: FTPSRV_DEBUG:FTPSERVER: Input path passed Top-dir(disk0:/syslogd.dir/)
test.
Mar  3 10:23:21: FTPSRV_DEBUG:(REPLY)  150
Mar  3 10:23:21: FTPSRV_DEBUG:Opening BINARY mode data connection for syslogd.1 (607317
bytes).
Mar  3 10:23:21: FTPSRV_DEBUG:(REPLY)  226
```

```
Mar  3 10:23:29: FTPSRV_DEBUG:Transfer complete.
```

The sample output corresponds to the following FTP client session. In this example, the user connects to the FTP server, views the contents of the top-level directory, and gets a file.

```
FTPclient% ftp FTProuter
Connected to FTProuter.cisco.com.
220 FTProuter IOS-FTP server (version 1.00) ready.
Name (FTProuter:me): aa
331 Password required for 'aa'.
Password:
230 Logged in.
Remote system type is Cisco.
ftp> pwd
257 "disk0:/syslogd.dir/" is current directory.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
syslogd.1
syslogd.2
syslogd.3
syslogd.4
syslogd.5
syslogd.6
syslogd.7
syslogd.8
syslogd.9
syslogd.cur
226 Transfer complete.
ftp> bin
200 Type set to I.
ftp> get syslogd.1
200 PORT command successful.
150 Opening BINARY mode data connection for syslogd.1 (607317 bytes).
226 Transfer complete.
607317 bytes received in 7.7 seconds (77 Kbytes/s)
ftp>
```

The following **debug ftpserver** command output indicates that no top-level directory is specified. Therefore, the client cannot access any location on the FTP server. Use the **ftp-server topdir** command to specify the top-level directory.

```
Mar  3 10:29:14: FTPSRV_DEBUG:(REPLY)  550
Mar  3 10:29:14: FTPSRV_DEBUG:Access denied to 'disk0:'
```

debug gatekeeper server

To trace all the message exchanges between the Cisco IOS Gatekeeper and the external applications, use the **debug gatekeeper server** command from EXEC mode. Enter the **no** form of this command to disable debugging output.

debug gatekeeper server

no debug gatekeeper server

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command Modes

EXEC

Command History

Table 1

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

Use this command to see information about a Gatekeeper server. This command shows any errors that occur in sending messages to the external applications or in parsing messages from the external applications.

Examples

The following example shows debugging information about a Gatekeeper server

```
Router# debug gatekeeper servers

Router# show debug

  Gatekeeper:
    Gatekeeper Server Messages debugging is on
```

To turn the Gatekeeper server debugging message off, see the following examples:

```
Router# no debug all
```

or

```
Router# no debug Gatekeeper servers
```

Related Commands

Command	Description
show gatekeeper server	Displays information about the Gatekeeper servers configured on your network by ID.

debug gprs charging

To display information about GPRS charging functions on the GGSN, use the **debug gprs charging events** command. To disable debugging output, use the **no** form of the command.

debug gprs charging {**events** | **packets**}

no debug gprs charging {**events** | **packets**}

Syntax Description

events	Displays events related to GPRS charging processing on the GGSN.
packets	Displays GPRS charging packets that are sent between the GGSN and the charging gateway.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

This command is useful for system operators if problems are encountered with GPRS charging functions.

Caution Because the **debug gprs charging** command generates a substantial amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of events related to GPRS charging events on the GGSN:

```
Router# debug gprs charging events
```

The following example enables the display of GPRS charging packets sent between the GGSN and the charging gateway:

```
Router# debug gprs charging events
```

debug gprs gtp

To display information about the GPRS Tunneling Protocol (GTP), use the **debug gprs gtp** command. To disable debugging output, use the **no** form of the command.

debug gprs gtp {**events** | **messages** | **packets**}

no debug gprs gtp {**events** | **messages** | **packets**}

Syntax Description

events	Displays events related to GTP processing on the GGSN.
messages	Displays GTP signalling messages that are sent between the SGSN and GGSN.
packets	Displays GTP packets that are sent between the SGSN and GGSN.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated in Cisco IOS Release 12.1(3)T.

Usage Guidelines

This command is useful for system operators and development engineers if problems are encountered with communication between the GGSN and the SGSN using GTP.

Caution Because the **debug gprs gtp** command generates a significant amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of events related to GTP processing on the GGSN:

```
Router# debug gprs gtp events
```

The following example enables the display of GTP signalling messages:

```
Router# debug gprs gtp messages
```

The following example enables the display of GTP packets sent between the SGSN and GGSN:

```
Router# debug gprs gtp packets
```

debug h225

To display additional information about the actual contents of H.225 RAS messages, use the **debug h225** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug h225 {**asn1** | **events**}

no debug h225 {**asn1** | **events**}

Syntax Description

asn1	Indicates that only the ASN.1 contents of any H.225 message sent or received will be displayed.
events	Indicates that key Q.931 events that occur when placing an H.323 call from one gateway to another will be displayed.

Command History

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

Usage Guidelines

Both versions of the **debug H225** command display information about H.225 messages. H.225 messages are used to exchange RAS information between gateways and gatekeepers and to exchange Q.931 information between gateways.

The **debug h225 events** command displays key Q.931 events that occur when placing an H.323 call from one gateway to another. Q.931 events are carried in H.225 messages. This command enables you to monitor Q.931 state changes such as setup, alert, connected, and released.

Note Although the debug information includes the hexadecimal output of the entire H.225 message, only the key state changes are decoded.

The **debug h225 asn1** command displays the ASN.1 contents of any H.225 message sent or received that contains ASN.1 content. Not all H.225 messages contain ASN.1 content. Some messages contain both Q.931 information and ASN.1 information; if you enter this command, only ASN.1 information will be displayed.

Examples

The following sample display for the **debug h225 events** command shows a call being placed from gateway GW13 to gateway GW14. Before the call was placed, the gateway exchanged RAS messages with the Gatekeeper. Because RAS messages do not contain Q.931 information, these messages do not appear in this output.

```
Router# debug h225 events

H.225 Event Messages debugging is on
Router#

*Mar 2 02:47:14.689:      H225Lib::h225TConn:connect in progress on socket [2]
*Mar 2 02:47:14.689:      H225Lib::h225TConn:Q.931 Call State is initialized to be [Null].
*Mar 2 02:47:14.697:Hex representation of the SETUP TPKT to
send.0300004D080200DC05040380C0A36C0991313323313333303070099131342331343330307E00260500800
60008914A000102004B1F5E5D8990006C0000000005BF7454000C0700000000000000
*Mar 2 02:47:14.701:
```

```
*Mar 2 02:47:14.701:      H225Lib::h225SetupRequest:Q.931 SETUP sent from socket [2]
*Mar 2 02:47:14.701:      H225Lib::h225SetupRequest:Q.931 Call State changed to [Call
Initiated].
*Mar 2 02:47:14.729:Hex representation of the received
TPKT03000021080280DC013401017E0012050340060008914A000100000109350E2B28
*Mar 2 02:47:14.729:
*Mar 2 02:47:14.729:      H225Lib::h225RecvData:Q.931 ALERTING received from socket [2]
*Mar 2 02:47:14.729:      H225Lib::h225RecvData:Q.931 Call State changed to [Call
Delivered].
*Mar 2 02:47:17.565:Hex representation of the received
TPKT03000034080280DC07040380C0A37E0023050240060008914A0001000109350E2B2802004B1F5E5D899000
6C0000000005BF7454
*Mar 2 02:47:17.569:
*Mar 2 02:47:17.569:      H225Lib::h225RecvData:Q.931 CONNECT received from socket [2]
*Mar 2 02:47:17.569:      H225Lib::h225RecvData:Q.931 Call State changed to [Active].
*Mar 2 02:47:23.273:Hex representation of the received
TPKT0300001A080280DC5A080280107E000A050500060008914A0001
*Mar 2 02:47:23.273:
*Mar 2 02:47:23.273:      H225Lib::h225RecvData:Q.931 RELEASE COMPLETE received from
socket [2]
*Mar 2 02:47:23.273:      H225Lib::h225RecvData:Q.931 Call State changed to [Null].
*Mar 2 02:47:23.293:Hex representation of the RELEASE COMPLETE TPKT to
send.0300001A080200DC5A080280107E000A050500060008914A0001
*Mar 2 02:47:23.293:
*Mar 2 02:47:23.293:      H225Lib::h225TerminateRequest:Q.931 RELEASE COMPLETE sent from
socket [2]. Call state changed to [Null].
*Mar 2 02:47:23.293:      H225Lib::h225TClose:TCP connection from socket [2] closed
```

The following output shows the same call being placed from gateway GW13 to gateway GW14 using the **debug h225 asn1** command. The output is very long but you can track the following information:

- The admission request to the Gatekeeper.
- The admission confirmation from the Gatekeeper.
- The ASN.1 portion of the H.225/Q.931 setup message from the calling gateway to the called gateway.
- The ASN.1 portion of the H.225/Q.931 setup response from the called gateway, indicating that the call has proceeded to alerting state.
- The ASN.1 portion of the H.225/Q.931 message from the called gateway, indicating that the call has been connected.
- The ASN.1 portion of the H.225/Q.931 message from the called gateway, indicating that the call has been released.
- The ANS.1 portion of the H.225 RAS message from the calling gateway to the Gatekeeper, informing it that the call has been disengaged.
- The ASN.1 portion of the H.225 RAS message from the Gatekeeper to the calling gateway, confirming the disengage request.
- The ASN.1 portion of the H.225/Q.931 release complete message sent from the called gateway to the calling gateway.

```
Router# debug h225 asn1

H.225 ASN1 Messages debugging is on
Router#

value RasMessage ::= admissionRequest :
*Mar 2 02:48:18.445: {
*Mar 2 02:48:18.445:    requestSeqNum 03320,
```

```
*Mar 2 02:48:18.445:    callType pointToPoint :NULL,
*Mar 2 02:48:18.445:    callModel direct :NULL,
*Mar 2 02:48:18.445:    endpointIdentifier "60D6BA4C00000001",
*Mar 2 02:48:18.445:    destinationInfo
*Mar 2 02:48:18.445:    {
*Mar 2 02:48:18.445:      e164 :"14#14300"
*Mar 2 02:48:18.445:    },
*Mar 2 02:48:18.449:    srcInfo
*Mar 2 02:48:18.449:    {
*Mar 2 02:48:18.449:      e164 :"13#13300"
*Mar 2 02:48:18.449:    },
*Mar 2 02:48:18.449:    bandWidth 0640,
*Mar 2 02:48:18.449:    callReferenceValue 0224,
*Mar 2 02:48:18.449:    conferenceID '4B1F5E5D8990007200000000005C067A4'H,
*Mar 2 02:48:18.449:    activeMC FALSE,
*Mar 2 02:48:18.449:    answerCall FALSE
*Mar 2 02:48:18.449: }
*Mar 2 02:48:18.449:25800CF7 00F00036 00300044 00360042 00410034 00430030 00300030
00300030
00300030 00310103 80470476 33010380 46046633 40028000 E04B1F5E 5D899000
72000000 0005C067 A400
29000CF7 40028000 0109350E 06B80077
value RasMessage ::= admissionConfirm :
*Mar 2 02:48:18.469: {
*Mar 2 02:48:18.469:    requestSeqNum 03320,
*Mar 2 02:48:18.469:    bandWidth 0640,
*Mar 2 02:48:18.469:    callModel direct :NULL,
*Mar 2 02:48:18.469:    destCallSignalAddress ipAddress :
*Mar 2 02:48:18.469:      {
*Mar 2 02:48:18.469:        ip '0109350E'H,
*Mar 2 02:48:18.469:        port 01720
*Mar 2 02:48:18.469:      },
*Mar 2 02:48:18.469:    irrFrequency 0120
*Mar 2 02:48:18.473: }
*Mar 2 02:48:18.473:value H323-UserInformation ::=
*Mar 2 02:48:18.481:{
*Mar 2 02:48:18.481: h323-uu-pdu
*Mar 2 02:48:18.481: {
*Mar 2 02:48:18.481:    h323-message-body setup :
*Mar 2 02:48:18.481:      {
*Mar 2 02:48:18.481:        protocolIdentifier { 0 0 8 2250 0 1 },
*Mar 2 02:48:18.481:        sourceInfo
*Mar 2 02:48:18.481:        {
*Mar 2 02:48:18.481:          terminal
*Mar 2 02:48:18.481:          {
*Mar 2 02:48:18.481:          },
*Mar 2 02:48:18.481:          mc FALSE,
*Mar 2 02:48:18.481:          undefinedNode FALSE
*Mar 2 02:48:18.481:        },
*Mar 2 02:48:18.481:        activeMC FALSE,
*Mar 2 02:48:18.481:        conferenceID '4B1F5E5D8990007200000000005C067A4'H,
*Mar 2 02:48:18.481:        conferenceGoal create :NULL,
*Mar 2 02:48:18.485:        callType pointToPoint :NULL,
*Mar 2 02:48:18.485:        sourceCallSignalAddress ipAddress :
*Mar 2 02:48:18.485:          {
*Mar 2 02:48:18.485:            ip '00000000'H,
*Mar 2 02:48:18.485:            port 00
*Mar 2 02:48:18.485:          }
*Mar 2 02:48:18.485:      }
*Mar 2 02:48:18.485: }
*Mar 2 02:48:18.485:}
*Mar 2 02:48:18.485:00800600 08914A00 0102004B 1F5E5D89 90007200 00000005 C067A400
0C070000
00000000 00
```

```
value H323-UserInformation ::=
*Mar 2 02:48:18.525:{
*Mar 2 02:48:18.525: h323-uu-pdu
*Mar 2 02:48:18.525: {
*Mar 2 02:48:18.525:    h323-message-body alerting :
*Mar 2 02:48:18.525:      {
*Mar 2 02:48:18.525:        protocolIdentifier { 0 0 8 2250 0 1 },
*Mar 2 02:48:18.525:        destinationInfo
*Mar 2 02:48:18.525:        {
*Mar 2 02:48:18.525:          mc FALSE,
*Mar 2 02:48:18.525:          undefinedNode FALSE
*Mar 2 02:48:18.525:        },
*Mar 2 02:48:18.525:        h245Address ipAddress :
*Mar 2 02:48:18.525:          {
*Mar 2 02:48:18.525:            ip '0109350E'H,
*Mar 2 02:48:18.525:            port 011050
*Mar 2 02:48:18.525:          }
*Mar 2 02:48:18.525:      }
*Mar 2 02:48:18.525: }
*Mar 2 02:48:18.525:}
*Mar 2 02:48:18.525:value H323-UserInformation ::=
*Mar 2 02:48:22.753:{
*Mar 2 02:48:22.753: h323-uu-pdu
*Mar 2 02:48:22.753: {
*Mar 2 02:48:22.753:    h323-message-body connect :
*Mar 2 02:48:22.753:      {
*Mar 2 02:48:22.753:        protocolIdentifier { 0 0 8 2250 0 1 },
*Mar 2 02:48:22.753:        h245Address ipAddress :
*Mar 2 02:48:22.753:          {
*Mar 2 02:48:22.753:            ip '0109350E'H,
*Mar 2 02:48:22.753:            port 011050
*Mar 2 02:48:22.753:          },
*Mar 2 02:48:22.753:        destinationInfo
*Mar 2 02:48:22.753:        {
*Mar 2 02:48:22.753:          terminal
*Mar 2 02:48:22.753:          {
*Mar 2 02:48:22.753:          },
*Mar 2 02:48:22.757:          mc FALSE,
*Mar 2 02:48:22.757:          undefinedNode FALSE
*Mar 2 02:48:22.757:        },
*Mar 2 02:48:22.757:        conferenceID '4B1F5E5D8990007200000000005C067A4'H
*Mar 2 02:48:22.757:      }
*Mar 2 02:48:22.757: }
*Mar 2 02:48:22.757:}
*Mar 2 02:48:22.757:value H323-UserInformation ::=
*Mar 2 02:48:27.109:{
*Mar 2 02:48:27.109: h323-uu-pdu
*Mar 2 02:48:27.109: {
*Mar 2 02:48:27.109:    h323-message-body releaseComplete :
*Mar 2 02:48:27.109:      {
*Mar 2 02:48:27.109:        protocolIdentifier { 0 0 8 2250 0 1 }
*Mar 2 02:48:27.109:      }
*Mar 2 02:48:27.109: }
*Mar 2 02:48:27.109:}
*Mar 2 02:48:27.109:value RasMessage ::= disengageRequest :
*Mar 2 02:48:27.117: {
*Mar 2 02:48:27.117:    requestSeqNum 03321,
*Mar 2 02:48:27.117:    endpointIdentifier "60D6BA4C00000001",
*Mar 2 02:48:27.117:    conferenceID '4B1F5E5D8990007200000000005C067A4'H,
*Mar 2 02:48:27.121:    callReferenceValue 0224,
*Mar 2 02:48:27.121:    disengageReason normalDrop :NULL
*Mar 2 02:48:27.121: }
*Mar 2 02:48:27.121:3C0CF81E 00360030 00440036 00420041 00340043 00300030 00300030
00300030
```

```
00300031 4B1F5E5D 89900072 00000000 05C067A4 00E020
400CF8
value RasMessage ::= disengageConfirm :
*Mar 2 02:48:27.133: {
*Mar 2 02:48:27.133:    requestSeqNum 03321
*Mar 2 02:48:27.133: }
*Mar 2 02:48:27.133:value H323-UserInformation ::=
*Mar 2 02:48:27.133:{
*Mar 2 02:48:27.133: h323-uu-pdu
*Mar 2 02:48:27.133: {
*Mar 2 02:48:27.133:    h323-message-body releaseComplete :
*Mar 2 02:48:27.133:      {
*Mar 2 02:48:27.133:        protocolIdentifier { 0 0 8 2250 0 1 }
*Mar 2 02:48:27.133:      }
*Mar 2 02:48:27.133: }
*Mar 2 02:48:27.133:}
*Mar 2 02:48:27.133:05000600 08914A00 01
.
```

debug h225 asn1

To display ASN1 contents of RAS and Q.931 messages, use the **debug h255 asn1** privileged EXEC command. The **no** form of this command disables debugging output.

debug h255 asn1

no debug h255 asn1

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)NA	This command was introduced.
12.0(3)T	This command was modified.

Usage Guidelines

Caution This command slows down the system considerably. Connections may time out.

Examples

Example 1

The following output shows two proxy call scenarios. A trace is collected on the gatekeeper with ASN1 turned on. The call is being established.

```
Router# debug h225 asn1

H.225 ASN1 Messages debugging is on
Router#24800006 03C00030 00300036 00380041 00450037 00430030 00300030 00300030
00300030 00310140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D020180 AAAA4006 00700074 0065006C 00320031 0033401E
0000015F C8490FB4 B9D111BF AF0060B0 00E94500
value RasMessage ::= admissionRequest :
  {
    requestSeqNum 7,
    callType pointToPoint : NULL,
    endpointIdentifier "0068AE7C00000001",
    destinationInfo
    {
      h323-ID : "ptel23@zone2.com"
    },
    srcInfo
    {
      e164 : "7777",
      h323-ID : "ptel213"
    },
    bandWidth 7680,
    callReferenceValue 1,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    activeMC FALSE,
    answerCall FALSE
  }
```

```
value RasMessage ::= admissionConfirm :
  {
    requestSeqNum 7,
    bandWidth 7680,
    callModel direct : NULL,
    destCallSignalAddress ipAddress :
      {
        ip '65000001'H,
        port 1720
      },
    irrFrequency 30
  }
29000006 401E0000 65000001 06B8001D
2480001D 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D014006 00700074 0065006C 00320031 00334002 8000015F
C8490FB4 B9D111BF AF0060B0 00E94540
value RasMessage ::= admissionRequest :
  {
    requestSeqNum 30,
    callType pointToPoint : NULL,
    endpointIdentifier "0068A96000000002",
    destinationInfo
    {
      h323-ID : "ptel23@zone2.com"
    },
    srcInfo
    {
      h323-ID : "ptel213"
    },
    bandWidth 640,
    callReferenceValue 1,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    activeMC FALSE,
    answerCall TRUE
  }
value ACFnonStandardInfo ::=
{
  srcTerminalAlias
  {
    e164 : "7777",
    h323-ID : "ptel213"
  },
  dstTerminalAlias
  {
    h323-ID : "ptel23@zone2.com"
  },
  dstProxyAlias
  {
    h323-ID : "px2"
  },
  dstProxySignalAddress
  {
    ip '66000001'H,
    port 1720
  }
}
C00203AA AA800600 70007400 65006C00 32003100 3301800F 00700074 0065006C
00320033 0040007A 006F006E 00650032 002E0063 006F006D 01800200 70007800
32660000 0106B8
value RasMessage ::= admissionConfirm :
  {
    requestSeqNum 30,
    bandWidth 7680,
```

```
      callModel direct : NULL,
      destCallSignalAddress ipAddress :
        {
          ip '66000001'H,
          port 1720
        },
      irrFrequency 30,
      nonStandardData
      {
        nonStandardIdentifier h221NonStandard :
          {
            t35CountryCode 181,
            t35Extension 0,
            manufacturerCode 18
          },
        data
'C00203AAAA80060070007400650006C0032003100330180OF007000740065006C003200 ...'H
      }
    }
2980001D 401E0000 66000001 06B8001D 40B50000 1247C002 03AAAA80 06007000
74006500 6C003200 31003301 800F0070 00740065 006C0032 00330040 007A006F
006E0065 0032002E 0063006F 006D0180 02007000 78003266 00000106 B8
24C0001E 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D006600 000106B8 020180AA AA400600 70007400 65006C00
32003100 33401E00 00435FC8 490FB4B9 D111BFAF 0060B000 E94500
value RasMessage ::= admissionRequest :
    {
      requestSeqNum 31,
      callType pointToPoint : NULL,
      endpointIdentifier "0068A96000000002",
      destinationInfo
      {
        h323-ID : "ptel23@zone2.com"
      },
      destCallSignalAddress ipAddress :
        {
          ip '66000001'H,
          port 1720
        },
      srcInfo
      {
        e164 : "7777",
        h323-ID : "ptel213"
      },
      bandWidth 7680,
      callReferenceValue 67,
      conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
      activeMC FALSE,
      answerCall FALSE
    }
value RasMessage ::= admissionConfirm :
    {
      requestSeqNum 31,
      bandWidth 7680,
      callModel direct : NULL,
      destCallSignalAddress ipAddress :
        {
          ip '66000001'H,
          port 1720
        },
      irrFrequency 30
    }
```

Example 2

The following output shows two proxy call scenarios. A trace is collected on the source proxy with ASN1 turned on. The call is being torn down

```
Router# debug h225 asn1

H.225 ASN1 Messages debugging is on
Router#
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body setup :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        sourceAddress
        {
          h323-ID : "ptel213"
        },
        sourceInfo
        {
          terminal
          {
          },
          mc FALSE,
          undefinedNode FALSE
        },
        destinationAddress
        {
          h323-ID : "ptel23@zone2.com"
        },
        activeMC FALSE,
        conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
        conferenceGoal create : NULL,
        callType pointToPoint : NULL,
        sourceCallSignalAddress ipAddress :
          {
            ip '3200000C'H,
            port 1720
          }
      }
  }
}
value RasMessage ::= admissionRequest :
  {
    requestSeqNum 30,
    callType pointToPoint : NULL,
    endpointIdentifier "0068A96000000002",
    destinationInfo
    {
      h323-ID : "ptel23@zone2.com"
    },
    srcInfo
    {
      h323-ID : "ptel213"
    },
    bandWidth 640,
    callReferenceValue 1,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    activeMC FALSE,
    answerCall TRUE
  }
2480001D 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
```

```
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D014006 00700074 0065006C 00320031 00334002 8000015F
C8490FB4 B9D111BF AF0060B0 00E94540
2980001D 401E0000 66000001 06B8001D 40B50000 1247C002 03AAAA80 06007000
74006500 6C003200 31003301 800F0070 00740065 006C0032 00330040 007A006F
006E0065 0032002E 0063006F 006D0180 02007000 78003266 00000106 B8
value RasMessage ::= admissionConfirm :
  {
    requestSeqNum 30,
    bandWidth 7680,
    callModel direct : NULL,
    destCallSignalAddress ipAddress :
      {
        ip '66000001'H,
        port 1720
      },
    irrFrequency 30,
    nonStandardData
    {
      nonStandardIdentifier h221NonStandard :
        {
          t35CountryCode 181,
          t35Extension 0,
          manufacturerCode 18
        },
      data
'C00203AAAA8006007000740065006C0032003100330180F007000740065006C003200 ...'H
    }
  }
C00203AA AA800600 70007400 65006C00 32003100 3301800F 00700074 0065006C
00320033 0040007A 006F006E 00650032 002E0063 006F006D 01800200 70007800
32660000 0106B8
value ACFnonStandardInfo ::=
{
  srcTerminalAlias
  {
    e164 : "7777",
    h323-ID : "ptel213"
  },
  dstTerminalAlias
  {
    h323-ID : "ptel23@zone2.com"
  },
  dstProxyAlias
  {
    h323-ID : "px2"
  },
  dstProxySignalAddress
  {
    ip '66000001'H,
    port 1720
  }
}
value RasMessage ::= admissionRequest :
  {
    requestSeqNum 31,
    callType pointToPoint : NULL,
    endpointIdentifier "0068A96000000002",
    destinationInfo
    {
      h323-ID : "ptel23@zone2.com"
    },
    destCallSignalAddress ipAddress :
      {
```

```
        ip '66000001'H,
        port 1720
      },
    srcInfo
    {
      e164 : "7777",
      h323-ID : "ptel213"
    },
    bandWidth 7680,
    callReferenceValue 67,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    activeMC FALSE,
    answerCall FALSE
  }
24C0001E 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D006600 000106B8 020180AA AA400600 70007400 65006C00
32003100 33401E00 00435FC8 490FB4B9 D111BFAF 0060B000 E94500
2900001E 401E0000 66000001 06B8001D
value RasMessage ::= admissionConfirm :
  {
    requestSeqNum 31,
    bandWidth 7680,
    callModel direct : NULL,
    destCallSignalAddress ipAddress :
      {
        ip '66000001'H,
        port 1720
      },
    irrFrequency 30
  }
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body callProceeding :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        destinationInfo
        {
          gateway
          {
            protocol
            {
              h323 :
                {
                }
            }
          },
          mc FALSE,
          undefinedNode FALSE
        }
      }
  }
}
01000600 08914A00 01088001 2800
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body setup :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        sourceAddress
```

```
          {
            h323-ID : "ptel213"
          },
          sourceInfo
          {
            vendor
            {
              vendor
              {
                t35CountryCode 181,
                t35Extension 0,
                manufacturerCode 18
              }
            },
            gateway
            {
              protocol
              {
                h323 :
                  {
                  }
              }
            },
            mc FALSE,
            undefinedNode FALSE
          },
          destinationAddress
          {
            h323-ID : "ptel23@zone2.com"
          },
          destCallSignalAddress ipAddress :
            {
              ip '66000001'H,
              port 1720
            },
          activeMC FALSE,
          conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
          conferenceGoal create : NULL,
          callType pointToPoint : NULL,
          sourceCallSignalAddress ipAddress :
            {
              ip '65000001'H,
              port 1720
            },
          remoteExtensionAddress h323-ID : "ptel23@zone2.com"
        }
    }
}
00B80600 08914A00 01014006 00700074 0065006C 00320031 00332800 B5000012
40012800 01400F00 70007400 65006C00 32003300 40007A00 6F006E00 65003200
2E006300 6F006D00 66000001 06B8005F C8490FB4 B9D111BF AF0060B0 00E94500
0E070065 00000106 B822400F 00700074 0065006C 00320033 0040007A 006F006E
00650032 002E0063 006F006D
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body callProceeding :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        destinationInfo
        {
          gateway
          {
```

```
            protocol
            {
              h323 :
                {
                }
            }
          },
          mc FALSE,
          undefinedNode FALSE
        }
      }
  }
}
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body alerting :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        destinationInfo
        {
          mc FALSE,
          undefinedNode FALSE
        }
      }
  }
}
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body alerting :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        destinationInfo
        {
          mc FALSE,
          undefinedNode FALSE
        }
      }
  }
}
03000600 08914A00 010000
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body connect :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        h245Address ipAddress :
          {
            ip '66000001'H,
            port 11011
          },
        destinationInfo
        {
          gateway
          {
            protocol
            {
              h323 :
                {
```

```
                }
              }
            },
            mc FALSE,
            undefinedNode FALSE
          },
          conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H
        }
    }
}
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body connect :
      {
        protocolIdentifier { 0 0 8 2250 0 1 },
        h245Address ipAddress :
          {
            ip '65000001'H,
            port 11007
          },
        destinationInfo
        {
          gateway
          {
            protocol
            {
              h323 :
                {
                }
            }
          },
          mc FALSE,
          undefinedNode FALSE
        },
        conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H
      }
  }
}
02400600 08914A00 01006500 00012AFF 08800128 005FC849 0FB4B9D1 11BFAF00
60B000E9 45
```

Example 3

The following output shows two proxy call scenarios. A trace is collected on a destination router where both destination proxy and destination Gatekeeper coexist. Both RAS and H.225 traces are enabled for one complete call.

```
px2#
      RASLib::RASRecvData: successfully rcvd message of length 80 from 40.0.0.33:1585
      RASLib::RASRecvData: LRQ rcvd from [40.0.0.33:1585] on sock [6880372]
      RASlib::ras_sendto: msg length 111 sent to 40.0.0.33
      RASLib::RASSendLCF: LCF sent to 40.0.0.33
      H225Lib::h225TAccept: TCP connection accepted from 101.0.0.1:11002 on
socket [2]
      H225Lib::h225TAccept: Q.931 Call State is initialized to be [Null].
Hex representation of the received TPKT
030000A608020080050404889 88CA56C0591373737377E008D0500B8060008914A000101400
6007000740065006C0032003100332800B50000124001280001400F007000740065006C00320
0330040007A006F006E00650032002E0063006F006D006600000106B8003DC8490FB4B9D111B
FAF0060B000E945000E07006500000106B822400F007000740065006C003200330040007A006
F006E00650032002E0063006F006D
```

```
      H225Lib::h225RecvData: Q.931 SETUP received from socket [2]
      H225Lib::h225RecvData: State changed to [Call Present].
      RASlib::ras_sendto: msg length 119 sent to 102.0.0.1
      RASLib::RASSendARQ: ARQ sent to 102.0.0.1
      RASLib::RASRecvData: successfully rcvd message of length 119 from 102.0.0.1:24999
      RASLib::RASRecvData: ARQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
      RASlib::ras_sendto: msg length 16 sent to 70.0.0.31
      RASLib::RASSendACF: ACF sent to 70.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 16 from 102.0.0.1:1719
      RASLib::RASRecvData: ACF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
      RASlib::ras_sendto: msg length 119 sent to 102.0.0.1
      RASLib::RASSendARQ: ARQ sent to 102.0.0.1
      RASLib::RASRecvData: successfully rcvd message of length 119 from 102.0.0.1:24999
      RASLib::RASRecvData: ARQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
      RASlib::ras_sendto: msg length 16 sent to 70.0.0.31
      RASLib::RASSendACF: ACF sent to 70.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 16 from 102.0.0.1:1719
      RASLib::RASRecvData: ACF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
Hex representation of the CALL PROCEEDING TPKT to send.
0300001B08028080027E000F050100060008914A00010880012800
      H225Lib::h225CallProcRequest: Q.931 CALL PROCEEDING sent from socket
[2]. Call state remains unchanged (Q.931 FSM simplified for H.225.0)
      H225Lib::h225TConn: connect in progress on socket [4]
      H225Lib::h225TConn: Q.931 Call State is initialized to be [Null].
Hex representation of the SETUP TPKT to send.
030000A50802008005040388C0A56C0591373737377E008D0500B8060008914A00010140060
07000740065006C0032003100332800B500001240012800001400F007000740065006C0032003
30040007A006F006E00650032002E0063006F006D005A00000D06B8003DC8490FB4B9D111BFA
F0060B000E945000E07006600000106B822400F007000740065006C003200330040007A006F0
06E00650032002E0063006F006D
      H225Lib::h225SetupRequest: Q.931 SETUP sent from socket [4]
      H225Lib::h225SetupRequest: Q.931 Call State changed to [Call Initiated].
      RASLib::RASRecvData: successfully rcvd message of length 123 from 90.0.0.13:1700
      RASLib::RASRecvData: ARQ rcvd from [90.0.0.13:1700] on sock [0x68FC74]
      RASlib::ras_sendto: msg length 16 sent to 90.0.0.13
      RASLib::RASSendACF: ACF sent to 90.0.0.13
Hex representation of the received TPKT
0300001808028080027E000C050100060008914A00010200
      H225Lib::h225RecvData: Q.931 CALL PROCEEDING received from socket [4]
Hex representation of the received TPKT
0300001808028080017E000C050300060008914A00010200
      H225Lib::h225RecvData: Q.931 ALERTING received from socket [4]
      H225Lib::h225RecvData: Q.931 Call State changed to [Call Delivered].
Hex representation of the ALERTING TPKT to send.
0300001808028080017E000C050300060008914A00010000
      H225Lib::h225AlertRequest: Q.931 ALERTING sent from socket [2]. Call
state changed to [Call Received].
Hex representation of the received TPKT
030000350802808007040488988 6A57E0023050240060008914A0001005A00000D06A402003
DC8490FB4B9D111BFAF0060B000E945
      H225Lib::h225RecvData: Q.931 CONNECT received from socket [4]
      H225Lib::h225RecvData: Q.931 Call State changed to [Active].
Hex representation of the CONNECT TPKT to send.
03000037080280800704038 8C0A57E0026050240060008914A000100660000012AFC0880012
8003DC8490FB4B9D111BFAF0060B000E945
      H225Lib::h225SetupResponse: Q.931 CONNECT sent from socket [2]
      H225Lib::h225SetupResponse: Q.931 Call State changed to [Active].
      RASlib::ras_sendto: msg length 108 sent to 102.0.0.1
      RASLib::RASSendIRR: IRR sent to 102.0.0.1
      RASLib::RASRecvData: successfully rcvd message of length 108 from 102.0.0.1:24999
      RASLib::RASRecvData: IRR rcvd from [102.0.0.1:24999] on sock [0x68FC74]
      RASLib::RASRecvData: successfully rcvd message of length 101 from 90.0.0.13:1700
      RASLib::RASRecvData: IRR rcvd from [90.0.0.13:1700] on sock [0x68FC74]
Hex representation of the received TPKT
```

```
0300001A080280805A080280107E000A050500060008914A0001
      H225Lib::h225RecvData: Q.931 RELEASE COMPLETE received from socket [2]
      H225Lib::h225RecvData: Q.931 Call State changed to [Null].
      RASlib::ras_sendto: msg length 55 sent to 102.0.0.1
      RASLib::RASSendDRQ: DRQ sent to 102.0.0.1
      H225Lib::h225RecvData: no connection on socket [2]
      RASLib::RASRecvData: successfully rcvd message of length 55 from 102.0.0.1:24999
      RASLib::RASRecvData: DRQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
      RASlib::ras_sendto: msg length 3 sent to 70.0.0.31
      RASLib::RASSendDCF: DCF sent to 70.0.0.31
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280805A080280107E000A050500060008914A0001
      H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [2]. Call
state changed to [Null].
      H225Lib::h225TClose: TCP connection from socket [2] closed
      RASlib::ras_sendto: msg length 55 sent to 102.0.0.1
      RASLib::RASSendDRQ: DRQ sent to 102.0.0.1
      RASLib::RASRecvData: successfully rcvd message of length 3 from 102.0.0.1:1719
      RASLib::RASRecvData: DCF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
      RASLib::RASRecvData: successfully rcvd message of length 55 from 102.0.0.1:24999
      RASLib::RASRecvData: DRQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
      RASlib::ras_sendto: msg length 3 sent to 70.0.0.31
      RASLib::RASSendDCF: DCF sent to 70.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 3 from 102.0.0.1:1719
      RASLib::RASRecvData: DCF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280805A080280107E000A050500060008914A0001
      H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [4]. Call
state changed to [Null].
      H225Lib::h225TClose: TCP connection from socket [4] closed
      RASLib::RASRecvData: successfully rcvd message of length 55 from 90.0.0.13:1700
      RASLib::RASRecvData: DRQ rcvd from [90.0.0.13:1700] on sock [0x68FC74]
      RASlib::ras_sendto: msg length 3 sent to 90.0.0.13
      RASLib::RASSendDCF: DCF sent to 90.0.0.13
```

debug h225 events

To display Q.931 events, use the **debug h225 events** privileged EXEC command. The **no** form of this command disables debugging output.

debug h225 events

no debug h255 events

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)NA	This command was introduced.
12.0(3)T	This command was modified.

Examples

The following are sample output from the **debug h225 events** command.

Example 1

The following output shows two proxy call scenarios. A trace is collected on the source proxy with H.225 turned on. The call is being established.

```
Router# debug h225 events
H.225 Event Messages debugging is on
Router#  H225Lib::h225TAccept: TCP connection accepted from 50.0.0.12:1701 on
socket [2]
      H225Lib::h225TAccept: Q.931 Call State is initialized to be [Null].
Hex representation of the received TPKT
03000074080200010504048898 86A56C0580373737377E005B0500B0060008914A000101400
6007000740065006C00320031003302000140 0F0070007400 65006C003200330040007A006F0
06E00650032002E0063006F006D004EC8490FB4B9D111BFAF0060B000E945000C07003200000
C06B8
      H225Lib::h225RecvData: Q.931 SETUP received from socket [2]
      H225Lib::h225RecvData: State changed to [Call Present].
Hex representation of the CALL PROCEEDING TPKT to send.
0300001B08028001027E000F050100060008914A00010880012800
      H225Lib::h225CallProcRequest: Q.931 CALL PROCEEDING sent from socket
[2]. Call state remains unchanged (Q.931 FSM simplified for H.225.0)
      H225Lib::h225TConn: connect in progress on socket [4]
      H225Lib::h225TConn: Q.931 Call State is initialized to be [Null].
Hex representation of the SETUP TPKT to send.
030000A60802008405040488988CA56C0591373737377E008D0500B8060008914A000101400
6007000740065006C0032003100332800B50000124001280001400F007000740065006C00320
0330040007A006F006E00650032002E0063006F006D006600000106B8004EC8490FB4B9D111B
FAF0060B000E945000E07006500000106B822400F007000740065006C003200330040007A006
F006E00650032002E0063006F006D
      H225Lib::h225SetupRequest: Q.931 SETUP sent from socket [4]
      H225Lib::h225SetupRequest: Q.931 Call State changed to [Call Initiated].
Hex representation of the received TPKT
0300001B08028084027E000F050100060008914A00010880012800
      H225Lib::h225RecvData: Q.931 CALL PROCEEDING received from socket [4]
Hex representation of the received TPKT
03000018080280840 17E000C050300060008914A00010000
      H225Lib::h225RecvData: Q.931 ALERTING received from socket [4]
```

```
      H225Lib::h225RecvData: Q.931 Call State changed to [Call Delivered].
Hex representation of the ALERTING TPKT to send.
0300001808028001017E000C050300060008914A00010000
      H225Lib::h225AlertRequest: Q.931 ALERTING sent from socket [2]. Call
state changed to [Call Received].
Hex representation of the received TPKT
03000037080280840704038 8C0A57E002605024006000891 4A000100660000012AFF0880012
8004EC8490FB4B9D111BFAF0060B000E945
      H225Lib::h225RecvData: Q.931 CONNECT received from socket [4]
      H225Lib::h225RecvData: Q.931 Call State changed to [Active].
Hex representation of the CONNECT TPKT to send.
030000380802800107040488 9886A57E002605024006000891 4A000100650000012AFC08800
128004EC8490FB4B9D111BFAF0060B000E945
      H225Lib::h225SetupResponse: Q.931 CONNECT sent from socket [2]
      H225Lib::h225SetupResponse: Q.931 Call State changed to [Active].
```

Example 2

The following output shows two proxy call scenarios. A trace is collected on the source proxy with H.225 turned on. The call is being torn down.

```
Router# debug h225 events

H.225 Event Messages debugging is on
Router#
Hex representation of the received TPKT
0300001A080200015A080200907E000A050500060008914A0001
      H225Lib::h225RecvData: Q.931 RELEASE COMPLETE received from socket [2]
      H225Lib::h225RecvData: Q.931 Call State changed to [Null].
      H225Lib::h225RecvData: no connection on socket [2]
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280015A080280107E000A050500060008914A0001
      H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [2]. Call
state changed to [Null].
      H225Lib::h225TClose: TCP connection from socket [2] closed
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280845A080280107E000A050500060008914A0001
      H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [4]. Call
state changed to [Null].
      H225Lib::h225TClose: TCP connection from socket [4] closed
```

debug h245 asn1

To display ASN1 contents of H.245 messages, use the **debug h245 asn1** privileged EXEC command. The **no** form of this command disables debugging output.

debug h245 asn1

no debug h245 asn1

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)NA	This command was introduced.
12.0(3)T	This command was modified.

Usage Guidelines

Caution This command slows the system down considerably. Connections may time out.

debug h245 events

To display H.245 events, use the **debug h245 events** privileged EXEC command. The **no** form of this command disables debugging output.

debug h245 events

no debug h245 events

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)NA	This command was introduced.
12.0(3)T	This command was modified.

debug ima

To display debug messages for IMA groups and links, enter the **debug ima** privileged EXEC command. Enter the **no** form of this command to disable debugging output.

debug ima

no debug ima

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for IMA groups is not enabled.

Command History

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

Examples

The following example shows output when you enter the **debug ima** command while adding two ATM links to an IMA group. Notice that the group has not yet been created with the **interface atm** *slot*/**ima***group-number* command, so the links are not activated yet as group members. However, the individual ATM links are deactivated.

```
Router# debug ima

IMA network interface debugging is on
Router# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# interface atm1/0
Router(config-if)# ima-group 1
Router(config-if)#
01:35:08:IMA shutdown atm layer of link ATM1/0
01:35:08:ima_clear_atm_layer_if ATM1/0
01:35:08:IMA link ATM1/0 removed in firmware
01:35:08:ima_release_channel:ATM1/0 released channel 0.
01:35:08:Bring up ATM1/4 that had been waiting for a free channel.
01:35:08:IMA:no shut the ATM interface.
01:35:08:IMA allocate_channel:ATM1/4 using channel 0.
01:35:08:IMA config_restart ATM1/4
01:35:08:IMA adding link 0 to Group ATM1/IMA1ATM1/0 is down waiting for IMA group 1 to be
activated
01:35:08:Link 0 was added to Group ATM1/IMA1
01:35:08:ATM1/0 is down waiting for IMA group 1 to be created.
01:35:08:IMA send AIS on link ATM1/0
01:35:08:IMA Link up/down Alarm:port 0, new status 0x10, old_status 0x1.
01:35:10:%LINK-3-UPDOWN:Interface ATM1/4, changed state to up
01:35:10:%LINK-3-UPDOWN:Interface ATM1/0, changed state to down
01:35:11:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/4, changed state to up
01:35:11:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/0, changed state to down
Router(config-if)# int atm1/1
Router(config-if)# ima-group 1
Router(config-if)#
01:37:19:IMA shutdown atm layer of link ATM1/1
```

```
01:37:19:ima_clear_atm_layer_if ATM1/1
01:37:19:IMA link ATM1/1 removed in firmware
01:37:19:ima_release_channel:ATM1/1 released channel 1.
01:37:19:Bring up ATM1/5 that had been waiting for a free channel.
01:37:19:IMA:no shut the ATM interface.
01:37:19:IMA allocate_channel:ATM1/5 using channel 1.
01:37:19:IMA config_restart ATM1/5
01:37:19:IMA adding link 1 to Group ATM1/IMA1ATM1/1 is down waiting for IMA group 1 to be
activated
01:37:19:Link 1 was added to Group ATM1/IMA1
01:37:19:ATM1/1 is down waiting for IMA group 1 to be created.
01:37:19:IMA send AIS on link ATM1/1
01:37:19:IMA Link up/down Alarm:port 1, new status 0x10, old_status 0x1.
Router(config-if)#
01:37:21:%LINK-3-UPDOWN:Interface ATM1/5, changed state to up
01:37:21:%LINK-3-UPDOWN:Interface ATM1/1, changed state to down
01:37:22:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/5, changed state to up
01:37:22:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1, changed state to down
```

Related Commands

Command	Description
debug backhaul-session-manager set	Displays debug messages for ATM errors, and reports specific problems such as encapsulation errors and errors related to OAM cells.
debug events	Displays debug messages for ATM events, and reports specific events such as PVC setup completion, changes in carrier states, and interface rates.

debug ip auth-proxy

To display the authentication proxy configuration information on the router, use the **debug ip auth-proxy** command in privileged EXEC mode.

debug ip auth-proxy {**ftp** | **function-trace** | **http** | **object-creation** | **object-deletion** | **tcp** | **telnet** | **timer**}

Syntax Description

ftp	Displays FTP events related to the authentication proxy.
function-trace	Displays the authentication proxy functions.
http	Displays HTTP events related to the authentication proxy.
object-creation	Displays additional entries to the authentication proxy cache.
object-deletion	Displays deletion of cache entries for the authentication proxy.
tcp	Displays TCP events related to the authentication proxy.
telnet	Displays Telnet-related authentication proxy events.
timer	Displays authentication proxy timer-related events.

Command History

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

Usage Guidelines

Use the **debug ip auth-proxy** command to display authentication proxy activity. See the "Examples" section for more information about the debug options.

Note The **function-trace** debugging information provides low-level software information for Cisco technical support representatives. No output examples are provided for this keyword option.

Examples

The following examples illustrates the output of the **debug ip auth-proxy** command. In these examples, debugging is on for object creations, object deletions, HTTP, and TCP.

In this example, the client host at 192.168.201.1 is attempting to make an HTTP connection to the web server located at 192.168.21.1. The HTTP debugging information is on for the authentication proxy. The output shows that the router is setting up an authentication proxy entry for the login request:

```
00:11:10: AUTH-PROXY creates info:
    cliaddr - 192.168.21.1, cliport - 36583
    seraddr - 192.168.201.1, serport - 80
        ip-srcaddr 192.168.21.1
        pak-srcaddr 0.0.0.0
```

Following a successful login attempt, the debugging information shows the authentication proxy entries created for the client. In this example, the client is authorized for SMTP (port 25), FTP data (port 20), FTP control (port 21), and Telnet (port 23) traffic. The dynamic ACL entries are included in the display.

```
00:11:25:AUTH_PROXY OBJ_CREATE:acl item 61AD60CC

00:11:25:AUTH-PROXY OBJ_CREATE:create acl wrapper 6151C7C8 -- acl item 61AD60CC
00:11:25:AUTH-PROXY Src 192.168.162.216 Port [0]
00:11:25:AUTH-PROXY Dst 192.168.162.220 Port [25]
```

```
00:11:25:AUTH_PROXY OBJ_CREATE:acl item 6151C908

00:11:25:AUTH-PROXY OBJ_CREATE:create acl wrapper 6187A060 -- acl item 6151C908
00:11:25:AUTH-PROXY Src 192.168.162.216 Port [0]
00:11:25:AUTH-PROXY Dst 192.168.162.220 Port [20]
00:11:25:AUTH_PROXY OBJ_CREATE:acl item 61A40B88

00:11:25:AUTH-PROXY OBJ_CREATE:create acl wrapper 6187A0D4 -- acl item 61A40B88
00:11:25:AUTH-PROXY Src 192.168.162.216 Port [0]
00:11:25:AUTH-PROXY Dst 192.168.162.220 Port [21]
00:11:25:AUTH_PROXY OBJ_CREATE:acl item 61879550

00:11:25:AUTH-PROXY OBJ_CREATE:create acl wrapper 61879644 -- acl item 61879550
00:11:25:AUTH-PROXY Src 192.168.162.216 Port [0]
00:11:25:AUTH-PROXY Dst 192.168.162.220 Port [23]
```

The next example shows the debug output following a **clear ip auth-proxy cache** command to clear the authentication entries from the router. The dynamic ACL entries are removed from the router.

```
00:12:36:AUTH-PROXY OBJ_DELETE:delete auth_proxy cache 61AD6298
00:12:36:AUTH-PROXY OBJ_DELETE:delete create acl wrapper 6151C7C8 -- acl item 61AD60CC
00:12:36:AUTH-PROXY OBJ_DELETE:delete create acl wrapper 6187A060 -- acl item 6151C908
00:12:36:AUTH-PROXY OBJ_DELETE:delete create acl wrapper 6187A0D4 -- acl item 61A40B88
00:12:36:AUTH-PROXY OBJ_DELETE:delete create acl wrapper 61879644 -- acl item 61879550
```

The following example shows the timer information for a dynamic ACL entry. All times are expressed in milliseconds. The *first laststart* is the time that the ACL entry is created relative to the startup time of the router. The *lastref* is the time of the last packet to hit the dynamic ACL relative to the startup time of the router. The *exptime* is the next expected expiration time for the dynamic ACL. The *delta* indicates the remaining time before the dynamic ACL expires. After the timer expires, the debugging information includes a message indicating that the ACL and associated authentication proxy information for the client have been removed.

```
00:19:51:first laststart 1191112

00:20:51:AUTH-PROXY:delta 54220 lastref 1245332 exptime 1251112
00:21:45:AUTH-PROXY:ACL and cache are removed
```

Related Commands

Command	Description
show debug	Displays the debug options set on the router.

debug ip bgp

To display information related to processing BGPs, use the **debug ip bgp** privileged EXEC command. To disable the display of BGP information, use the **no** form of this command.

debug ip bgp [*A.B.C.D.* | **dampening** | **events** | **in** | **keepalives** | **out** | **updates** | **vpnv4**]

no debug ip bgp [*A.B.C.D.* | **dampening** | **events** | **in** | **keepalives** | **out** | **updates** | **vpnv4**]

Syntax Description

A.B.C.D.	(Optional) Displays the BGP neighbor IP address.
dampening	(Optional) Displays BGP dampening.
events	(Optional) Displays BGP events.
in	(Optional) BGP inbound information.
keepalives	(Optional) Displays BGP keepalives.
out	(Optional) Displays BGP outbound information.
updates	(Optional) Displays BGP updates.
vpnv4	(Optional) Displays VPNv4 NLRI information.

Command History

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

Examples

The following example displays the output from this command:

```
Router# debug ip bgp vpnv4

03:47:14:vpn:bgp_vpnv4_bnetinit:100:2:58.0.0.0/8
03:47:14:vpn:bnettable add:100:2:58.0.0.0 / 8
03:47:14:vpn:bestpath_hook route_tag_change for vpn2:58.0.0.0/255.0.0.0(ok)
03:47:14:vpn:bgp_vpnv4_bnetinit:100:2:57.0.0.0/8
03:47:14:vpn:bnettable add:100:2:57.0.0.0 / 8
03:47:14:vpn:bestpath_hook route_tag_change for vpn2:57.0.0.0/255.0.0.0(ok)
03:47:14:vpn:bgp_vpnv4_bnetinit:100:2:14.0.0.0/8
03:47:14:vpn:bnettable add:100:2:14.0.0.0 / 8
03:47:14:vpn:bestpath_hook route_tag_chacle ip bgp *nge for vpn2:14.0.0.0/255.0.0.0(ok)
```

debug ip casa affinities

To display debug messages for affinities, use the **debug ip casa affinities** privileged EXEC command. Use the **no** form of the command to disable debugging.

debug ip casa affinities

no debug ip casa affinities

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for affinities is not enabled.

Command History

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

Examples

The following is output from the **debug ip casa affinities** command:

```
Router# debug ip casa affinities

16:15:36:Adding fixed affinity:
16:15:36:    10.10.1.1:54787 -> 10.10.10.10:23 proto = 6
16:15:36:Updating fixed affinity:
16:15:36:    10.10.1.1:54787 -> 10.10.10.10:23 proto = 6
16:15:36:    flags = 0x2, appl addr = 10.10.3.2, interest = 0x5/0x100
16:15:36:    int ip:port = 10.10.2.2:1638, sequence delta = 0/0/0/0
16:15:36:Adding fixed affinity:
16:15:36:    10.10.10.10:23 -> 10.10.1.1:54787 proto = 6
16:15:36:Updating fixed affinity:
16:15:36:    10.10.10.10:23 -> 10.10.1.1:54787 proto = 6
16:15:36:    flags = 0x2, appl addr = 0.0.0.0, interest = 0x3/0x104
16:15:36:    int ip:port = 10.10.2.2:1638, sequence delta = 0/0/0/0
```

Table 60 describes the significant fields shown in the display.

Table 60 debug ip casa affinities Field Descriptions

Field	Description
Adding fixed affinity	Adding a fixed affinity to affinity table.
Updating fixed affinity	Modifying a fixed affinity table with information from the services manager.
flags	Bit field indicating actions to be taken on this affinity.
fwd addr	Address to which packets will be directed.
interest	Services manager that is interested in packets for this affinity.
int ip:port	Services manager port to which interest packets are sent.
sequence delta	Used to adjust TCP sequence numbers for this affinity.

debug ip casa packets

To display debug messages for packets, use the **debug ip casa packets** privileged EXEC command. Use the **no** form of the command to disable debugging.

debug ip casa packets

no debug ip casa packets

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for packets is not enabled.

Command History

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

Examples

The following is output from the **debug ip casa packets** command:

```
Router# debug ip casa packets

16:15:36:Routing CASA packet - TO_MGR:
16:15:36:    10.10.1.1:55299 -> 10.10.10.10:23 proto = 6
16:15:36:    Interest Addr:10.10.2.2   Port:1638
16:15:36:Routing CASA packet - FWD_PKT:
16:15:36:    10.10.1.1:55299 -> 10.10.10.10:23 proto = 6
16:15:36:    Fwd Addr:10.10.3.2
16:15:36:Routing CASA packet - TO_MGR:
16:15:36:    10.10.10.10:23 -> 10.10.1.1:55299 proto = 6
16:15:36:    Interest Addr:10.10.2.2   Port:1638
16:15:36:Routing CASA packet - FWD_PKT:
16:15:36:    10.10.10.10:23 -> 10.10.1.1:55299 proto = 6
16:15:36:    Fwd Addr:0.0.0.0
16:15:36:Routing CASA packet - TICKLE:
16:15:36:    10.10.10.10:23 -> 10.10.1.1:55299 proto = 6
16:15:36:    Interest Addr:10.10.2.2   Port:1638  Interest Mask:SYN
16:15:36:    Fwd Addr:0.0.0.0
16:15:36:Routing CASA packet - FWD_PKT:
16:15:36:    10.10.1.1:55299 -> 10.10.10.10:23 proto = 6
16:15:36:    Fwd Addr:10.10.3.2
```

Table 61 describes the significant fields shown in the display.

Table 61 debug ip casa packets Commands Field Descriptions

Field	Description
Routing CASA packet - TO_MGR	Forwarding Agent is routing a packet to the services manager.
Routing CASA packet - FWD_PKT	Forwarding Agent is routing a packet to the forwarding address.
Routing CASA packet - TICKLE	Forwarding Agent is signalling services manager while allowing the packet in question to take the appropriate action.
Interest Addr	Services manager address.
Interest Port	Port on the services manager where packet is sent.
Fwd Addr	Address to which packets matching the affinity are sent.
Interest Mask	Services manager that is interested in packets for this affinity.

debug ip casa wildcards

To display debug messages for wildcards, use the **debug ip casa wildcards** privileged EXEC command. Use the **no** form of this command to disable debugging.

debug ip casa wildcards

no debug ip casa wildcards

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for wildcards is not enabled.

Command History

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

Examples

The following is output from the **debug ip casa wildcards** command:

```
Router# debug ip casa wildcards

16:13:23:Updating wildcard affinity:
16:13:23:    10.10.10.10:0 -> 0.0.0.0:0 proto = 6
16:13:23:    src mask = 255.255.255.255, dest mask = 0.0.0.0
16:13:23:    no frag, not advertising
16:13:23:    flags = 0x0, appl addr = 0.0.0.0, interest = 0x8107/0x8104
16:13:23:    int ip:port = 10.10.2.2:1638, sequence delta = 0/0/0/0
16:13:23:Updating wildcard affinity:
16:13:23:    0.0.0.0:0 -> 10.10.10.10:0 proto = 6
16:13:23:    src mask = 0.0.0.0, dest mask = 255.255.255.255
16:13:23:    no frag, advertising
16:13:23:    flags = 0x0, appl addr = 0.0.0.0, interest = 0x8107/0x8102
16:13:23    int ip:port = 10.10.2.2:1638, sequence delta = 0/0/0/0
```

Table 62 describes the significant fields in the display.

.

Table 62 debug ip casa wildcards Commands Field Descriptions

Field	Description
src mask	Source of connection.
dest mask	Destination of connection.
no frag, not advertising	Not accepting IP fragments.
flags	Bit field indicating actions to be taken on this affinity.
fwd addr	Address to which packets matching the affinity will be directed.
interest	Services manager that is interested in packets for this affinity.
int ip: port	Services manager port to which interest packets are sent.
sequence delta	Used to adjust sequence numbers for this affinity.

debug ip cef

To troubleshoot various Cisco Express Forwarding (CEF) events, use the **debug ip cef** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip cef {**drops** [**rpf** [*access-list*]] [*access-list*] | **receive** [*access-list*] | **events** [*access-list*] | **interface**}

no debug ip cef {**drops** [**rpf** [*access-list*]] [*access-list*] | **receive** [*access-list*] | **events** [*access-list*] | **interface**}

Specific to IPC Records

debug ip cef {**ipc** | **interface-ipc** | **prefix-ipc** [*access-list*]}

no debug ip cef {**ipc** | **interface-ipc** | **prefix-ipc** [*access-list*]}

Syntax Description

drops	Records dropped packets.
rpf	(Optional) Records the result of the Reverse Path Forwarding check for packets.
access-list	(Optional) Limits debugging collection to packets that match the list.
receive	Records packets that are ultimately destined to the router, as well as packets destined to a tunnel endpoint on the router. If the decapsulated tunnel is IP, it is CEF switched; otherwise packets are process switched.
events	Records general CEF events.
interface	Records IP CEF interface events.
ipc	Records information related to Interprocess communications (IPC) in CEF. Possible types of events include the following: • Transmission status of IPC messages • Status of buffer space for IPC messages • IPC messages received out of sequence • Status of resequenced messages • Throttle requests sent from a line card to the Route Processor
interface-ipc	Records IPC updates related to interfaces. Possible reporting includes an interface coming up or going down, and updates to fibhwidb, fibidb, and so on.
prefix-ipc	Records updates related to IP prefix information. Possible updates include the following: • Debugging of IP routing updates in a line card • Reloading of a line card with a new table • Updates related to exceeding the maximum number of routes • Control messages related to forwarding information base (FIB) table prefixes

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2 GS	This command was introduced.
11.1 CC	Multiple platform support was added.
12.0(5)T	The **rpf** keyword was added.

Usage Guidelines

This command gathers additional information for the handling of CEF interface, IPC, or packet events.

Note For packet events, we recommend that you use an Access Control List (ACL) to limit the messages recorded.

Examples

The following is sample output from the **debug ip cef rpf** command for a packet that is dropped when it fails the RPF check. IP address 172.17.249.252 is the source address and Ethernet 2/0/0 is the input interface:

```
Router# debug ip cef drops rpf

IP CEF drops for RPF debugging is on
00:42:02:CEF-Drop:Packet from 172.17.249.252 via Ethernet2/0/0 -- unicast rpf check
```

The following is sample output for CEF packets that are not switched using information from the FIB table, but are received and sent to the next switching layer:

```
Router# debug ip cef receive

IP CEF received packets debugging is on
00:47:52:CEF-receive:Receive packet for 9.1.104.13
```

Table 63 describes the significant fields shown in the display.

Table 63 debug ip cef Field Descriptions

Field	Description
CEF-Drop:Packet from 172.17.249.252 via Ethernet2/0/0 -- unicast rpf check	A packet from IP address 172.17.249.252 is dropped because it failed the reverse path forwarding check.
CEF-receive:Receive packet for 9.1.104.13	CEF has received a packet addressed to the router.

debug ip cef accounting non-recursive

To troubleshoot Cisco Express Forwarding (CEF) accounting records, use the **debug ip cef accounting non-recursive** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip cef accounting non-recursive

no debug ip cef acounting non-recursive

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1 CC	This command was introduced.

Usage Guidelines

This command records accounting events for nonrecursive prefixes when the **ip cef accounting non-recursive** command is enabled in global configuration mode.

Examples

The following is sample output from the **debug ip cef accounting non-recursive** command.

```
Router# debug ip cef accounting non-recursive

03:50:19:CEF-Acct:tmstats_binary:Beginning generation of tmstats
ephemeral file (mode binary)
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF2000
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1EA0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF17C0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1D40
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1A80
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0740
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF08A0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0B60
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0CC0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0F80
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF10E0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1240
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF13A0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1500
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1920
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0E20
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1660
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF05E0
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0A00
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF1BE0
```

```
03:50:19:CEF-Acct:snapshoting loadinfo 0x63FF0480
03:50:19:CEF-Acct:tmstats_binary:aggregation complete, duration 0 seconds
03:50:21:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:24:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:24:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:27:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:29:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:32:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:35:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:38:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:41:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:45:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:48:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:49:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:52:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:55:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:writing 45 bytes
03:50:57:CEF-Acct:tmstats_binary:tmstats file written, status 0
```

Table 64 describes the significant fields shown in the display.

Table 64 debug ip cef accounting non-recursive Field Descriptions

Field	Description
Beginning generation of tmstats ephemeral file (mode binary)	Tmstats file is being created.
CEF-Acct:snapshoting loadinfo 0x63FF2000	Baseline counters are being written to the tmstats file for each nonrecursive prefix.
CEF-Acct:tmstats_binary:aggregation complete, duration 0 seconds	Tmstats file creation is complete.
CEF-Acct:tmstats_binary:writing 45 bytes	Nonrecursive accounting statistics are being updated to the tmstats file.
CEF-Acct:tmstats_binary:tmstats file written, status 0	Update of the tmstats file is complete.

debug ip cef fragmentation

To report fragmented IP packets when Cisco Express Forwarding (CEF) is enabled, use the **debug ip cef fragmentation** command in privileged EXEC mode. To disable debugging, use the **no** form of this command:

debug ip cef fragmentation

no debug ip cef fragmentation

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(14)S	This command was introduced.
12.2(2)T	This command was integrated into Cisco IOS Release 12.2(2)T.

Usage Guidelines

This command is used to troubleshoot fragmentation problems when CEF switching is enabled.

Examples

The following is sample output from the **debug ip cef fragmentation** command:

```
Router# debug ip cef fragmentation

00:59:45:CEF-FRAG:no_fixup path:network_start 0x5397CF8E datagramstart 0x5397CF80
data_start 0x397CF80 data_block 0x397CF40 mtu 1000 datagramsize 1414 data_bytes 1414
00:59:45:CEF-FRAG:send frag:datagramstart 0x397CF80 datagramsize 442 data_bytes 442
00:59:45:CEF-FRAG:send frag:datagramstart 0x38BC266 datagramsize 1006 data_bytes 1006
00:59:45:CEF-FRAG:no_fixup path:network_start 0x5397C60E datagramstart 0x5397C600
data_start 0x397C600 data_block 0x397C5C0 mtu 1000 datagramsize 1414 data_bytes 1414
00:59:45:CEF-FRAG:send frag:datagramstart 0x397C600 datagramsize 442 data_bytes 442
00:59:45:CEF-FRAG:send frag:datagramstart 0x38BC266 datagramsize 1006 data_bytes 1006
```

Table 65 describes the significant fields shown in the display.

Table 65 debug ip cef fragmentation Field Descriptions

Field	Description
no_fixup path	A packet is being fragmented in the no_fixup path.
network_start 0x5397CF8E	Memory address of the IP packet.
datagramstart 0x5397CF80	Memory address of the encapsulated IP packet.

Table 65 debug ip cef fragmentation Field Descriptions

Field	Description
data_start 0x397CF80	For particle systems, the memory address where data starts for the first packet particle.
data_block 0x397C5C0	For particle systems, the memory address of the first packet particle data block.
mtu 1000	Maximum transmission unit of the output interface.
datagramsize 1414	Size of the encapsulated IP packet.
data_bytes 1414	For particle systems, the sum of the particle data bytes that make up the packet.
send frag	Fragment is being forwarded.

debug ip cef hash

To record Cisco Express Forwarding (CEF) load sharing hash algorithm events, use the **debug ip cef hash** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip cef hash

no debug ip cef hash

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(12)S	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

Use this command when changing the load sharing algorithm to view the hash table details.

Examples

The following is sample output from the **debug ip cef hash** command with IP CEF load algorithm tunnel information:

```
Router# debug ip cef hash

01:15:06:%CEF:ip cef load-sharing algorithm tunnel 0
01:15:06:%CEF:Load balancing algorithm:tunnel
01:15:06:%CEF:Load balancing unique id:1F2BA5F6
01:15:06:%CEF:Destroyed load sharing hash table
01:15:06:%CEF:Sending hash algorithm id 2, unique id 1F2BA5F6 to slot 255
```

The following lines showIP CEF load algorithm universal information:

```
01:15:28:%CEF:ip cef load-sharing algorithm universal 0
01:15:28:%CEF:Load balancing algorithm:universal
01:15:28:%CEF:Load balancing unique id:062063A4
01:15:28:%CEF:Creating load sharing hash table
01:15:28:%CEF:Hash table columns for valid max_index:
01:15:28:12: 9  7  7 4  4 10  0  7 10  4  5  0  4  7  8  4
01:15:28:15: 3 10 10  4 10  4  0  7  1  7 14  6 13 13 11 13
01:15:28:16: 1  3  7 12  4 14  8  7 10  4  1 12  8 15  4  8
01:15:28:%CEF:Sending hash algorithm id 3, unique id 062063A4 to slot 255
```

Table 66 describes the significant fields shown in the display.

Table 66 debug ip cef hash Field Descriptions

Field	Description
ip cef load-sharing algorithm tunnel 0	Echo of the user command.
Load balancing algorithm:tunnel	Load sharing algorithm is set to tunnel.
Load balancing unique id:1F2BA5F6	ID field in the command is usually 0. In this instance, the router chose a pseudo-random ID of 1F2BA5F6.
Destroyed load sharing hash table	Purge the existing hash table.
Sending hash algorithm id 2, unique id 1F2BA5F6 to slot 255	Algorithm is being distributed.
Creating load sharing hash table	Hash table is being created.
Hash table columns for valid max_index:	Generated hash table.

debug ip cef rrhash

To record Cisco Express Forwarding (CEF) removal of receive hash events, use the **debug ip cef rrhash** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip cef rrhash

no debug ip cef rrhash

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.

Usage Guidelines

Use this command to verify the removal of receive hash events when you are shutting down or deleting an interface.

Examples

The following is sample output from the **debug ip cef rrhash** command.

```
Router# debug ip cef rrhash

00:27:15:CEF:rrhash/check:found 9.1.104.7 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.0 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.255 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.7 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.7 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.0 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.255 on down idb [ok to delete]
00:27:15:CEF:rrhash/check:found 9.1.104.7 on down idb [ok to delete]
```

Table 67 describes the significant fields shown in the display.

Table 67 debug ip cef rrhash Field Descriptions

Field	Description
rrhash/check	Verify address is on the receive list.
found 9.1.104.7 on down idb [ok to delete]	Found a valid address on the receive list for a shutdown interface which is okay to delete.

debug ip cef subblock

To troubleshoot Cisco Express Forwarding (CEF) subblock events, use the **debug ip cef subblock** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip cef subblock [**id** {**all** | **hw** *hw-id* | **sw** *sw-id* }] [**xdr** {**all** | **control** | **event** | **none** | **statistic**}]

no debug ip cef subblock

Syntax Description

id	(Optional) Subblock types.
all	(Optional) All subblock types.
hw *hw-id*	(Optional) Hardware subblock and identifier.
sw *sw-id*	(Optional) Software subblock and identifier.
xdr	(Optional) XDR message types.
control	(Optional) All XDR message types.
event	(Optional) Event XDR messages only.
none	(Optional) No XDR messages.
statistic	(Optional) Statistic XDR messages.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0 S	This command was introduced.
12.2(2)T	This command was integrated into Cisco IOS Release 12.2(2)T.

Usage Guidelines

This command is used to record CEF subblock messages and events.

Examples

The following is sample output from the **debug ip cef subblock** command:

```
Router# debug ip cef subblock

00:28:12:CEF-SB:Creating unicast RPF subblock for FastEthernet6/0
00:28:12:CEF-SB:Linked unicast RPF subblock to FastEthernet6/0.
00:28:12:CEF-SB:Encoded unit of unicast RPF data (length 16) for FastEthernet6/0
00:28:12:CEF-SB:Sent 1 data unit to slot 6 in 1 XDR message
```

Table 68 describes the significant fields shown in the display.

Table 68 debug ip cef subblock Field Descriptions

Field	Description
Creating unicast RPF subblock for FastEthernet6/0	Creating an RPF interface descriptor subblock.
Linked unicast RPF subblock to FastEthernet6/0	Linked the subblock to the specified interface.
Encoded unit of unicast RPF data (length 16) for FastEthernet6/0	Encoded the subblock information in an XDR.
Sent 1 data unit to slot 6 in 1 XDR message	Sent the XDR message to a line card through the IPC.

debug ip cef table

To enable the collection of events that affect entries in the Cisco Express Forwarding (CEF) tables, use the **debug ip cef table** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip cef table [*access-list* | **consistency-checkers**]

no debug ip cef table [*access-list* | **consistency-checkers**]

Syntax Description

access-list	(Optional) Controls collection of consistency checker parameters from specified lists.
consistency-checkers	(Optional) Sets consistency checking characteristics.

Defaults

This command is disabled by default.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2 GS	This command was introduced.
11.1 CC	Multiple platform support was added.
12.0(15)S	The **consistency-checkers** keyword was added.
12.2(2)T	This command was integrated into Cisco IOS Release 12.2(2)T.

Usage Guidelines

This command is used to record CEF table events related to the forwarding information base (FIB) table. Possible types of events include the following:

- Routing updates that populate the FIB table
- Flushing of the FIB table
- Adding or removing of entries to the FIB table
- Table reloading process

Examples

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

```
Router# debug ip cef table

01:25:46:CEF-Table:Event up, 1.1.1.1/32 (rdbs:1, flags:1000000)
01:25:46:CEF-IP:Checking dependencies of 0.0.0.0/0
01:25:47:CEF-Table:attempting to resolve 1.1.1.1/32
01:25:47:CEF-IP:resolved 1.1.1.1/32 via 9.1.104.1 to 9.1.104.1 Ethernet2/0/0
01:26:02:CEF-Table:Event up, default, 0.0.0.0/0 (rdbs:1, flags:400001)
01:26:02:CEF-IP:Prefix exists - no-op change
```

Table 69 describes the significant fields shown in the display.

Table 69 debug ip cef table Field Descriptions

Field	Description
CEF-Table	Indicates a table event.
Event up, 1.1.1.1/32	IP prefix 1.1.1.1/32 is being added.
rdbs:1	Event is from routing descriptor block 1.
flags:1000000	Indicates the network descriptor block flags.
CEF-IP	Indicates a CEF IP event.
Checking dependencies of 0.0.0.0/0	Resolves the next hop dependencies for 0.0.0.0/0.
attempting to resolve 1.1.1.1/32	Resolves the next hop dependencies.
resolved 1.1.1.1/32 via 9.1.104.1 to 9.1.104.1 Ethernet2/0/0	Next hop to IP prefix 1.1.1.1/32 is set and is added to the table.
Event up, default, 0.0.0.0/0 Prefix exists - no-op change	Indicates no table change is necessary for 0.0.0.0/32.

debug ip dhcp server

To enable DHCP Server debugging, use the **debug ip dhcp server** privileged EXEC command.

debug ip dhcp server {*events* | *packets* | *linkage*}

Syntax Description

events	Reports server events, like address assignments and database updates.
packets	Decodes DHCP receptions and transmissions.
linkage	Displays database linkage information (such as parent-child relationships in a radix tree).

Defaults

Disabled by default

Command History

Release	Modification
12.0(1)T	This command was introduced.

debug ip drp

To display Director Response Protocol (DRP) information, use the **debug ip drp** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip drp

no debug ip drp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug ip drp** command is used to debug the director response agent used by the Distributed Director product. The Distributed Director can be used to dynamically respond to Domain Name System (DNS) queries with the IP address of the "best" host based on various criteria.

Examples

The following is sample output from the **debug ip drp** command. This example shows the packet origination, the IP address that information is routed to, and the route metrics that were returned.

```
Router# debug ip drp

DRP: received v1 packet from 172.69.232.8, via Ethernet0
DRP: RTQUERY for 172.69.58.94 returned internal=0, external=0
```

Table 70 describes the significant fields shown in the display.

Table 70 debug ip drp Field Descriptions

Field	Description
DRP: received v1 packet from 172.69.232.8, via Ethernet0	Router received a version 1 DRP packet from the IP address shown, via the interface shown.
DRP: RTQUERY for 172.69.58.94	DRP packet contained two Route Query requests. The first request was for the distance to the IP address 171.69.113.50.
internal	If nonzero, the metric for the internal distance of the route that the router uses to send packets in the direction of the client. The internal distance is the distance within the autonomous system of the router.
external	If nonzero, the metric for the Border Gateway Protocol (BGP) or external distance used to send packets to the client. The external distance is the distance outside the autonomous system of the router.

debug ip dvmrp

To display information on Distance Vector Multiprotocol Routing Protocol (DVMRP) packets received and sent, use the **debug ip dvmrp** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip dvmrp [**detail** [*access-list*] [**in** | **out**]]

no debug ip dvmrp [**detail** [*access-list*] [**in** | **out**]]

Syntax Description

detail	(Optional) Enables a more detailed level of output and displays packet contents.
access-list	(Optional) Causes the **debug ip dvmrp** command to restrict output to one access list.
in	(Optional) Causes the **debug ip dvmrp** command to output packets received in DVMRP reports.
out	(Optional) Causes the **debug ip dvmrp** command to output packets sent in DVMRP reports.

Usage Guidelines

Use the **debug ip dvmrp detail** command with care. This command generates a substantial amount of output and can interrupt other activity on the router when it is invoked.

Examples

The following is sample output from the **debug ip dvmrp** command:

```
Router# debug ip dvmrp

DVMRP: Received Report on Ethernet0 from 172.19.244.10
DVMRP: Received Report on Ethernet0 from 172.19.244.11
DVMRP: Building Report for Ethernet0 224.0.0.4
DVMRP: Send Report on Ethernet0 to 224.0.0.4
DVMRP: Sending IGMP Reports for known groups on Ethernet0
DVMRP: Received Report on Ethernet0 from 172.19.244.10
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Building Report for Tunnel0 224.0.0.4
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Send Report on Tunnel0 to 192.168.199.254
DVMRP: Radix tree walk suspension
DVMRP: Send Report on Tunnel0 to 192.168.199.254
```

The following lines show that the router received DVMRP routing information and placed it in the mroute table:

```
DVMRP: Received Report on Ethernet0 from 172.19.244.10
DVMRP: Received Report on Ethernet0 from 172.19.244.11
```

The following lines show that the router is creating a report to send to another DVMRP router:

```
DVMRP: Building Report for Ethernet0 224.0.0.4
DVMRP: Send Report on Ethernet0 to 224.0.0.4
```

Table 71 provides a list of internet multicast addresses supported for host IP implementations.

Table 71 Internet Multicast Addresses

Address	Description	RFC
224.0.0.0	Base address (reserved)	RFC 1112
224.0.0.1	All systems on this subnet	RFC 1112
224.0.0.2	All routers on this subnet	
224.0.0.3	Unassigned	
224.0.0.4	DVMRP routers	RFC 1075
224.0.0.5	OSPFIGP all routers	RFC 1583

The following lines show that a protocol update report has been sent to all known multicast groups. Hosts use IGMP reports to communicate with routers and to request to join a multicast group. In this case, the router is sending an IGMP report for every known group to the host, which is running mrouted. The host the responds as though the router was a host on the LAN segment that wants to receive multicast packets for the group.

```
DVMRP: Sending IGMP Reports for known groups on Ethernet0
```

The following is sample output from the **debug ip dvmrp detail** command:

```
Router# debug ip dvmrp detail

DVMRP: Sending IGMP Reports for known groups on Ethernet0
DVMRP: Advertise group 224.2.224.2 on Ethernet0
DVMRP: Advertise group 224.2.193.34 on Ethernet0
DVMRP: Advertise group 224.2.231.6 on Ethernet0
DVMRP: Received Report on Tunnel0 from 192.168.199.254
DVMRP: Origin 150.166.53.0/24, metric 13, distance 0
DVMRP: Origin 150.166.54.0/24, metric 13, distance 0
DVMRP: Origin 150.166.55.0/24, metric 13, distance 0
DVMRP: Origin 150.166.56.0/24, metric 13, distance 0
DVMRP: Origin 150.166.92.0/24, metric 12, distance 0
DVMRP: Origin 150.166.100.0/24, metric 12, distance 0
DVMRP: Origin 150.166.101.0/24, metric 12, distance 0
DVMRP: Origin 150.166.142.0/24, metric 8, distance 0
DVMRP: Origin 150.166.200.0/24, metric 12, distance 0
DVMRP: Origin 150.166.237.0/24, metric 12, distance 0
DVMRP: Origin 150.203.5.0/24, metric 8, distance 0
```

The following lines show that this group is available to the DVMRP router. The mrouted process on the host will forward the source and multicast information for this group through the DVMRP cloud to other members.

```
DVMRP: Advertise group 224.2.224.2 on Ethernet0
```

The following lines show the DVMRP route information:

```
DVMRP: Origin 150.166.53.0/24, metric 13, distance 0
DVMRP: Origin 150.166.54.0/24, metric 13, distance 0
```

The *metric* is the number of hops the route has covered, and the *distance* is the administrative distance.

debug ip eigrp

To display information on Enhanced Interior Gateway Routing Protocol (EIGRP) packets, use the **debug ip eigrp** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip eigrp

no debug ip eigrp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command helps you analyze the packets that are sent and received on an interface. Because the **debug ip eigrp** command generates a substantial amount of output, only use it when traffic on the network is light.

Examples

The following is sample output from the **debug ip eigrp** command:

```
Router# debug ip eigrp

IP-EIGRP: Processing incoming UPDATE packet
IP-EIGRP: Ext 192.168.3.0 255.255.255.0 M 386560 - 256000 130560 SM 360960 - 256000 104960
IP-EIGRP: Ext 192.168.0.0 255.255.255.0 M 386560 - 256000 130560 SM 360960 - 256000 104960
IP-EIGRP: Ext 192.168.3.0 255.255.255.0 M 386560 - 256000 130560 SM 360960 - 256000 104960
IP-EIGRP: 172.69.43.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 172.69.43.0 255.255.255.0 metric 371200 - 256000 115200
IP-EIGRP: 192.135.246.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 192.135.246.0 255.255.255.0 metric 46310656 - 45714176 596480
IP-EIGRP: 172.69.40.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 172.69.40.0 255.255.255.0 metric 2272256 - 1657856 614400
IP-EIGRP: 192.135.245.0 255.255.255.0, - do advertise out Ethernet0/1
IP-EIGRP: Ext 192.135.245.0 255.255.255.0 metric 40622080 - 40000000 622080
IP-EIGRP: 192.135.244.0 255.255.255.0, - do advertise out Ethernet0/1
```

Table 72 describes the significant fields shown in the display.

Table 72 debug ip eigrp Field Descriptions

Field	Description
IP-EIGRP:	Indicates EIGRP packet information.
Ext	Indicates that the following address is an external destination rather than an internal destination, which would be labeled as Int.
M	Displays the computed metric, which includes SM and the cost between this router and the neighbor. The first number is the composite metric. The next two numbers are the inverse bandwidth and the delay, respectively.
SM	Displays the metric as reported by the neighbor.

debug ip error

To display IP errors, use the **debug ip error** command in privileged EXEC mode. To disable debugging errors, use the **no** form of this command.

debug ip error *access-list-number* [**detail**] [**dump**]

no debug ip error

Syntax Description

access-list-number	(Optional) The IP access list number that you can specify. If the datagram is not permitted by that access list, the related debugging output (or IP error) is suppressed. Standard, extended, and expanded access lists are supported. The range of standard and extended access lists is from 1 to 199. The range of expanded access lists is from 1300 to 2699.
detail	(Optional) Displays detailed IP error debugging information.
dump	(Hidden) Displays IP error debugging information along with raw packet data in hexadecimal and ASCII forms. This keyword can be enabled with individual access lists and also with the **detail** keyword. **Note** The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution notes below, in the usage guidelines, for more specific information.

Defaults

No default behavior or values

Command Modes

Privileged EXEC

Usage Guidelines

This command is used for IP error debugging. The output displays IP errors which are locally detected by this router.

Caution Enabling this command will generate output only if IP errors occur. However, if the router starts to receive many packets that contain errors, substantial output may be generated and severely affect system performance. This command should be used with caution in production networks. It should only be enabled when traffic on the IP network is low, so other activity on the system is not adversely affected. Enabling the **detail** and **dump** keywords use the highest level of system resources of the available configuration options for this command, so a high level of caution should be applied when enabling either of these keywords.

Caution The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. Because of the risk of using significant CPU utilization, the dump keyword is hidden from the user and cannot be seen using the “?” prompt. The length of the displayed packet information may exceed the actual packet length and include additional padding bytes that do not belong to the IP packet.

Also note that the beginning of a packet may start at different locations in the dump output depending on the specific router, interface type, and packet header processing that may have occurred before the output is displayed.

Examples

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

```
debug ip error

IP packet errors debugging is on

04:04:45:IP:s=10.8.8.1 (Ethernet0/1), d=10.1.1.1, len 28, dispose ip.hopcount
```

The IP error in the above output was caused when the router attempted to forward a packet with a time-to-live (TTL) value of 0. The “ip.hopcount” traffic counter is incremented when a packet is dropped because of an error. This error is also displayed in the output of the **show ip traffic** command by the “bad hop count” traffic counter.

Table 73 describes the significant fields shown in the display.

Table 73 debug ip error Field Descriptions

Field	Description
IP:s=10.8.8.1 (Ethernet0/1)	The packet source IP address and interface.
d=10.1.1.1, len 28	The packet destination IP address and prefix length.
dispose ip.hopcount	This traffic counter increments when an IP packet is dropped because of an error.

The following is sample output from the **debug ip error** command enabled with the **detail** keyword:

```
debug ip error detail

IP packet errors debugging is on (detailed)

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.1.1.1, len 28, dispose udp.noport
1d08h:    UDP src=41921, dst=33434

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.2.2.2, len 28, dispose ip.hopcount
1d08h:    UDP src=33691, dst=33434
```

The detailed output includes layer 4 information in addition to the standard output. The IP error in the above output was caused when the router received a UDP packet when no application was listening to the UDP port. The “udp.noport” traffic counter is incremented when the router drops a UDP packet because of this error. This error is also displayed in the output of the **show ip traffic** command by the “no port” traffic counter under “UDP statistics.”

Table 74 describes the significant fields shown in the display.

Table 74 debug ip error detail Field Descriptions

Field	Description
IP:s=10.0.19.100 (Ethernet0/1)	The IP packet source IP address and interface.

Table 74 debug ip error detail Field Descriptions (continued)

Field	Description
d=10.1.1.1, len 28	The IP packet destination and prefix length.
dispose udp.noport	The traffic counter that is incremented when a UDP packet is dropped because of this error.

The following is sample output from the **debug ip error** command enabled with the **detail** and **dump** keywords:

```
debug ip error detail dump

IP packet errors debugging is on (detailed) (dump)

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.1.1.1, len 28, dispose udp.noport
1d08h:    UDP src=37936, dst=33434
03D72360:                      0001 42AD4242            ..B-BB
03D72370:0002FCA5 DC390800 4500001C 30130000  ..|%\9..E...0...
03D72380:01116159 0A001364 0A010101 9430829A  ..aY...d.....0..
03D72390:0008C0AD                             ..@-

1d08h:IP:s=10.0.19.100 (Ethernet0/1), d=10.2.2.2, len 28, dispose ip.hopcount
1d08h:    UDP src=41352, dst=33434
03C01600:                      0001 42AD4242            ..B-BB
03C01610:0002FCA5 DC390800 4500001C 302A0000  ..|%\9..E...0*..
03C01620:01116040 0A001364 0A020202 A188829A  ..`@...d....!...
03C01630:0008B253                             ..2S
```

Note The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution in the usage guidelines section of this command reference page for more specific information.

The output from the **debug ip error** command, when the **dump** keyword is enabled, provides raw packet data in hexadecimal and ASCII forms. This addtional output is displayed in addition to the standard output. The **dump** keyword can be used with all of the available configuration options of this command.

Table 75 describes the standard output fields shown in the display.

Table 75 debug ip error detail dump Field Descriptions

Field	Description
IP:s=10.0.19.100 (Ethernet0/1)	The IP packet source IP address and interface.
d=10.1.1.1, len 28	The IP packet destination and prefix length.
dispose udp.noport	The traffic counter that is incremented when a UDP packet is dropped because of this error.

Related Commands

Command	Description
show ip traffic	Displays statistics about IP traffic.

debug ip ftp

To activate the debugging option to track the transactions submitted during an FTP session, use the **debug ip ftp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip ftp

no debug ip ftp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug ip ftp** command is useful for debugging problems associated with FTP.

Examples

The following is an example of the **debug ip ftp** command:

```
Router# debug ip ftp

FTP transactions debugging is on
```

The following is sample output from the **debug ip ftp** command:

```
FTP: 220 ProFTPD 1.2.0pre8 Server (DFW Nostrum FTP Server) [defiant.dfw.nostrum.com]
Dec 27 22:12:09.133: FTP: ---> USER router
Dec 27 22:12:09.133: FTP: 331 Password required for router.
Dec 27 22:12:09.137: FTP: ---> PASS WQHK5JY2
Dec 27 22:12:09.153: FTP: 230 Anonymous access granted, restrictions apply.
Dec 27 22:12:09.153: FTP: ---> TYPE I
Dec 27 22:12:09.157: FTP: 200 Type set to I.
Dec 27 22:12:09.157: FTP: ---> PASV
.
.
.
.
.
.
.
.
.
.
.
.
.
.
Dec 27 22:12:09.173: FTP: ---> QUIT
Dec 27 22:12:09.181: FTP: 221 Goodbye.
```

debug ip http authentication

To troubleshoot HTTP authentication problems, use privileged EXEC command. The **no** form of this command disables debugging output.

debug ip http authentication

no debug ip http authentication

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug ip http authentication** command displays the authentication method the router attempted and authentication-specific status messages.

Examples

The following is sample output from the **debug ip http authentication** command:

```
Router# debug ip http authentication

Authentication for url '/' '/' level 15 privless '/'
Authentication username = 'local15' priv-level = 15 auth-type = local
```

Table 76 describes the significant fields shown in the display.

Table 76 debug ip http authentication Command Descriptions

Field	Description
Authentication for url	Provides information about the URL in different forms.
Authentication username	Identifies the user.
priv-level	Indicates the user privilege level.
auth-type	Indicates the authentication method.

debug ip http ezsetup

To display the configuration changes that occur during the EZ Setup process, use the **debug ip http ezsetup** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip http ezsetup

no debug ip http ezsetup

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug ip http ezsetup** command to verify the EZ Setup actions without changing the configuration of the router.

EZ Setup is a form you fill out to perform basic router configuration from most HTML browsers.

Examples

The following is sample output from the **debug ip http ezsetup** that shows the configuration changes for the router when the EZ Setup form has been submitted:

```
Router# debug ip http ezsetup

service timestamps debug
service timestamps log
service password-encryption
!
hostname router-name
!
enable secret router-pw
line vty 0 4
password router-pw
!
interface ethernet 0
 ip address 172.69.52.9 255.255.255.0
 no shutdown
 ip helper-address 172.31.2.132
ip name-server 172.31.2.132
isdn switch-type basic-5ess
username Remote-name password Remote-chap
interface bri 0
 ip unnumbered ethernet 0
 encapsulation ppp
 no shutdown
 dialer map ip 192.168.254.254 speed 56 name Remote-name Remote-number
 isdn spid1 spid1
 isdn spid2 spid2
 ppp authentication chap callin
 dialer-group 1
!
ip classless
access-list 101 deny udp any any eq snmp
access-list 101 deny udp any any eq ntp
access-list 101 permit ip any any
dialer-list 1 list 101
ip route 0.0.0.0 0.0.0.0 192.168.254.254
ip route 192.168.254.254 255.255.255.255 bri 0
logging buffered
```

```
snmp-server community public RO
ip http server
ip classless
ip subnet-zero
!
end
```

Related Commands

Command	Description
debug ip http token	Displays individual tokens parsed by the HTTP server.
debug ip http transaction	Displays HTTP server transaction processing.
debug ip http url	Displays the URLs accessed from the router.

debug ip http ssi

To display information about the HTML SSI EXEC command or HTML SSI ECHO command, use the **debug ip http ssi** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip http ssi

no debug ip http ssi

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug ip http ssi** command:

```
Router# debug ip http ssi

HTML: filtered command ‘exec cmd="show users"’
HTML: SSI command ‘exec’
HTML: SSI tag ‘cmd’ = "show users"
HTML: Executing CLI ‘show users’ in mode ‘exec’ done
```

The following line shows the contents of the SSI EXEC command:

```
HTML: filtered command ‘exec cmd="show users"’
```

The following line indicates the type of SSI command that was requested:

```
HTML: SSI command ‘exec’
```

The following line shows the argument *show users* assigned to the tag cmd:

```
HTML: SSI tag ’cmd’ = "show users"
```

The following line indicates that the

show users command is being executed in EXEC mode:

```
HTML: Executing CLI ‘show users’ in mode ‘exec’ done
```

debug ip http token

To display individual tokens parsed by the HTTP server, use the **debug ip http token** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip http token

no debug ip http token

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug ip http token** command to display low-level HTTP server parsings. To display high-level HTTP server parsings, use the **debug ip http transaction** command.

Examples

The following is part of a sample output from the **debug ip http token** command. In this example, the browser accessed the router's home page *http://router-name/*. The output gives the token parsed by the HTTP server and its length.

```
Router# debug ip http token

HTTP: token len 3: 'GET'
HTTP: token len 1: ' '
HTTP: token len 1: '/'
HTTP: token len 1: ' '
HTTP: token len 4: 'HTTP'
HTTP: token len 1: '/'
HTTP: token len 1: '1'
HTTP: token len 1: '.'
HTTP: token len 1: '0'
HTTP: token len 2: '\15\12'
HTTP: token len 7: 'Referer'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 4: 'http'
HTTP: token len 1: ':'
HTTP: token len 1: '/'
HTTP: token len 1: '/'
HTTP: token len 3: 'www'
HTTP: token len 1: '.'
HTTP: token len 3: 'thesite'
HTTP: token len 1: '.'
HTTP: token len 3: 'com'
HTTP: token len 1: '/'
HTTP: token len 2: '\15\12'
HTTP: token len 10: 'Connection'
HTTP: token len 1: ':'
HTTP: token len 1: ' '
HTTP: token len 4: 'Keep'
HTTP: token len 1: '-'
HTTP: token len 5: 'Alive'
HTTP: token len 2: '\15\12'
HTTP: token len 4: 'User'
HTTP: token len 1: '-'
HTTP: token len 5: 'Agent'
HTTP: token len 1: ':'
```

```
HTTP: token len 1: ' '
HTTP: token len 7: 'Mozilla'
HTTP: token len 1: '/'
HTTP: token len 1: '2'
HTTP: token len 1: '.'
.
.
.
```

Related Commands

Command	Description
debug ip http ezsetup	Displays the configuration changes that occur during the EZ Setup process.
debug ip http transaction	Displays HTTP server transaction processing.
debug ip http url	Displays the URLs accessed from the router.

debug ip http transaction

To display HTTP server transaction processing, use the **debug ip http transaction** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip http transaction

no debug ip http transaction

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug ip http transaction** command to display what the HTTP server is parsing at a high level. To display what the HTTP server is parsing at a low level, use the **debug ip http token** command.

Examples

The following is sample output from the **debug ip http transaction** command. In this example, the browser accessed the router's home page *http://router-name/*.

```
Router# debug ip http transaction

HTTP: parsed uri '/'
HTTP: client version 1.0
HTTP: parsed extension Referer
HTTP: parsed line http://www.company.com/
HTTP: parsed extension Connection
HTTP: parsed line Keep-Alive
HTTP: parsed extension User-Agent
HTTP: parsed line Mozilla/2.01 (X11; I; FreeBSD 2.1.0-RELEASE i386)
HTTP: parsed extension Host
HTTP: parsed line router-name
HTTP: parsed extension Accept
HTTP: parsed line image/gif, image/x-xbitmap, image/jpeg, image/
HTTP: parsed extension Authorization
HTTP: parsed authorization type Basic
HTTP: received GET ''
```

Table 77 lists describes some of the fields in the output.

Table 77 debug ip http transaction Field Descriptions

Field	Description
HTTP: parsed uri '/'	Uniform resource identifier that is requested.
HTTP: client version 1.0	Client HTTP version.
HTTP: parsed extension Referer	HTTP extension.
HTTP: parsed line http://www.company.com/	Value of HTTP extension.
HTTP: received GET ''	HTTP request method.

Related Commands

Command	Description
debug ip http ezsetup	Displays the configuration changes that occur during the EZ Setup process.
debug ip http token	Displays individual tokens parsed by the HTTP server.
debug ip http url	Shows the URLs accessed from the router.

debug ip http url

To show the URLs accessed from the router, use the **debug ip http url** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip http url

no debug ip http url

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug ip http url** command to keep track of the URLs that are accessed and to determine from which hosts the URLs are accessed.

Examples

The following output is from the **debug ip http url** command. In this example, the HTTP server accessed the URLs and */exec*. The output shows the URL being requested and the IP address of the host requesting the URL.

```
Router# debug ip http url

HTTP: processing URL '/' from host 172.31.2.141
HTTP: processing URL '/exec' from host 172.31.2.141
```

Related Commands

Command	Description
debug ip http ezsetup	Displays the configuration changes that occur during the EZ Setup process.
debug ip http token	Displays individual tokens parsed by the HTTP server.
debug ip http transaction	Displays HTTP server transaction processing.

debug ip icmp

To display information on Internal Control Message Protocol (ICMP) transactions, use the **debug ip icmp** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip icmp

no debug ip icmp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command helps you determine whether the router is sending or receiving ICMP messages. Use it, for example, when you are troubleshooting an end-to-end connection problem.

Note For more information about the fields in **debug ip icmp** command output, refer to RFC-792, Internet Control Message Protocol; Appendix I of RFC-950, Internet Standard Subnetting Procedure; and RFC-1256, ICMP Router Discovery Messages.

Examples

The following is sample output from the **debug ip icmp** command:

```
Router# debug ip icmp

ICMP: rcvd type 3, code 1, from 10.95.192.4
ICMP: src 10.56.0.202, dst 172.69.16.1, echo reply
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: src 172.69.12.35, dst 172.69.20.7, echo reply
ICMP: dst (255.255.255.255) protocol unreachable rcv from 10.31.7.21
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: dst (255.255.255.255) protocol unreachable rcv from 10.31.7.21
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: src 10.56.0.202, dst 172.69.16.1, echo reply
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
ICMP: dst (255.255.255.255) protocol unreachable rcv from 10.31.7.21
ICMP: dst (10.120.1.0) port unreachable rcv from 10.120.1.15
```

Table 78 describes the significant fields shown in the display.

Table 78 debug ip icmp Field Descriptions

Field	Description
ICMP:	Indication that this message describes an ICMP packet.
rcvd type 3	The type field can be one of the following: • 0—Echo Reply • 3—Destination Unreachable • 4—Source Quench • 5—Redirect • 8—Echo • 9—Router Discovery Protocol Advertisement • 10—Router Discovery Protocol Solicitations • 11—Time Exceeded • 12—Parameter Problem • 13—Timestamp • 14—Timestamp Reply • 15—Information Request • 16—Information Reply • 17—Mask Request • 18—Mask Reply

Table 78 debug ip icmp Field Descriptions (continued)

Field	Description
code 1	This field is a code. The meaning of the code depends upon the type field value, as follows: • Echo and Echo Reply—The code field is always zero. • Destination Unreachable—The code field can have the following values: —0—Network unreachable —1—Host unreachable —2—Protocol unreachable —3—Port unreachable —4—Fragmentation needed and DF bit set —5—Source route failed • Source Quench—The code field is always 0. • Redirect—The code field can have the following values: —0—Redirect datagrams for the network —1—Redirect datagrams for the host —2—Redirect datagrams for the command mode of service and network —3—Redirect datagrams for the command mode of service and host • Router Discovery Protocol Advertisements and Solicitations—The code field is always zero. • Time Exceeded—The code field can have the following values: —0—Time to live exceeded in transit —1—Fragment reassembly time exceeded • Parameter Problem—The code field can have the following values: —0—General problem —1—Option is missing —2—Option missing, no room to add • Timestamp and Timestamp Reply—The code field is always zero. • Information Request and Information Reply—The code field is always zero. • Mask Request and Mask Reply—The code field is always zero.
from 10.95.192.4	Source address of the ICMP packet.

Table 79 describes the significant fields in the second line of the display.

Table 79 debug ip icmp Field Descriptions

Field	Description
ICMP:	Indicates that this message describes an ICMP packet.
src 10.56.10.202	Address of the sender of the echo.
dst 172.69.16.1	Address of the receiving router.
echo reply	Indicates that the router received an echo reply.

Other messages that the **debug ip icmp** command can generate follow.

When an IP router or host sends out an ICMP mask request, the following message is generated when the router sends a mask reply:

```
ICMP: sending mask reply (255.255.255.0) to 172.69.80.23 via Ethernet0
```

The following two lines are examples of the two forms of this message. The first form is generated when a mask reply comes in after the router sends out a mask request. The second form occurs when the router receives a mask reply with a nonmatching sequence and ID. Refer to Appendix I of RFC 950, *Internet Standard Subnetting Procedures*, for details.

```
ICMP: mask reply 255.255.255.0 from 172.69.80.31
ICMP: unexpected mask reply 255.255.255.0 from 172.69.80.32
```

The following output indicates that the router sent a redirect packet to the host at address 172.69.80.31, instructing that host to use the gateway at address 172.69.80.23 in order to reach the host at destination address 172.69.1.111:

```
ICMP: redirect sent to 172.69.80.31 for dest 172.69.1.111 use gw 172.69.80.23
```

The following message indicates that the router received a redirect packet from the host at address 172.69.80.23, instructing the router to use the gateway at address 172.69.80.28 in order to reach the host at destination address 172.69.81.34:

```
ICMP: redirect rcvd from 172.69.80.23 -- for 172.69.81.34 use gw 172.69.80.28
```

The following message is displayed when the router sends an ICMP packet to the source address (172.69.94.31 in this case), indicating that the destination address (172.69.13.33 in this case) is unreachable:

```
ICMP: dst (172.69.13.33) host unreachable sent to 172.69.94.31
```

The following message is displayed when the router receives an ICMP packet from an intermediate address (172.69.98.32 in this case), indicating that the destination address (172.69.13.33 in this case) is unreachable:

```
ICMP: dst (172.69.13.33) host unreachable rcv from 172.69.98.32
```

Depending on the code received (as Table 78 describes), any of the unreachable messages can have any of the following "strings" instead of the "host" string in the message:

```
net
protocol
port
frag. needed and DF set
source route failed
prohibited
```

The following message is displayed when the TTL in the IP header reaches zero and a time exceed ICMP message is sent. The fields are self-explanatory.

```
ICMP: time exceeded (time to live) send to 10.95.1.4 (dest was 172.69.1.111)
```

The following message is generated when parameters in the IP header are corrupted in some way and the parameter problem ICMP message is sent. The fields are self-explanatory.

```
ICMP: parameter problem sent to 128.121.1.50 (dest was 172.69.1.111)
```

Based on the preceding information, the remaining output can be easily understood:

```
ICMP: parameter problem rcvd 172.69.80.32
ICMP: source quench rcvd 172.69.80.32
ICMP: source quench sent to 128.121.1.50 (dest was 172.69.1.111)
ICMP: sending time stamp reply to 172.69.80.45
ICMP: sending info reply to 172.69.80.12
ICMP: rdp advert rcvd type 9, code 0, from 172.69.80.23
ICMP: rdp solicit rcvd type 10, code 0, from 172.69.80.43
```

debug ip igmp

To display Internet Group Management Protocol (IGMP) packets received and sent, and IGMP-host related events, use the **debug ip igmp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip igmp

no debug ip igmp

Syntax Description

This command has no arguments or keywords.

Defaults

None

Command History

Release	Modification
10.2	This command was introduced.
12.1(3)T	Additional fields were added to the output of this command to support the Source Specific Multicast (SSM) feature.

Usage Guidelines

This command helps discover whether the IGMP processes are functioning. In general, if IGMP is not working, the router process never discovers that another host is on the network that is configured to receive multicast packets. In dense mode, this situation will result in packets being delivered intermittently (a few every 3 minutes). In sparse mode, packets will never be delivered.

Use this command in conjunction with the **debug ip pim** and **debug ip mrouting** commands to observe additional multicast activity and to learn the status of the multicast routing process, or why packets are forwarded out of particular interfaces.

Examples

The following is sample output from the **debug ip igmp** command:

```
Router# debug ip igmp

IGMP: Received Host-Query from 172.69.37.33 (Ethernet1)
IGMP: Received Host-Report from 172.69.37.192 (Ethernet1) for 224.0.255.1
IGMP: Received Host-Report from 172.69.37.57 (Ethernet1) for 224.2.127.255
IGMP: Received Host-Report from 172.69.37.33 (Ethernet1) for 225.2.2.2
```

The messages displayed by the **debug ip igmp** command show query and report activity received from other routers and multicast group addresses.

The following is sample output from the **debug ip igmp** command when SSM is enabled. Because IGMP Version 3 lite (IGMP v3lite) requires the host to send IGMP Version 2 (IGMPv2) packets, IGMPv2 host reports also will be displayed in response to the router IGMPv2 queries. If SSM is disabled, the word "ignored" will be displayed in the **debug ip igmp** command output.

```
IGMP:Received v3-lite Report from 10.0.119.142 (Ethernet3/3), group count 1
IGMP:Received v3 Group Record from 10.0.119.142 (Ethernet3/3) for 232.10.10.10
IGMP:Update source 1.1.1.1
IGMP:Send v2 Query on Ethernet3/3 to 224.0.0.1
```

```
IGMP:Received v2 Report from 10.0.119.142 (Ethernet3/3) for 232.10.10.10
IGMP:Update source 1.1.1.1
```

Related Commands

Command	Description
debug ip mrm	Displays MRM control packet activity.
debug ip pim	Displays PIM packets received and sent, and PIM-related events.

debug ip igrp events

To display summary information on Interior Gateway Routing Protocol (IGRP) routing messages that indicate the source and destination of each update, and the number of routes in each update, use the **debug ip igrp events** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip igrp events [*ip-address*]

no debug ip igrp events [*ip-address*]

Syntax Description

ip-address	(Optional) The IP address of an IGRP neighbor.

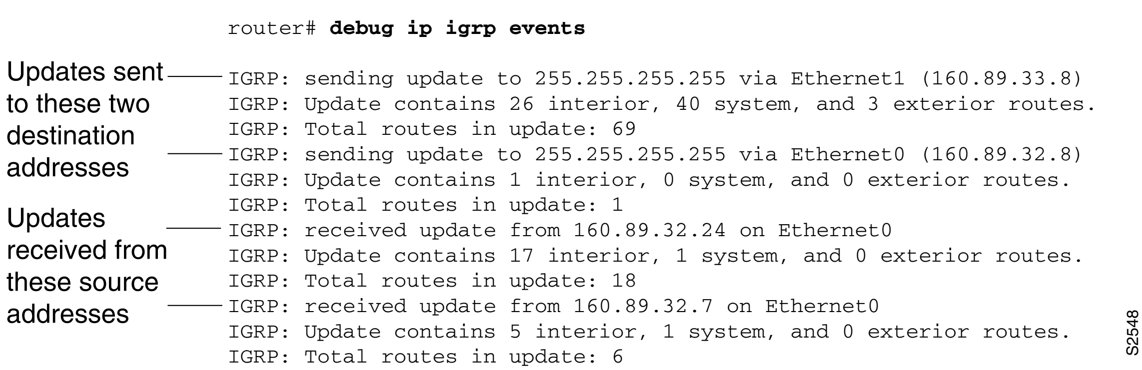
Usage Guidelines

If the IP address of an IGRP neighbor is specified, the resulting **debug ip igrp events** output includes messages describing updates from that neighbor and updates that the router broadcasts toward that neighbor. Messages are not generated for each route.

This command is particularly useful when there are many networks in your routing table. In this case, using **debug ip igrp transactions** could flood the console and make the router unusable. Use **debug ip igrp events** instead to display summary routing information.

Examples

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



This shows that the router has sent two updates to the broadcast address 255.255.255.255. The router also received two updates. Three lines of output describe each of these updates.

The first line indicates whether the router sent or received the update packet, the source or destination address, and the interface through which the update was sent or received. If the update was sent, the IP address assigned to this interface is shown (in parentheses).

```
IGRP: sending update to 255.255.255.255 via Ethernet1 (160.89.33.8)
```

The second line summarizes the number and types of routes described in the update:

```
IGRP: Update contains 26 interior, 40 system, and 3 exterior routes.
```

The third line indicates the total number of routes described in the update:

```
IGRP: Total routes in update: 69
```

debug ip igrp transactions

To display transaction information on Interior Gateway Routing Protocol (IGRP) routing transactions, use the **debug ip igrp transactions** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip igrp transactions [*ip-address*]

no debug ip igrp transactions [*ip-address*]

Syntax Description

ip-address	(Optional) The IP address of an IGRP neighbor.

Usage Guidelines

If the IP address of an IGRP neighbor is specified, the resulting **debug ip igrp transactions** output includes messages describing updates from that neighbor and updates that the router broadcasts toward that neighbor.

When many networks are in your routing table, the **debug ip igrp transactions** command can flood the console and make the router unusable. In this case, use the **debug ip igrp events** command instead to display summary routing information.

Examples

The following is sample output from the **debug ip igrp transactions** command:

```
Router# debug ip igrp transactions

IGRP: received update from 160.89.80.240 on Ethernet
 subnet 160.89.66.0, metric 1300 (neighbor 1200)
 subnet 160.89.56.0, metric 8676 (neighbor 8576)
 subnet 160.89.48.0, metric 1200 (neighbor 1100)
 subnet 160.89.50.0, metric 1300 (neighbor 1200)
 subnet 160.89.40.0, metric 8676 (neighbor 8576)
 network 192.82.152.0, metric 158550 (neighbor 158450)
 network 192.68.151.0, metric 1115511 (neighbor 1115411)
 network 150.136.0.0, metric 16777215 (inaccessible)
 exterior network 129.140.0.0, metric 9676 (neighbor 9576)
 exterior network 140.222.0.0, metric 9676 (neighbor 9576)
IGRP: received update from 160.89.80.28 on Ethernet
 subnet 160.89.95.0, metric 180671 (neighbor 180571)
 subnet 160.89.81.0, metric 1200 (neighbor 1100)
 subnet 160.89.15.0, metric 16777215 (inaccessible)
IGRP: sending update to 255.255.255.255 via Ethernet0 (160.89.64.31)
 subnet 160.89.94.0, metric=847
IGRP: sending update to 255.255.255.255 via Serial1 (160.89.94.31)
 subnet 160.89.80.0, metric=16777215
 subnet 160.89.64.0, metric=1100
```

Updates sent to these two source addresses

Updates received from these two destination addresses

S2549

The output shows that the router being debugged has received updates from two other routers on the network. The router at source address 160.89.80.240 sent information about ten destinations in the update; the router at source address 160.89.80.28 sent information about three destinations in its update. The router being debugged also sent updates—in both cases to the broadcast address 255.255.255.255 as the destination address.

On the second line the first field refers to the type of destination information: “subnet” (interior), “network” (system), or “exterior” (exterior). The second field is the Internet address of the destination network. The third field is the metric stored in the routing table and the metric advertised by the neighbor sending the information. “Metric... inaccessible” usually means that the neighbor router has put the destination in a hold down state.

The entries show that the router is sending updates that are similar, except that the numbers in parentheses are the source addresses used in the IP header. A metric of 16777215 is inaccessible.

Other examples of output that the **debug ip igrp transactions** command can produce follow.

The following entry indicates that the routing table was updated and shows the new edition number (97 in this case) to be used in the next IGRP update:

```
IGRP: edition is now 97
```

Entries such as the following occur on startup or when some event occurs such as an interface making a transition or a user manually clearing the routing table:

```
IGRP: broadcasting request on Ethernet0
IGRP: broadcasting request on Ethernet1
```

The following type of entry can result when routing updates become corrupted between sending and receiving routers:

```
IGRP: bad checksum from 172.69.64.43
```

An entry such as the following should never appear. If it does, the receiving router has a bug in the software or a problem with the hardware. In either case, contact your technical support representative.

```
IGRP: system 45 from 172.69.64.234, should be system 109
```

debug ip inspect

To display messages about Context-Based Access Control (CBAC) events, use the **debug ip inspect** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip inspect {**function-trace** | **object-creation** | **object-deletion** | **events** | **timers** | *protocol* | **detailed**}

no debug ip inspect detailed

Syntax Description

function-trace	Displays messages about software functions called by CBAC.
object-creation	Display messages about software objects being created by CBAC. Object creation corresponds to the beginning of CBAC-inspected sessions.
object-deletion	Displays messages about software objects being deleted by CBAC. Object deletion corresponds to the closing of CBAC-inspected sessions.
events	Displays messages about CBAC software events, including information about CBAC packet processing.
timers	Displays messages about CBAC timer events such as when a CBAC idle timeout is reached.
protocol	Displays messages about CBAC-inspected protocol events, including details about the packets of the protocol. Table 3 provides a list of *protocol* keywords.
detailed	Causes detailed information to be displayed for all the other enabled CBAC debugging. Use this form of the command in conjunction with other CBAC debugging commands.

Table 80 Protocol Keywords for the debug ip inspect Command

Application Protocol	protocol keyword
Transport-layer protocols	
TCP	tcp
UDP	udp
Application-layer protocols	
CU-SeeMe	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the FTP tokens parsed)	ftp-tokens
H.323 (version 1 and version 2)	h323
HTTP	http
Microsoft NetShow	netshow
UNIX r-commands (rlogin, rexec, rsh)	rcmd
RealAudio	realaudio
RPC	rpc
RTSP	rtsp

Table 80 Protocol Keywords for the debug ip inspect Command (continued)

Application Protocol	protocol keyword
SMTP	smtp
SQL*Net	sqlnet
StreamWorks	streamworks
TFTP	tftp
VDOLive	vdolive

Command History

Release	Modification
11.2P	This command was introduced.
12.0(5)T	NetShow support was introduced.
12.0(7)T	H.323 V2 and RTSP protocol support was introduced

Examples

The following is sample output from the **debug ip inspect function-trace** command:

```
*Mar  2 01:16:16: CBAC FUNC: insp_inspection
*Mar  2 01:16:16: CBAC FUNC: insp_pre_process_sync
*Mar  2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar  2 01:16:16: CBAC FUNC: insp_find_pregen_session
*Mar  2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar  2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar  2 01:16:16: CBAC FUNC: insp_get_irc_of_idb
*Mar  2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar  2 01:16:16: CBAC FUNC: insp_create_sis
*Mar  2 01:16:16: CBAC FUNC: insp_inc_halfopen_sis
*Mar  2 01:16:16: CBAC FUNC: insp_link_session_to_hash_table
*Mar  2 01:16:16: CBAC FUNC: insp_inspect_pak
*Mar  2 01:16:16: CBAC FUNC: insp_l4_inspection
*Mar  2 01:16:16: CBAC FUNC: insp_process_tcp_seg
*Mar  2 01:16:16: CBAC FUNC: insp_listen_state
*Mar  2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar  2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar  2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar  2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar  2 01:16:16: CBAC FUNC: insp_process_syn_packet
*Mar  2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar  2 01:16:16: CBAC FUNC: insp_create_tcp_host_entry
*Mar  2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar  2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar  2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar  2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar  2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar  2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar  2 01:16:16: CBAC* FUNC: insp_inspect_pak
```

```
*Mar  2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar  2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar  2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar  2 01:16:16: CBAC FUNC: insp_dec_halfopen_sis
*Mar  2 01:16:16: CBAC FUNC: insp_remove_sis_from_host_entry
*Mar  2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
```

This output shows the functions called by CBAC as a session is inspected. Entries with an asterisk (*) after the word "CBAC" are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect object-creation** and **debug ip inspect object-deletion** command:

```
*Mar  2 01:18:30: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar  2 01:18:30: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar  2 01:18:30: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar  2 01:18:30: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar  2 01:18:30: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar  2 01:18:30: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar  2 01:18:30: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item 25A3634
*Mar  2 01:18:31: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1
```

The following is sample output from the **debug ip inspect object-creation**, **debug ip inspect object-deletion**, and **debug ip inspect events** commands:

```
*Mar  2 01:18:51: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar  2 01:18:51: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar  2 01:18:51: CBAC Src 10.1.0.1 Port [1:65535]
*Mar  2 01:18:51: CBAC Dst 10.0.0.1 Port [46406:46406]
*Mar  2 01:18:51: CBAC Pre-gen sis 25A3574 created: 10.1.0.1[1:65535]
30.0.0.1[46406:46406]
*Mar  2 01:18:51: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar  2 01:18:51: CBAC sis 25C1CC4 initiator_addr (10.1.0.1:20) responder_addr
(30.0.0.1:46406) initiator_alt_addr (40.0.0.1:20) responder_alt_addr (10.0.0.1:46406)
*Mar  2 01:18:51: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar  2 01:18:51: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar  2 01:18:51: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar  2 01:18:51: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item 25A3634
*Mar  2 01:18:51: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1
```

The following is sample output from the **debug ip inspect timers** command:

```
*Mar  2 01:19:15: CBAC Timer Init Leaf: Pre-gen sis 25A3574
*Mar  2 01:19:15: CBAC Timer Start: Pre-gen sis 25A3574 Timer: 25A35D8 Time: 30000
milisecs
*Mar  2 01:19:15: CBAC Timer Init Leaf: sis 25C1CC4
*Mar  2 01:19:15: CBAC Timer Stop: Pre-gen sis 25A3574 Timer: 25A35D8
*Mar  2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 30000 milisecs
*Mar  2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 3600000 milisecs
*Mar  2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 5000 milisecs
*Mar  2 01:19:15: CBAC Timer Stop: sis 25C1CC4 Timer: 25C1D5C
```

The following is sample output from the **debug ip inspect tcp** command:

```
*Mar  2 01:20:43: CBAC* sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar  2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar  2 01:20:43: CBAC sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar  2 01:20:43: CBAC sis 25A3604 ftp L7 inspect result: PASS packet
*Mar  2 01:20:43: CBAC* sis 25A3604 pak 2544374 TCP P ack 4200176247 seq 4223720032(30)
(10.0.0. 1:46409) <= (10.1.0.1:21)
*Mar  2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar  2 01:20:43: CBAC* sis 25A3604 pak 25412F8 TCP P ack 4223720062 seq 4200176247(15)
(10.0.0. 1:46409) => (10.1.0.1:21)
```

```
*Mar  2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar  2 01:20:43: CBAC sis 25C1CC4 pak 2544734 TCP S seq 4226992037(0) (10.1.0.1:20) =>
(10.0.0.1:46411)
*Mar  2 01:20:43: CBAC* sis 25C1CC4 pak 2541E38 TCP S ack 4226992038 seq 4203405054(0)
(10.1.0.1:20) <= (10.0.0.1:46411)
```

This sample shows TCP packets being processed, and lists the corresponding acknowledge (ACK) packet numbers and sequence (SEQ) numbers. The number of data bytes in the TCP packet is shown in parentheses—for example, (22). For each packet shown, the addresses and port numbers are shown separated by a colon. For example, (10.1.0.1:21) indicates an IP address of 10.1.0.1 and a TCP port number of 21.

Entries with an asterisk (*) after the word "CBAC" are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect tcp** and **debug ip inspect detailed** commands:

```
*Mar  2 01:20:58: CBAC* Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar  2 01:20:58:  P ack 4223720160 seq 4200176262(22)
*Mar  2 01:20:58: CBAC* Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar  2 01:20:58: CBAC* sis 25A3604 SIS_OPEN
*Mar  2 01:20:58: CBAC* Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1),
len 76,proto=6
*Mar  2 01:20:58: CBAC sis 25A3604 Saving State: SIS_OPEN/ESTAB iisn 4200176160 i_rcvnxt
4223720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262 r_sndnxt
4223720160 r_rcvwnd 8760
*Mar  2 01:20:58: CBAC* sis 25A3604 pak 2541E38 TCP P ack 4223720160 seq 4200176262(22)
(30.0.0.1:46409) => (40.0.0.1:21)
*Mar  2 01:20:58: CBAC* sis 25A3604 pak 2541E38 SIS_OPEN/ESTAB TCP seq 4200176262(22)
Flags: ACK 4223720160 PSH
*Mar  2 01:20:58: CBAC* sis 25A3604 pak 2541E38 --> SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223720160 i_sndnxt 4200176284 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262
r_sndnxt 4223720160 r_rcvwnd 8760
*Mar  2 01:20:58: CBAC* sis 25A3604 L4 inspect result: PASS packet 2541E38
(30.0.0.1:46409) (40.0.0.1:21) bytes 22 ftp
*Mar  2 01:20:58: CBAC sis 25A3604 Restoring State: SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223
720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262 r_sndnxt
4223720160 r_rcvwnd 8760
*Mar  2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar  2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar  2 01:20:58: CBAC* Bump up: inspection requires the packet in the process
path(30.0.0.1) (40.0.0.1)
*Mar  2 01:20:58: CBAC Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar  2 01:20:58:  P ack 4223720160 seq 4200176262(22)
*Mar  2 01:20:58: CBAC Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar  2 01:20:58: CBAC sis 25A3604 SIS_OPEN
*Mar  2 01:20:58: CBAC Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1), len
76, proto=6
```

debug ip mbgp dampening

To log route flap dampening activity related to multiprotocol Border Gateway Protocol (BGP), use the **debug ip mbgp dampening** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip mbgp dampening [*access-list-number*]

no debug ip mbgp dampening [*access-list-number*]

Syntax Description

access-list-number	(Optional) The number of an access list in the range from 1 to 99. If an access list number is specified, debugging occurs only for the routes permitted by the access list.

Defaults

Logging for route flap dampening activity is not enabled.

Command History

Release	Modification
11.1(20)CC	This command was introduced.

Examples

The following example shows sample **debug ip mbgp dampening** output:

```
Router# debug ip mbgp dampening
```

```
BGP: charge penalty for 173.19.0.0/16 path 49 with halflife-time 15 reuse/suppress
750/2000
BGP: flapped 1 times since 00:00:00. New penalty is 1000
BGP: charge penalty for 173.19.0.0/16 path 19 49 with halflife-time 15 reuse/suppress
750/2000
BGP: flapped 1 times since 00:00:00. New penalty is 1000
```

debug ip mbgp updates

To log multiprotocol Border Gateway Protocol (BGP)-related information passed in BGP update messages, use the **debug ip mbgp updates** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip mbgp updates

no debug ip mbgp updates

Syntax Description

This command has no arguments or keywords.

Defaults

Logging for multiprotocol BGP-related information in BGP update messages is not enabled.

Command History

Release	Modification
11.1(20)CC	This command was introduced.

Examples

The following example shows sample **debug ip mbgp updates** output:

```
Router# debug ip mbgp updates

BGP: NEXT_HOP part 1 net 200.10.200.0/24, neigh 171.69.233.49, next 171.69.233.34
BGP: 171.69.233.49 send UPDATE 200.10.200.0/24, next 171.69.233.34, metric 0, path 33 34
19 49 109 65000 297 3561 6503
BGP: NEXT_HOP part 1 net 200.10.202.0/24, neigh 171.69.233.49, next 171.69.233.34
BGP: 171.69.233.49 send UPDATE 200.10.202.0/24, next 171.69.233.34, metric 0, path 33 34
19 49 109 65000 297 1239 1800 3597
BGP: NEXT_HOP part 1 net 200.10.228.0/22, neigh 171.69.233.49, next 171.69.233.34
BGP: 171.69.233.49 rcv UPDATE about 222.2.2.0/24, next hop 171.69.233.49, path 49 109
metric 0
BGP: 171.69.233.49 rcv UPDATE about 131.103.0.0/16, next hop 171.69.233.49, path 49 109
metric 0
BGP: 171.69.233.49 rcv UPDATE about 206.205.242.0/24, next hop 171.69.233.49, path 49 109
metric 0
BGP: 171.69.233.49 rcv UPDATE about 1.0.0.0/8, next hop 171.69.233.49, path 49 19 metric 0
BGP: 171.69.233.49 rcv UPDATE about 198.1.2.0/24, next hop 171.69.233.49, path 49 19
metric 0
BGP: 171.69.233.49 rcv UPDATE about 171.69.0.0/16, next hop 171.69.233.49, path 49 metric
0
BGP: 171.69.233.49 rcv UPDATE about 172.19.0.0/16, next hop 171.69.233.49, path 49 metric
0
BGP: nettable_walker 172.19.0.0/255.255.0.0 calling revise_route
BGP: revise route installing 172.19.0.0/255.255.0.0 -> 171.69.233.49
BGP: 171.69.233.19 computing updates, neighbor version 267099, table version 267100,
starting at 0.0.0.0
BGP: NEXT_HOP part 1 net 172.19.0.0/16, neigh 171.69.233.19, next 171.69.233.49
BGP: 171.69.233.19 send UPDATE 172.19.0.0/16, next 171.69.233.49, metric 0, path 33 49
BGP: 1 updates (average = 46, maximum = 46)
BGP: 171.69.233.19 updates replicated for neighbors : 171.69.233.34, 171.69.233.49,
171.69.233.56
BGP: 171.69.233.19 1 updates enqueued (average=46, maximum=46)
```

```
BGP: 171.69.233.19 update run completed, ran for 0ms, neighbor version 267099, start
version 267100, throttled to 267100, check point net 0.0.0.0
```

debug ip mcache

To display IP multicast fast-switching events, use the **debug ip mcache** command. The **no** form of this command disables debugging output.

debug ip mcache [*name* | *address*]

no debug ip mcache [*name* | *address*]

Syntax Description

name	(Optional) The host name.
address	(Optional) The group address.

Usage Guidelines

Use this command when multicast fast switching appears not to be functioning.

Examples

The following is sample output from the **debug ip mcache** command when an IP multicast route is cleared:

```
Router# debug ip mcache

IP multicast fast-switching debugging is on

Router# clear ip mroute *

MRC: Build MAC header for (172.31.60.185/32, 224.2.231.173), Ethernet0
MRC: Fast-switch flag for (172.31.60.185/32, 224.2.231.173), off -> on, caller
ip_mroute_replicate-1
MRC: Build MAC header for (172.31.191.10/32, 224.2.127.255), Ethernet0
MRC: Build MAC header for (172.31.60.152/32, 224.2.231.173), Ethernet0
```

Table 81 explains the significant fields in the display.

Table 81 debug ip mcache Field Descriptions

Field	Description
MRC	Multicast route cache.
Fast-switch flag	Route is fast switched.
(*address*/32)	Host route with 32 bits of mask.
off -> on	State has changed.
caller *string*	The code function that activated the state change.

Related Commands

Command	Description
debug ip dvmrp	Displays information on DVMRP packets received and sent.
debug ip igmp	Displays IGMP packets received and sent, and IGMP-host related events.
debug ip igrp transactions	Displays transaction information on IGRP routing transactions.
debug ip mrm	Displays MRM control packet activity.
debug ip sd	Displays all SD announcements received.

debug ip mds ipc

To debug MDS interprocessor communication, that is, synchronization between the MFIB on the line card and the multicast routing table in the RP, use the **debug ip mds ipc** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip mds ipc {**event** | **packet**}

no debug ip mds ipc {**event** | **packet**}

Syntax Description

event	Displays MDS events when there is a problem.
packet	Displays MDS packets.

Usage Guidelines

Use this command on the line card or RP.

Examples

The following is sample output from the **debug ip mds ipc packet** command:

```
Router# debug ip mds ipc packet

MDFS ipc packet debugging is on
Router#
MDFS: LC sending statistics message to RP with code 0 of size 36
MDFS: LC sending statistics message to RP with code 1 of size 680
MDFS: LC sending statistics message to RP with code 2 of size 200
MDFS: LC sending statistics message to RP with code 3 of size 152
MDFS: LC sending window message to RP with code 36261 of size 8
MDFS: LC received IPC packet of size 60 sequence 36212
```

The following is sample output from the **debug ip mds ipc event** command:

```
Router# debug ip mds ipc event

MDFS: LC received invalid sequence 21 while expecting 20
```

debug ip mds mevent

To debug MFIB route creation, route updates, and so on, use the **debug ip mds mevent** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip mds mevent

no debug ip mds mevent

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command on the line card.

Examples

The following is sample output from the **debug ip mds mevent** command:

```
Router# debug ip mds mevent

MDFS mroute event debugging is on
Router#clear ip mdfs for *
Router#
MDFS: Create (*, 239.255.255.255)
MDFS: Create (192.168.1.1/32, 239.255.255.255), RPF POS2/0/0
MDFS: Add OIF for mroute (192.168.1.1/239.255.255.255) on Fddi0/0/0
MDFS: Create (*, 224.2.127.254)
MDFS: Create (192.168.1.1/32, 224.2.127.254), RPF POS2/0/0
MDFS: Add OIF for mroute (192.168.1.1/224.2.127.254) on Fddi0/0/0
MDFS: Create (128.9.160.67/32, 224.2.127.254), RPF POS2/0/0
```

debug ip mds mpacket

To debug multicast distributed switching (MDS) events such as packet drops, interface drops, and switching failures, use the **debug ip mds mpacket** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip mds mpacket

no debug ip mds mpacket

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command on the line card.

Examples

The following is sample output from the **debug ip mds** mpacket command:

```
Router# debug ip mds mpacket
```

debug ip mds process

To debug line card process level events, use the **debug ip mds process** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip mds process

no debug ip mds process

Usage Guidelines

Use this command on the line card or RP.

Examples

The following is sample output from the **debug ip mds process** command:

```
Router# debug ip mds process

MDFS process debugging is on
Mar 19 16:15:47.448: MDFS: RP queueing mdb message for (210.115.194.5, 224.2.127.254) to
all linecards
Mar 19 16:15:47.448: MDFS: RP queueing midb message for (210.115.194.5, 224.2.127.254) to
all linecards
Mar 19 16:15:47.628: MDFS: RP servicing low queue for LC in slot 0
Mar 19 16:15:47.628: MDFS: RP servicing low queue for LC in slot 2
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.68.224.10, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.68.224.10, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.69.67.106, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (171.69.67.106, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (206.14.154.181, 224.2.127.254) to
all linecards
Mar 19 16:15:48.229: MDFS: RP queueing mdb message for (206.14.154.181, 224.2.127.254) to
all linecards
Mar 19 16:15:48.233: MDFS: RP queueing mdb message for (210.115.194.5, 224.2.127.254) to
all linecards
```

debug ip mhbeat

To monitor the action of the heartbeat trap, use the **debug ip mhbeat** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip mhbeat

no debug ip mhbeat

Syntax Description

This command has no keywords or arguments.

Defaults

Debugging is not enabled.

Command History

Release	Modification
12.1(2)XH	This command was introduced.

Examples

The following is output from the **debug ip mhbeat** command.

```
Router# debug ip mhbeat

 IP multicast heartbeat debugging is on
Router# debug snmp packets

 SNMP packet debugging is on

!
Router(config)# ip multicast heartbeat intervals-of 10

 Dec 23 13:34:21.132: MHBEAT: ip multicast-heartbeat group 224.0.1.53 port 0
         source 0.0.0.0 0.0.0.0 at-least 3 in 5 intervals-of 10 secondsd
Router#
 Dec 23 13:34:23: %SYS-5-CONFIG_I: Configured from console by console
 Dec 23 13:34:31.136: MHBEAT: timer ticked, t=1,i=1,c=0
 Dec 23 13:34:41.136: MHBEAT: timer ticked, t=2,i=2,c=0
 Dec 23 13:34:51.136: MHBEAT: timer ticked, t=3,i=3,c=0
 Dec 23 13:35:01.136: MHBEAT: timer ticked, t=4,i=4,c=0
 Dec 23 13:35:11.136: MHBEAT: timer ticked, t=5,i=0,c=0
 Dec 23 13:35:21.135: Send SNMP Trap for missing heartbeat
 Dec 23 13:35:21.135: SNMP: Queuing packet to 171.69.55.12
 Dec 23 13:35:21.135: SNMP: V1 Trap, ent ciscoExperiment.2.3.1, addr 4.4.4.4, gentrap 6,
spectrap 1
  ciscoIpMRouteHeartBeat.1.0 = 224.0.1.53
  ciscoIpMRouteHeartBeat.2.0 = 0.0.0.0
  ciscoIpMRouteHeartBeat.3.0 = 10
  ciscoIpMRouteHeartBeat.4.0 = 5
  ciscoIpMRouteHeartBeat.5.0 = 0
  ciscoIpMRouteHeartBeat.6.0 = 3
```

Related Commands

Command	Description
ip multicast heartbeat	Monitors the health of multicast delivery, and alerts when the delivery fails to meet certain parameters.

debug ip mobile

To display IP mobility activities, use the **debug ip mobile** command.

debug ip mobile [**advertise** | **host** [*access-list-number*] | **local-area** | **standby**]

Syntax Description

advertise	(Optional) Advertisement information.
host	(Optional) The mobile node host.
access-list-number	(Optional) The number of an IP access list.
local-area	(Optional) The local area.
standby	(Optional) Redundancy activities.

Command History

Release	Modification
12.0(1)T	This command was introduced.
12.0(2)T	The **standby** keyword was added.

Usage Guidelines

Use the **debug ip mobile standby** command to troubleshoot redundancy problems.

Examples

The following is sample output from the **debug ip mobile standby** command. In this example, the active HA receives a registration request from mobile node (MN) 20.0.0.2 and sends a binding update to peer HA 1.0.0.2:

```
MobileIP:MN 20.0.0.2 - sent BindUpd to HA 1.0.0.2 HAA 20.0.0.1
MobileIP:HA standby maint started - cnt 1
MobileIP:MN 20.0.0.2 - sent BindUpd id 3780410816 cnt 0 elapsed 0
adjust -0 to HA 1.0.0.2 in grp 1.0.0.10 HAA 20.0.0.1
```

In this example, the standby HA receives a binding update for MN 20.0.0.2 sent by the active HA:

```
MobileIP:MN 20.0.0.2 - HA rcv BindUpd from 1.0.0.3 HAA 20.0.0.1
```

debug ip mobile advertise

To display advertisement information, use the **debug ip mobile advertise** privileged EXEC command.

debug ip mobile advertise

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(1)T	This command was introduced.

Examples

The following is sample output from the **debug ip mobile advertise** command:

```
Router# debug ip mobile advertise

MobileIP: Agent advertisement sent out Ethernet1/2: type=16, len=10, seq=1,
lifetime=36000,
flags=0x1400(rbhFmGv-rsv-),
Care-of address: 68.0.0.31
Prefix Length ext: len=1 (8 )
```

Table 82 describes the significant fields shown in the display.

Table 82 debug ip mobile advertise Field Descriptions

Field	Description
type	Type of advertisement.
len	Length of extension (in bytes).
seq	Sequence number of this advertisement.
lifetime	Lifetime (in seconds).
flags	Capital letters represent bits that are set; lowercase letters represent unset bits.
Care-of address	IP address.
Prefix Length ext	Number of prefix lengths advertised. This is the bits in the mask of the interface sending this advertisement. Used for roaming detection.

debug ip mobile host

To display IP mobility events, use the **debug ip mobile host** privileged EXEC command.

debug ip mobile host *acl*

Syntax Description

acl	(Optional) Access list.

Command History

Release	Modification
12.0(1)T	This command was introduced.

Examples

The following is sample output from the **debug ip mobile host** command:

```
Router# debug ip mobile host

MobileIP: HA received registration for MN 20.0.0.6 on interface Ethernet1 using COA
68.0.0.31 HA 66.0.0.5 lifetime 30000 options sbdmgvT
MobileIP: Authenticated FA 68.0.0.31 using SPI 110 (MN 20.0.0.6)
MobileIP: Authenticated MN 20.0.0.6 using SPI 300

MobileIP: HA accepts registration from MN 20.0.0.6
MobileIP: Mobility binding for MN 20.0.0.6 updated
MobileIP: Roam timer started for MN 20.0.0.6, lifetime 30000
MobileIP: MH auth ext added (SPI 300) in reply to MN 20.0.0.6
MobileIP: HF auth ext added (SPI 220) in reply to MN 20.0.0.6

MobileIP: HA sent reply to MN 20.0.0.6
```

debug ip mpacket

To display IP multicast packets received and sent, use the **debug ip mpacket** privileged EXEC command. To disable the debugging output, use the **no** form of this command.

debug ip mpacket [**detail** | **fastswitch**] [*access-list*] [*group*]

no debug ip mpacket [**detail** | **fastswitch**] [*access-list*] [*group*]

Syntax Description

detail	(Optional) Causes the **debug ip mpacket** command to display IP header information and MAC address information.
fastswitch	(Optional) Displays IP packet information in the fast path.
access-list	(Optional) The access list number.
group	(Optional) The group name or address.

Defaults

The **debug ip mpacket** command displays all IP multicast packets switched at the process level.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.2	This command was introduced.
12.1(2)T	The **fastswitch** keyword was introduced.

Usage Guidelines

This command displays information for multicast IP packets that are forwarded from this router. By using the *access-list* or *group* argument, you can limit the display to multicast packets from sources described by the access list or a specific multicast group.

Use this command with the **debug ip packet** command to observe additional packet information.

Note The **debug ip mpacket** command generates many messages. Use this command with care so that performance on the network is not affected by the **debug** message traffic.

Examples

The following is sample output from the **debug ip mpacket** command:

```
Router# debug ip mpacket 224.2.0.1

IP: s=10.188.34.54 (Ethernet1), d=224.2.0.1 (Tunnel0), len 88, mforward
IP: s=10.188.34.54 (Ethernet1), d=224.2.0.1 (Tunnel0), len 88, mforward
IP: s=10.188.34.54 (Ethernet1), d=224.2.0.1 (Tunnel0), len 88, mforward
IP: s=10.162.3.27 (Ethernet1), d=224.2.0.1 (Tunnel0), len 68, mforward
```

Table 83 describes the significant fields shown in the display.

Table 83 debug ip mpacket Field Descriptions

Field	Description
IP	IP packet.
s=*address*	Source address of the packet.
(Ethernet1)	Name of the interface that received the packet.
d=*address*	Multicast group address that is the destination for this packet.
(Tunnel0)	Outgoing interface for the packet.
len 88	Number of bytes in the packet. This value will vary depending on the application and the media.
mforward	Packet has been forwarded.

Related Commands

Command	Description
debug ip dvmrp	Displays information on DVMRP packets received and sent.
debug ip igmp	Displays IGMP packets received and sent, and IGMP host-related events.
debug ip mrm	Displays MRM control packet activity.
debug ip packet	Displays general IP debugging information and IPSO security transactions.
debug ip sd	Displays all SD announcements received.

debug ip mrm

To display Multicast Routing Monitor (MRM) control packet activity, use the **debug ip mrm** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug ip mrm

no debug ip mrm

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for MRM is not enabled.

Command History

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

Examples

The following example is sample output for the **debug ip mrm** command on the different devices:

On Manager

```
*Feb 28 16:25:44.009: MRM: Send Beacon for group 239.1.1.1, holdtime 86100 seconds
*Feb 28 16:26:01.095: MRM: Receive Status Report from 10.1.4.2 on Ethernet0
*Feb 28 16:26:01.099: MRM: Send Status Report Ack to 10.1.4.2 for group 239.1.1.1
*Feb 28 16:26:01.103: IP MRM status report -- Test:test2  Receiver:10.1.4.2
*Feb 28 16:26:01.107:   Sender:10.1.1.10        Pkt Loss:4(16%)  Ehsr:1380
```

The last two lines of output on the manager are not part of the debug output; they appeared because an error report was received.

On Test-Sender

```
MRM: Receive Test-Sender Request/Local trigger from 1.1.1.1 on Ethernet0
MRM: Send TS request Ack to 1.1.1.1 for group 239.1.2.3
MRM: Send test packet src:2.2.2.2 dst:239.1.2.3 manager:1.1.1.1
```

On Test-Receiver

```
MRM: Receive Test-Receiver Request/Monitor from 1.1.1.1 on Ethernet0
MRM: Send TR request Ack to 1.1.1.1 for group 239.1.2.3
MRM: Receive Beacon from 1.1.1.1 on Ethernet0
MRM: Send Status Report to 1.1.1.1 for group 239.1.2.3
MRM: Receive Status Report Ack from 1.1.1.1 on Ethernet0
```

debug ip mrouting

To display changes to the IP multicast routing table, use the **debug ip mrouting** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip mrouting [*group*]

no debug ip mrouting [*group*]

Syntax Description

group	(Optional) Group name or address to monitor a single group's packet activity.

Usage Guidelines

This command indicates when the router has made changes to the mroute table. Use the **debug ip pim** and **debug ip mrouting** commands concurrently to obtain additional multicast routing information. In addition, use the **debug ip igmp** command to see why an mroute message is being displayed.

This command generates a substantial amount of output. Use the optional *group* argument to limit the output to a single multicast group.

Examples

The following is sample output from the **debug ip mrouting** command:

```
Router# debug ip mrouting 224.2.0.1

MRT: Delete (10.0.0.0/8, 224.2.0.1)
MRT: Delete (10.4.0.0/16, 224.2.0.1)
MRT: Delete (10.6.0.0/16, 224.2.0.1)
MRT: Delete (10.9.0.0/16, 224.2.0.1)
MRT: Delete (10.16.0.0/16, 224.2.0.1)
MRT: Create (*, 224.2.0.1), if_input NULL
MRT: Create (172.69.15.0/24, 225.2.2.4), if_input Ethernet0, RPF nbr 172.69.61.15
MRT: Create (172.69.39.0/24, 225.2.2.4), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.0.0.0/8, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.4.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.6.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.9.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.16.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
```

The following lines show that multicast IP routes were deleted from the routing table:

```
MRT: Delete (10.0.0.0/8, 224.2.0.1)
MRT: Delete (10.4.0.0/16, 224.2.0.1)
MRT: Delete (10.6.0.0/16, 224.2.0.1)
```

The (*, G) entry in the following line is always because since it is a (*, G). The (*, G) entries are generally created by receipt of an IGMP host report from a group member on the directly connected LAN or by a PIM join message (in sparse mode) that this router receives from a router that is sending joins toward the RP. This router will in turn send a join toward the RP that creates the shared tree (or RP tree).

```
MRT: Create (*, 224.2.0.1), if_input NULL
```

The following lines are an example of creating an (S, G) entry that show a mpacket was received on E0. The second line shows a route being created for a source that is on a directly connected LAN. The RPF means “reverse path forwarding,” whereby the router looks up the source address of the multicast packet in the unicast routing table and asks which interface will be used to send a packet to that source.

```
MRT: Create (172.69.15.0/24, 225.2.2.4), if_input Ethernet0, RPF nbr 172.69.61.15
MRT: Create (172.69.39.0/24, 225.2.2.4), if_input Ethernet1, RPF nbr 0.0.0.0
```

The following lines show that multicast IP routes were added to the routing table. Note the 0.0.0.0 as the RPF, which means the route was created by a source that is directly connected to this router.

```
MRT: Create (10.9.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
MRT: Create (10.16.0.0/16, 224.2.0.1), if_input Ethernet1, RPF nbr 0.0.0.0
```

If the source is not directly connected, the nbr address shown in these lines will be the address of the router that forwarded the packet to this router.

The shortest path tree state maintained in routers consists of source (S), multicast address (G), outgoing interface (OIF), and incoming interface (IIF). The forwarding information is referred to as the multicast forwarding entry for (S,G).

An entry for a shared tree can match packets from any source for its associated group if the packets come through the proper incoming interface as determined by the RPF lookup. Such an entry is denoted as (*,G). A (*,G) entry keeps the same information a (S,G) entry keeps, except that it saves the rendezvous point (RP) address in place of the source address in sparse mode or 0.0.0.0 in dense mode.

Related Commands

Command	Description
debug ip dvmrp	Displays information on DVMRP packets received and transmitted.
debug ip igmp	Displays IGMP packets received and transmitted, as well as IGMP-host related events.
debug ip pim	Displays all SD announcements received.
debug ip packet	Displays general IP debugging information and IPSO security transactions.
debug ip sd	Displays all SD announcements received.

debug ip msdp

To debug MSDP activity, use the **debug ip msdp** privileged EXEC command.

debug ip msdp [*peer-address* | *name*] [**detail**] [**routes**]

Syntax Description

peer-address | *name*	(Optional) Logs debug events for that peer only.
detail	(Optional) Provides more detailed debugging information.
routes	(Optional) Displays the contents of Source-Active messages.

Command History

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

Examples

The following is sample output of the **debug ip msdp** command:

```
Router# debug ip msdp

MSDP debugging is on
Router#
MSDP: 192.150.44.254: Received 1388-byte message from peer
MSDP: 192.150.44.254: SA TLV, len: 1388, ec: 115, RP: 137.39.3.92
MSDP: 192.150.44.254: Peer RPF check passed for 137.39.3.92, used EMBGP peer
MSDP: 192.150.44.250: Forward 1388-byte SA to peer
MSDP: 192.150.44.254: Received 1028-byte message from peer
MSDP: 192.150.44.254: SA TLV, len: 1028, ec: 85, RP: 137.39.3.92
MSDP: 192.150.44.254: Peer RPF check passed for 137.39.3.92, used EMBGP peer
MSDP: 192.150.44.250: Forward 1028-byte SA to peer
MSDP: 192.150.44.254: Received 1388-byte message from peer
MSDP: 192.150.44.254: SA TLV, len: 1388, ec: 115, RP: 137.39.3.111
MSDP: 192.150.44.254: Peer RPF check passed for 137.39.3.111, used EMBGP peer
MSDP: 192.150.44.250: Forward 1388-byte SA to peer
MSDP: 192.150.44.250: Received 56-byte message from peer
MSDP: 192.150.44.250: SA TLV, len: 56, ec: 4, RP: 205.167.76.241
MSDP: 192.150.44.250: Peer RPF check passed for 205.167.76.241, used EMBGP peer
MSDP: 192.150.44.254: Forward 56-byte SA to peer
MSDP: 192.150.44.254: Received 116-byte message from peer
MSDP: 192.150.44.254: SA TLV, len: 116, ec: 9, RP: 137.39.3.111
MSDP: 192.150.44.254: Peer RPF check passed for 137.39.3.111, used EMBGP peer
MSDP: 192.150.44.250: Forward 116-byte SA to peer
MSDP: 192.150.44.254: Received 32-byte message from peer
MSDP: 192.150.44.254: SA TLV, len: 32, ec: 2, RP: 137.39.3.78
MSDP: 192.150.44.254: Peer RPF check passed for 137.39.3.78, used EMBGP peer
MSDP: 192.150.44.250: Forward 32-byte SA to peer
```

Table 84 describes the significant fields shown in the display.

Table 84 debug ip msdp Field Descriptions

Field	Description
MSDP	Protocol being debugged.
192.150.44.254:	IP address of the MSDP peer.
Received 1388-byte message from peer	MSDP event.

debug ip msdp resets

To debug MSDP peer reset reasons, use the **debug ip msdp resets** privileged EXEC command.

debug ip msdp resets

Syntax Description

This command has no arguments or keywords.

Command History

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

debug ip nat

To display information about IP packets translated by the IP Network Address Translation (NAT) feature, use the **debug ip nat** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip nat [*access-list* | **detailed** | **h323** | **pptp**]

no debug ip nat [*access-list* | **detailed** | **h323** | **pptp**]

Syntax Description

access-list	(Optional) The standard IP access list number. If the datagram is not permitted by the specified access list, the related debugging output is suppressed.
detailed	(Optional) Displays debug information in a detailed format.
h323	(Optional) Displays H.225/H.245 protocol information.
pptp	(Optional) Displays Point-to-Point Tunneling (PPTP) protocol information.

Defaults

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2	This command was introduced.
12.1(5)T	This command was modified to include the **h323** keyword.

Usage Guidelines

The NAT feature reduces the need for unique, registered IP addresses. It can also save private network administrators from needing to renumber hosts and routers that do not conform to global IP addressing.

Use the **debug ip nat** command to verify the operation of the NAT feature by displaying information about every packet that is translated by the router. The **debug ip nat detailed** command generates a description of each packet considered for translation. This command also outputs information about certain errors or exceptional conditions, such as the failure to allocate a global address. To display messages related to the processing of H.225 signalling and H.245 messages, use the **debug ip nat h323** command.

Caution Because the **debug ip nat** command generates a substantial amount of output, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Examples

The following is sample output from the **debug ip nat** command. In this example, the first two lines show the debugging output produced by a Domain Name System (DNS) request and reply. The remaining lines show the debugging output from a Telnet connection from a host on the inside of the network to a host on the outside of the network. All Telnet packets, except for the first packet, were translated in the fast path, as indicated by the asterisk (*).

```
Router# debug ip nat

NAT: s=192.168.1.95->172.31.233.209, d=172.31.2.132 [6825]
NAT: s=172.31.2.132, d=172.31.233.209->192.168.1.95 [21852]
NAT: s=192.168.1.95->172.31.233.209, d=172.31.1.161 [6826]
NAT*: s=172.31.1.161, d=172.31.233.209->192.168.1.95 [23311]
NAT*: s=192.168.1.95->172.31.233.209, d=172.31.1.161 [6827]
NAT*: s=192.168.1.95->172.31.233.209, d=172.31.1.161 [6828]
NAT*: s=172.31.1.161, d=172.31.233.209->192.168.1.95 [23313]
NAT*: s=172.31.1.161, d=172.31.233.209->192.168.1.95 [23325]
```

Table 85 describes the significant fields shown in the display.

Table 85 debug ip nat Field Descriptions

Field	Description
NAT:	Indicates that the packet is being translated by the NAT feature. An asterisk (*) indicates that the translation is occurring in the fast path. The first packet in a conversation always goes through the slow path (that is, they are process switched). The remaining packets go through the fast path if a cache entry exists.
s=192.168.1.95—172.31.233.209	Source address of the packet and how it is being translated.
d=172.31.2.132	Destination address of the packet.
[6825]	IP identification number of the packet. Might be useful in the debugging process to correlate with other packet traces from protocol analyzers.

The following is sample output from the **debug ip nat detailed** command. In this example, the first two lines show the debugging output produced by a DNS request and reply. The remaining lines show the debugging output from a Telnet connection from a host on the inside of the network to a host on the outside of the network. In this example, the inside host 192.168.1.95 was assigned the global address 172.31.233.193.

```
Router# debug ip nat detailed

NAT: i: udp (192.168.1.95, 1493) -> (172.31.2.132, 53) [22399]
NAT: o: udp (172.31.2.132, 53) -> (172.31.233.193, 1493) [63671]
NAT*: i: tcp (192.168.1.95, 1135) -> (172.31.2.75, 23) [22400]
NAT*: o: tcp (172.31.2.75, 23) -> (172.31.233.193, 1135) [22002]
NAT*: i: tcp (192.168.1.95, 1135) -> (172.31.2.75, 23) [22401]
NAT*: i: tcp (192.168.1.95, 1135) -> (172.31.2.75, 23) [22402]
NAT*: o: tcp (172.31.2.75, 23) -> (172.31.233.193, 1135) [22060]
NAT*: o: tcp (172.31.2.75, 23) -> (172.31.233.193, 1135) [22071]
```

Table 86 describes the significant fields shown in the display.

Table 86 debug ip nat detailed Field Descriptions

Field	Description
NAT:	Indicates that the packet is being translated by the NAT feature. An asterisk (*) indicates that the translation is occurring in the fast path.
i:	Indicates that the packet is moving from a host inside the network to one outside the network.
o:	Indicates that the packet is moving from a host outside the network to one inside the network.
udp	Protocol of the packet.
(192.168.1.95, 1493) - (172.31.2.132, 53)	Indicates that the packet is sent from IP address 192.168.1.95, port number 1493 to IP address 172.31.2.132, port number 53.
[22399]	IP identification number of the packet.

The following is sample output from the **debug ip nat h323** command. In this example, an H.323 call is established between two hosts, one host on the inside and the other one on the outside. The debug displays the H.323 messages names that NAT recognizes and the embedded IP addresses contained in those messages.

```
Router# debug ip nat h323

NAT:H225:[0] processing a Setup message
NAT:H225:[0] found Setup sourceCallSignalling
NAT:H225:[0] fix TransportAddress addr=192.168.122.50 port=11140
NAT:H225:[0] found Setup fastStart
NAT:H225:[0] Setup fastStart PDU length:18
NAT:H245:[0] processing OpenLogicalChannel message, forward channel
number 1
NAT:H245:[0] found OLC forward mediaControlChannel
NAT:H245:[0] fix TransportAddress addr=192.168.122.50 port=16517
NAT:H225:[0] Setup fastStart PDU length:29
NAT:H245:[0] processing OpenLogicalChannel message, forward channel
number 1
NAT:H245:[0] found OLC reverse mediaChannel
NAT:H245:[0] fix TransportAddress addr=192.168.122.50 port=16516
NAT:H245:[0] found OLC reverse mediaControlChannel
NAT:H245:[0] fix TransportAddress addr=192.168.122.50 port=16517
NAT:H225:[1] processing an Alerting  message
NAT:H225:[1] found Alerting fastStart
NAT:H225:[1] Alerting fastStart PDU length:25
NAT:H245:[1] processing OpenLogicalChannel message, forward channe
```

Table 87 describes the significant fields shown in the display.

Table 87 debug ip nat h323 Field Descriptions

Field	Description
NAT:	Indicates that the packet is being translated by the NAT feature.
H.225/H.245:	Protocol of the packet.
[1]	Indicates that the packet is moving from a host inside the network to one outside the network.
[0]	Indicates that the packet is moving from a host outside the network to one inside the network.

debug ip ospf events

To display information on Open Shortest Path First (OSPF)-related events, such as adjacencies, flooding information, designated router selection, and shortest path first (SPF) calculation, use the **debug ip ospf events** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip ospf events

no debug ip ospf events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ip ospf events

OSPF:hello with invalid timers on interface Ethernet0
hello interval received 10 configured 10
net mask received 255.255.255.0 configured 255.255.255.0
dead interval received 40 configured 30
```

The **debug ip ospf events** output shown might appear if any of the following situations occurs:

- The IP subnet masks for routers on the same network do not match.
- The OSPF hello interval for the router does not match that configured for a neighbor.
- The OSPF dead interval for the router does not match that configured for a neighbor.

If a router configured for OSPF routing is not seeing an OSPF neighbor on an attached network, perform the following tasks:

- Make sure that both routers have been configured with the same IP mask, OSPF hello interval, and OSPF dead interval.
- Make sure that both neighbors are part of the same area type.

In the following example line, the neighbor and this router are not part of a stub area (that is, one is a part of a transit area and the other is a part of a stub area, as explained in RFC 1247):

```
OSPF: hello packet with mismatched E bit
```

Related Commands

Command	Description
debug ip pgm host	Displays information about each OSPF packet received.

debug ip ospf mpls traffic-eng advertisements

To print information about traffic engineering advertisements in OSPF link state advertisement (LSA) messages, use the **debug ip ospf mpls traffic-eng advertisements** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip ospf mpls traffic-eng advertisements

no debug ip ospf mpls traffic-eng advertisements

Syntax Description

This command has no arguments or keywords

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information about traffic engineering advertisements is printed in OSPF LSA messages:

```
debug ip ospf mpls traffic-eng advertisements

OSPF:IGP delete router node 10.106.0.6 fragment 0 with 0 links
      TE Router ID 10.106.0.6
OSPF:IGP update router node 10.110.0.10 fragment 0 with 0 links
      TE Router ID 10.110.0.10
OSPF:MPLS announce router node 10.106.0.6 fragment 0 with 1 links
      Link connected to Point-to-Point network
      Link ID :10.110.0.10
      Interface Address :10.1.0.6
      Neighbor Address :10.1.0.10
      Admin Metric :10
      Maximum bandwidth :1250000
      Maximum reservable bandwidth :625000
      Number of Priority :8
      Priority 0 :625000       Priority 1 :625000
      Priority 2 :625000       Priority 3 :625000
      Priority 4 :625000       Priority 5 :625000
      Priority 6 :625000       Priority 7 :625000
      Affinity Bit :0x0
```

Table 88 describes the significant fields shown in the display.

Table 88 debug ip ospf mpls traffic-eng advertisements Field Descriptions

Field	Description
Link ID	Index of the link being described.
Interface Address	Address of the interface.
Neighbor Address	Address of the neighbor.
Admin Metric	Administrative weight associated with this link.
Maximum bandwidth	Bandwidth capacity of the link (kbps).
Maximum reservable bandwidth	Amount of reservable bandwidth on this link.
Number of Priority	Number of priority levels for which bandwidth is advertised.
Priority	Bandwidth available at indicated priority level.
Affinity Bit	Attribute flags of the link that are being flooded.

debug ip ospf packet

To display information about each Open Shortest Path First (OSPF) packet received, use the **debug ip ospf packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip ospf packet

no debug ip ospf packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ip ospf packet

OSPF: rcv. v:2 t:1 l:48 rid:200.0.0.117
      aid:0.0.0.0 chk:6AB2 aut:0 auk:
```

The **debug ip ospf packet** command produces one set of information for each packet received. The output varies slightly depending on which authentication is used. The following is sample output from the **debug ip ospf packet** command when MD5 authentication is used.

```
Router# debug ip ospf packet

OSPF: rcv. v:2 t:1 l:48 rid:200.0.0.116
      aid:0.0.0.0 chk:0 aut:2 keyid:1 seq:0x0
```

Table 89 describes the fields shown in the display.

Table 89 debug ip ospf packet Field Descriptions

Field	Description
v:	OSPF version.
t:	OSPF packet type. Possible packet types follow: • 1—Hello • 2—Data description • 3—Link state request • 4—Link state update • 5—Link state acknowledgment
l:	OSPF packet length in bytes.
rid:	OSPF router ID.
aid:	OSPF area ID.
chk:	OSPF checksum.

Table 89 debug ip ospf packet Field Descriptions (continued)

Field	Description
aut:	OSPF authentication type. Possible authentication types follow: • 0—No authentication • 1—Simple password • 2—MD5
auk:	OSPF authentication key.
keyid:	MD5 key ID.
seq:	Sequence number.

Related Commands

Command	Description
debug ip ospf events	Displays information on OSPF-related events, such as adjacencies, flooding information, designated router selection, and SPF calculation.

debug ip ospf spf statistic

To display statistical information while running the shortest path first algorithm (SPF), use the **debug ip ospf spf statistic** command in privileged EXEC mode. To disable the debugging output, use the **no** form of this command.

debug ip ospf spf statistic

no debug ip ospf spf statistic

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.2(12)	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.0(23)S	This command was integrated into Cisco IOS Release 12.0(23)S.
12.2(12)S	This command was integrated into Cisco IOS Release 12.2(12)S.

Usage Guidelines

The **debug ip ospf spf statistic** command displays the SPF calculation times in milliseconds, the node count, and a time stamp.

Examples

The following is sample output from the **debug ip ospf spf statistic** command:

```
Router# debug ip ospf spf statistic

00:05:59: OSPF: Begin SPF at 359.216ms, process time 60ms
00:05:59:       spf_time 00:05:59.216, wait_interval 0s
00:05:59: OSPF: End SPF at 359.216ms, Total elapsed time 0ms
00:05:59:       Intra: 0ms, Inter: 0ms, External: 0ms
00:05:59:       R: 4, N: 2, Stubs: 1
00:05:59:       SN: 1, SA: 0, X5: 1, X7: 0
00:05:59:       SPF suspends: 0 intra, 1 total
```

Table 90 describes the fields shown in the display.

Table 90 debug ip ospf spf statistic Field Descriptions

Field	Description
Begin SPF at	Absolute time in milliseconds when SPF is started.
process time	Cumulative time since the process has been created.
spf_time	Last time SPF was run or an event has happened to run SPF.
wait_interval	Time waited to run SPF.
End SPF at	Absolute time in milliseconds when SPF had ended.
Total elapsed time	Total time take to run SPF.

Table 90 debug ip ospf spf statistic Field Descriptions (continued)

Field	Description
Intra:	Time taken to process intra-area link-state advertisements (LSAs).
Inter:	Time taken to process interarea LSAs.
External:	Time taken to process external LSAs.
R:	Number of router LSAs.
N:	Number of network LSAs.
Stubs:	Number of stub links.
SN:	Number of summary network LSAs.
SA:	Number of summary LSAs describing autonomous system boundary routers (ASBRs).
X5:	Number of external type 5 LSAs.
X7:	Number of external type 7 LSAs.
SPF suspends: intra	Number of times process is suspended during intra-area SPF run.
total	Total number of times process is suspended during SPF run.

debug ip packet

To display general IP debugging information and IP security option (IPSO) security transactions, use the **debug ip packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip packet [*access-list-number*] [**detail] [dump**]

no debug ip packet [*access-list-number*]

Syntax Description

access-list-number	(Optional) The IP access list number that you can specify. If the datagram is not permitted by that access list, the related debugging output is suppressed. Standard, extended, and expanded access lists are supported. The range of standard and extended access lists is from 1 to 199. The range of expanded access lists is from 1300 to 2699.
detail	(Optional) Displays detailed IP packet debugging information. This information includes the packet types and codes as well as source and destination port numbers.
dump	(Hidden) Displays IP packet debugging information along with raw packet data in hexadecimal and ASCII forms. This keyword can be enabled with individual access lists and also with the **detail** keyword. **Note** The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution notes below, in the usage guidelines, for more specific information.

Command Modes

Privileged EXEC

Usage Guidelines

If a communication session is closing when it should not be, an end-to-end connection problem can be the cause. The **debug ip packet** command is useful for analyzing the messages traveling between the local and remote hosts. IP packet debugging captures the packets that are process switched including received, generated and forwarded packets. IP packets that are switched in the fast path are not captured.

IPSO security transactions include messages that describe the cause of failure each time a datagram fails a security test in the system. This information is also sent to the sending host when the router configuration allows it.

Caution Because the **debug ip packet** command generates a substantial amount of output and uses a substantial amount of system resources, this command should be used with caution in production networks. It should only be enabled when traffic on the IP network is low, so other activity on the system is not adversely affected. Enabling the **detail** and **dump** keywords use the highest level of system resources of the available configuration options for this command, so a high level of caution should be applied when enabling either of these keywords.

> **Caution** The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. Because of the risk of using significant CPU utilization, the dump keyword is hidden from the user and cannot be seen using the “?” prompt. The length of the displayed packet information may exceed the actual packet length and include additional padding bytes that do not belong to the IP packet. Also note that the beginning of a packet may start at different locations in the dump output depending on the specific router, interface type, and packet header processing that may have occurred before the output is displayed.

Examples

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

```
debug ip packet

IP packet debugging is on

IP: s=172.69.13.44 (Fddi0), d=10.125.254.1 (Serial2), g=172.69.16.2, forward
IP: s=172.69.1.57 (Ethernet4), d=10.36.125.2 (Serial2), g=172.69.16.2, forward
IP: s=172.69.1.6 (Ethernet4), d=255.255.255.255, rcvd 2
IP: s=172.69.1.55 (Ethernet4), d=172.69.2.42 (Fddi0), g=172.69.13.6, forward
IP: s=172.69.89.33 (Ethernet2), d=10.130.2.156 (Serial2), g=172.69.16.2, forward
IP: s=172.69.1.27 (Ethernet4), d=172.69.43.126 (Fddi1), g=172.69.23.5, forward
IP: s=172.69.1.27 (Ethernet4), d=172.69.43.126 (Fddi0), g=172.69.13.6, forward
IP: s=172.69.20.32 (Ethernet2), d=255.255.255.255, rcvd 2
IP: s=172.69.1.57 (Ethernet4), d=10.36.125.2 (Serial2), g=172.69.16.2, access denied
```

The output shows two types of messages that the **debug ip packet** command can produce; the first line of output describes an IP packet that the router forwards, and the third line of output describes a packet that is destined for the router. In the third line of output, rcvd 2 indicates that the router decided to receive the packet.

Table 91 describes the significant fields shown in the output.

Table 91 debug ip packet Field Descriptions

Field	Description
IP:	Indicates that this is an IP packet.
s=172.69.13.44 (Fddi0)	Indicates the source address of the packet and the name of the interface that received the packet.
d=10.125.254.1 (Serial2)	Indicates the destination address of the packet and the name of the interface (in this case, S2) through which the packet is being sent out on the network.
g=172.69.16.2	Indicates the address of the next-hop gateway.
forward	Indicates that the router is forwarding the packet. If a filter denies a packet, “access denied” replaces “forward,” as shown in the last line of output.

The following is sample output from the **debug ip packet** command enabled with the **detail** keyword:

```
debug ip packet detail

IP packet debugging is on (detailed)

001556: 19:59:30: CEF: Try to CEF switch 10.4.9.151 from FastEthernet0/0
```

```
001557: 19:59:30: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.151 (FastEthernet03
001558: 19:59:30:     TCP src=179, dst=11001, seq=3736598846, ack=2885081910, wH
001559: 20:00:09: CEF: Try to CEF switch 10.4.9.151 from FastEthernet0/0
001560: 20:00:09: IP: s=10.4.9.4 (FastEthernet0/0), d=10.4.9.151 (FastEthernet03
001561: 20:00:09:     TCP src=179, dst=11000, seq=163035693, ack=2948141027, wiH
001562: 20:00:14: CEF: Try to CEF switch 10.4.9.151 from FastEthernet0/0
001563: 20:00:14: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.151 (FastEthernet03
001564: 20:00:14:     ICMP type=8, code=0
001565: 20:00:14: IP: s=10.4.9.151 (local), d=10.4.9.6 (FastEthernet0/0), len 1g
001566: 20:00:14:     ICMP type=0, code=0
```

The format of the output with **detail** keyword provides additional information, such as the packet type, code, some field values, and source and destination port numbers.

Table 92 describes the significant fields shown in the output.

Table 92 debug ip packet detail Field Descriptions

Field	Description
CEF:	Indicates that the IP packet is being processed by CEF.
IP:	Indicates that this is an IP packet.
s=10.4.9.6 (FastEthernet0/0)	Indicates the source address of the packet and the name of the interface that received the packet.
d=10.4.9.151 (FastEthernet03)	Indicates the destination address of the packet and the name of the interface through which the packet is being sent out on the network.
TCP src=	Indicates the source TCP port number.
dst=	Indicates the destination TCP port number.
seq=	Value from the TCP packet sequence number field./
ack=	Value from the TCP packet acknowledgement field.
ICMP type=	Indicates ICMP packet type.
code=	Indicates ICMP return code.

The following is sample output from the **debug ip packet** command enabled with the **dump** keyword:

```
debug ip packet dump

IP packet debugging is on (detailed) (dump)

21:02:42: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.4 (FastEthernet0/0), len 13
07003A00:                        0005 00509C08            ...P..
07003A10: 0007855B 4DC00800 45000064 001E0000  ...[M@..E..d....
07003A20: FE019669 0A040906 0A040904 0800CF7C  ~..i..........O|
07003A30: 0D052678 00000000 0A0B7145 ABCDABCD  ..&x......qE+M+M
07003A40: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
07003A50: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
07003A60: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
07003A70: ABCDABCD ABCDABCD ABCDABCD           +M+M+M+M+M+M
21:02:42: IP: s=10.4.9.4 (local), d=10.4.9.6 (FastEthernet0/0), len 100, sending
07003A00:                        0005 00509C08            ...P..
07003A10: 0007855B 4DC00800 45000064 001E0000  ...[M@..E..d....
07003A20: FF019569 0A040904 0A040906 0000D77C  ...i..........W|
07003A30: 0D052678 00000000 0A0B7145 ABCDABCD  ..&x......qE+M+M
07003A40: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
07003A50: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
07003A60: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
```

```
07003A70: ABCDABCD ABCDABCD ABCDABCD          +M+M+M+M+M+M
21:02:42: CEF: Try to CEF switch 10.4.9.4 from FastEthernet0/0
21:02:42: IP: s=10.4.9.6 (FastEthernet0/0), d=10.4.9.4 (FastEthernet0/0), len 13
07003380:                      0005 00509C08            ...P..
07003390: 0007855B 4DC00800 45000064 001F0000  ...[M@..E..d....
070033A0: FE019668 0A040906 0A040904 0800CF77  ~..h..........Ow
070033B0: 0D062678 00000000 0A0B7149 ABCDABCD  ..&x......qI+M+M
070033C0: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
070033D0: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
070033E0: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
070033F0: ABCDABCD ABCDABCD ABCDABCD          +M+M+M+M+M+M
```

Note The **dump** keyword is not fully supported and should be used only in collaboration with Cisco Technical Support. See the caution in the usage guidelines section of this command reference page for more specific information.

The output from the **debug ip packet** command, when the **dump** keyword is enabled, provides raw packet data in hexadecimal and ASCII forms. This addtional output is displayed in addition to the standard output. The **dump** keyword can be used with all of the available configuration options of this command.

Table 93 describes the standard output fields shown.

Table 93 debug ip packet dump Field Descriptions

Field	Description
IP:	Indicates that this is an IP packet.
s=10.4.9.6 (FastEthernet0/0)	Indicates the source address of the packet and the name of the interface that received the packet.
d=10.4.9.4 (FastEthernet0/0) len 13	Indicates destination address and length of the packet and the name of the interface through which the packet is being sent out on the network.
sending	Indicates that the router is sending the packet.

The calculation on whether to send a security error message can be somewhat confusing. It depends upon both the security label in the datagram and the label of the incoming interface. First, the label contained in the datagram is examined for anything obviously wrong. If nothing is wrong, assume the datagram to be correct. If something is wrong, the datagram is treated as *unclassified genser*. Then the label is compared with the interface range, and the appropriate action is taken, as Table 94 describes.

Table 94 Security Actions

Classification	Authorities	Action Taken
Too low	Too low	No Response
	Good	No Response
	Too high	No Response

Table 94 Security Actions (continued)

Classification	Authorities	Action Taken
In range	Too low	No Response
	Good	Accept
	Too high	Send Error
Too high	Too low	No Response
	In range	Send Error
	Too high	Send Error

The security code can only generate a few types of Internet Control Message Protocol (ICMP) error messages. The only possible error messages and their meanings follow:

- ICMP Parameter problem, code 0—Error at pointer
- ICMP Parameter problem, code 1—Missing option
- ICMP Parameter problem, code 2—See Note that follows
- ICMP Unreachable, code 10—Administratively prohibited

Note The message "ICMP Parameter problem, code 2" identifies a specific error that occurs in the processing of a datagram. This message indicates that the router received a datagram containing a maximum length IP header but no security option. After being processed and routed to another interface, it is discovered that the outgoing interface is marked with "add a security label." Because the IP header is already full, the system cannot add a label and must drop the datagram and return an error message.

When an IP packet is rejected due to an IP security failure, an audit message is sent via Department of Defense Intelligence Information System Network Security for Information Exchange (DNSIX) Network Address Translation (NAT). Also, any **debug ip packet** output is appended to include a description of the reason for rejection. This description can be any of the following:

- No basic
- No basic, no response
- Reserved class
- Reserved class, no response
- Class too low, no response
- Class too high
- Class too high, bad authorities, no response
- Unrecognized class
- Unrecognized class, no response
- Multiple basic
- Multiple basic, no response
- Authority too low, no response
- Authority too high

- Compartment bits not dominated by maximum sensitivity level
- Compartment bits do not dominate minimum sensitivity level
- Security failure: extended security disallowed
- NLESO source appeared twice
- ESO source not found
- Postroute, failed xfc out
- No room to add IPSO

debug ip pgm host

To display debug messages for the PGM Host feature, use the **debug ip pgm host** privileged EXEC command. To disable PGM Host debugging output, use the **no** form of this command.

debug ip pgm host [**data** | **nak** | **spm**]

no debug ip pgm host [**data** | **nak** | **spm**]

Syntax Description

data	(Optional) Enables debugging for Pragmatic General Multicast (PGM) sent (ODATA) and re-sent (RDATA) data packets.
nak	(Optional) Enables debugging for PGM negative acknowledgment (NAK) data packets, NAK confirmation (NCF) data packets, and Null NAK data packets.
spm	(Optional) Enables debugging for PGM source path messages (SPMs).

Defaults

Debugging for PGM Host is not enabled. If the **debug ip pgm host** command is used with no additional keywords, debugging is enabled for all PGM Host message types.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following example shows output for the **debug ip pgm host** command:

```
Router# debug ip pgm host

Host SPM debugging is on
Host NAK/NCF debugging is on
Host ODATA/RDATA debugging is on
```

The following example shows output of the **debug ip pgm host** command when the **data** keyword is used.

```
Router# debug ip pgm host data

02:50:23:PGM Host:Received ODATA from 10.0.30.2 to 224.3.3.3 (74 bytes)
02:50:23:     ODATA TSI 00000A001E02-0401 data-dport BBBB csum 9317 tlen 74
02:50:23:     tsqn         31 dsqn         39
```

The following example shows output of the **debug ip pgm host** command when the **nak** keyword is used. In the following example, the host sends a NAK to the source for a missing packet and the source returns an NCF to the host followed by an RDATA data packet.

```
Router# debug ip pgm host nak
```

```
02:50:24:PGM Host:Sending NAK from 10.0.32.2 to 10.0.32.1 (36 bytes)
02:50:24:     NAK TSI 00000A001E02-0401 data-dport BBBB csum 04EC tlen 36
02:50:24:     dsqn         38 data source 10.0.30.2 group 224.3.3.3

02:50:24:PGM Host:Received NCF from 10.0.30.2 to 224.3.3.3 (36 bytes)
02:50:24:     NCF TSI 00000A001E02-0401 data-dport BBBB csum 02EC tlen 36
02:50:24:     dsqn         38 data source 10.0.30.2 group 224.3.3.3

02:50:24:PGM Host:Received RDATA from 10.0.30.2 to 224.3.3.3 (74 bytes)
02:50:24:     RDATA TSI 00000A001E02-0401 data-dport BBBB csum 9218 tlen 74
02:50:24:     tsqn         31 dsqn         38
```

The following example shows output of the **debug ip pgm host** command with the **spm** keyword:

```
Router# debug ip pgm host spm

02:49:39:PGM Host:Received SPM from 10.0.30.2 to 224.3.3.3 (36 bytes)
02:49:39:     SPM TSI 00000A001E02-0401 data-dport BBBB csum EA08 tlen 36
02:49:39:     dsqn        980 tsqn         31 lsqn         31   NLA 10.0.32.1
```

Related Commands

Command	Description
clear ip pgm host	Resets PGM Host connections to their default values and clears traffic statistics.
ip pgm host	Enables the PGM Host feature.
show ip pgm host defaults	Displays the default values for PGM Host traffic.
show ip pgm host sessions	Displays open PGM Host traffic sessions.
show ip pgm host traffic	Displays PGM Host traffic statistics.

debug ip pgm router

To display debug messages for PGM, use the **debug ip pgm router** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug ip pgm router [**spm** | **nak** | **data**]

no debug ip pgm router [**spm** | **nak** | **data**]

Syntax Description

spm	(Optional) Enables debugging for Source Path Messages (SPMs).
nak	(Optional) Enables debugging for negative acknowledgments (NAKs), NAK confirmations (NCFs), and Null NAKs.
data	(Optional) Enables debugging for Retransmissions (RDATA).

Defaults

Debugging for PGM is is not enabled. If the **debug ip pgm router** command is used with no additional keywords, debugging is enabled for all PGM message types.

Command History

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

Examples

The following example shows output of the **debug ip pgm router** command:

```
Router# debug ip pgm router

SPM debugging is on
NAK/NNAK/NCF debugging is on
RDATA debugging is on
```

The following example shows output of the **debug ip pgm router** command with the **spm** keyword:

```
Router# debug ip pgm router spm

PGM: Received SPM on Ethernet1/0/5 from 10.7.0.200 to 227.7.7.7 (52 bytes)
     SPM TSI 0A0700C85555-1000 data-dport 1001 csum CCCC tlen 52
     dsqn 3758096779 tsqn       1954 isqn       1979 lsqn       1990
     NLA 10.7.0.200
     SPM from source/RPF-neighbour 10.7.0.200 for 10.7.0.200 (SPT)
     Forwarded SPM from 10.7.0.200 to 227.7.7.7
```

The following is a debug message for a selective SPM:

```
Router# debug ip pgm router spm
PGM: Received SPM on Ethernet1/0/5 from 10.7.0.200 to 234.4.3.2 (52 bytes)
     SPM TSI 0A0700C85555-2000 data-dport 2001 csum CCCC tlen 52    Options  P N O
     dsqn 3758096768 tsqn       1986 isqn       1994 lsqn       2006
     NLA 10.7.0.200
     SPM from source/RPF-neighbour 10.7.0.200 for 10.7.0.200 (SPT)
     Forwarded SPM from 10.7.0.200 to 227.7.7.7
```

The "P N O" flags indicate which options are present in this packet:

- "P" indicates that this is a parity packet.
- "N" indicates that options are network significant.
- "O" indicates that options are present.

The following example shows output of the **debug ip pgm router** command with the **nak** keyword:

```
Router# debug ip pgm router nak

PGM: Received NAK on Ethernet1/0/0 from 10.1.0.4 to 10.1.0.2 (36 bytes)
     NAK TSI 0A0700C85555-1000 data-dport 1001 csum CCCC tlen 36
     dsqn       1990 data source 10.7.0.200 group 227.7.7.7
     NAK unicast routed to RPF neighbour 10.4.0.1
     Forwarding NAK from 10.1.0.4 to 10.4.0.1 for 10.7.0.200
PGM: Received NCF on Ethernet1/0/5 from 10.7.0.200 to 227.7.7.7 (36 bytes)
     NCF TSI 0A0700C85555-1000 data-dport 1001 csum CACC tlen 36
     dsqn       1990 data source 10.7.0.200 group 227.7.7.7
     NAK retx canceled for TSI 0A0700C85555-1000 dsqn       1990
     NAK elimination started for TSI 0A0700C85555-1000 dsqn       1990
PGM: Received NCF on Ethernet1/0/5 from 10.7.0.200 to 227.7.7.7 (36 bytes)
     NCF TSI 0A0700C85555-1000 data-dport 1001 csum CACC tlen 36
     dsqn       1991 data source 10.7.0.200 group 227.7.7.7
     No NAK retx outstanding for TSI 0A0700C85555-1000 dsqn       1991
     NAK anticipated for TSI 0A0700C85555-1000 dsqn       1991
```

The following example shows output of the **debug ip pgm router** command with the **data** keyword. The debug message is for an RDATA packet for which the router has only anticipated state, sqn 1991. Because it did not actually get a NAK, this RDATA is not forwarded by the PGM router.

```
Router# debug ip pgm router data

PGM: Received RDATA on Ethernet1/0/5 from 10.7.0.200 to 227.7.7.7 (70 bytes)
     RDATA TSI 0A0700C85555-1000 data-dport 1001 csum CCCC tlen 32
     tsqn       1954 dsqn       1990
     Marking Ethernet1/0/0 for forwarding
     Marking Serial5/0 for skipping
     Forwarded RDATA from 10.7.0.200 to 227.7.7.7

Debug message for RDATA packet corresponding to a NAK for sqn
1990. Since the NAK was received on Ethernet1/0/0, RDATA is forwarded
out only that interface and another interface in the multicast olist
Serial5/0 is skipped.

PGM: Received RDATA on Ethernet1/0/5 from 10.7.0.200 to 227.7.7.7 (70 bytes)
     RDATA TSI 0A0700C85555-1000 data-dport 1001 csum CCCC tlen 32
     tsqn       1954 dsqn       1991
     Eliminated RDATA (null oif) from 10.7.0.200 to 227.7.7.7
```

Related Commands

Command	Description
debug ip pgm router	Clears PGM traffic statistics.
ip pgm router	Enables the PGM Router Assist feature for the interface.
show ip pgm router	Displays PGM traffic statistics and TSI state.

debug ip pim

To display Protocol Independent Multicast (PIM) packets received and sent, and to display PIM-related events, use the **debug ip pim** privileged EXEC command. To disable the debugging output, use the **no** form of this command.

debug ip pim [*group* | **df** [*rp-address*]]

no debug ip pim [*group* | **df** [*rp-address*]]

Syntax Description

group	(Optional) The group name or address to monitor the packet activity of a single group.
df	(Optional) When bidir-PIM is used, displays all designated forwarder (DF) election messages.
rp-address	(Optional) The rendezvous point (RP) IP address.

Defaults

All PIM packets are displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
10.2	This command was introduced.
12.1(2)T	The **df** keyword was added.

Usage Guidelines

PIM uses Internet Group Management Protocol (IGMP) packets to communicate between routers and advertise reachability information.

Use this command with the **debug ip igmp** and **debug ip mrouting** commands to observe additional multicast routing information.

Examples

The following is sample output from the **debug ip pim** command:

```
Router# debug ip pim 224.2.0.1

PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Received Join/Prune on Tunnel0 from 10.3.84.1
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Received RP-Reachable on Ethernet1 from 172.69.20.31
PIM: Update RP expiration timer for 224.2.0.1
PIM: Forward RP-reachability packet for 224.2.0.1 on Tunnel0
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Prune-list (10.221.196.51/32, 224.2.0.1)
PIM: Set join delay timer to 2 seconds for (10.221.0.0/16, 224.2.0.1) on Ethernet1
PIM: Received Join/Prune on Ethernet1 from 172.69.37.6
```

```
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Received Join/Prune on Tunnel0 from 10.3.84.1
PIM: Join-list: (*, 224.2.0.1) RP 172.69.20.31
PIM: Add Tunnel0 to (*, 224.2.0.1), Forward state
PIM: Join-list: (10.0.0.0/8, 224.2.0.1)
PIM: Add Tunnel0 to (10.0.0.0/8, 224.2.0.1), Forward state
PIM: Join-list: (10.4.0.0/16, 224.2.0.1)
PIM: Prune-list (172.69.84.16/28, 224.2.0.1) RP-bit set RP 172.69.84.16
PIM: Send Prune on Ethernet1 to 172.69.37.6 for (172.69.84.16/28, 224.2.0.1), RP
PIM: For RP, Prune-list: 10.9.0.0/16
PIM: For RP, Prune-list: 10.16.0.0/16
PIM: For RP, Prune-list: 10.49.0.0/16
PIM: For RP, Prune-list: 10.84.0.0/16
PIM: For RP, Prune-list: 10.146.0.0/16
PIM: For 10.3.84.1, Join-list: 172.69.84.16/28
PIM: Send periodic Join/Prune to RP via 172.69.37.6 (Ethernet1)
```

The following lines appear periodically when PIM is running in sparse mode and indicate to this router the multicast groups and multicast sources in which other routers are interested:

```
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
PIM: Received Join/Prune on Ethernet1 from 172.69.37.33
```

The following lines appear when an RP message is received and the RP timer is reset. The expiration timer sets a checkpoint to make sure the RP still exists; otherwise a new RP must be discovered.

```
PIM: Received RP-Reachable on Ethernet1 from 172.69.20.31
PIM: Update RP expiration timer for 224.2.0.1
PIM: Forward RP-reachability packet for 224.2.0.1 on Tunnel0
```

The prune message in the following line states that this router is not interested in the source address information. This message tells an upstream router to stop forwarding multicast packets from this source.

```
PIM: Prune-list (10.221.196.51/32, 224.2.0.1)
```

In the following line, a second router on the network wants to override the prune message that the upstream router just received. The timer is set at a random value so that if additional routers on the network still want to receive multicast packets for the group, only one will actually send the message. The other routers will receive the join message and then suppress sending their own message.

```
PIM: Set join delay timer to 2 seconds for (10.221.0.0/16, 224.2.0.1) on Ethernet1
```

In the following line, a join message is sent toward the RP for all sources:

```
PIM: Join-list: (*, 224.2.0.1) RP 172.69.20.31
```

In the following lines, the interface is being added to the outgoing interface (OIF) of the (*, G) and (S, G) mroute table entry so that packets from the source will be forwarded out that particular interface:

```
PIM: Add Tunnel0 to (*, 224.2.0.1), Forward state
PIM: Add Tunnel0 to (10.0.0.0/8, 224.2.0.1), Forward state
```

The following line appears in sparse mode only. There are two trees on which data may be received: the RP tree and the source tree. In dense mode there is no RP. After the source and the receiver have discovered one another at the RP, the first hop router for the receiver will usually join to the source tree rather than the RP tree.

```
PIM: Prune-list (172.69.84.16/28, 224.2.0.1) RP-bit set RP 172.69.84.16
```

The send prune message in the next line shows that a router is sending a message to a second router saying that the first router no longer wants to receive multicast packets for the (S, G). The RP at the end of the message indicates that the router is pruning the RP tree and is most likely joining the source tree, although the router may not have downstream members for the group or downstream routers with members of the group. The output shows the specific sources from which this router no longer wants to receive multicast.

```
PIM: Send Prune on Ethernet1 to 172.69.37.6 for (172.69.84.16/28, 224.2.0.1), RP
```

The following lines indicate that a prune message is sent toward the RP so that the router can join the source tree rather than the RP tree:

```
PIM: For RP, Prune-list: 10.9.0.0/16
PIM: For RP, Prune-list: 10.16.0.0/16
PIM: For RP, Prune-list: 10.49.0.0/16
```

In the following line, a periodic message is sent toward the RP. The default period is once per minute. Prune and join messages are sent toward the RP or source rather than directly to the RP or source. It is the responsibility of the next hop router to take proper action with this message, such as continuing to forward it to the next router in the tree.

```
PIM: Send periodic Join/Prune to RP via 172.69.37.6 (Ethernet1)
```

Related Commands

Command	Description
debug ip dvmrp	Displays information on DVMRP packets received and sent.
debug ip igmp	Displays IGMP packets received and sent, and displays IGMP host-related events.
debug ip igrp transactions	Displays transaction information on IGRP routing transactions.
debug ip mrouting	Displays changes to the IP multicast routing table.
debug ip sd	Displays all SD announcements received.

debug ip pim atm

To log PIM ATM signalling activity, use the **debug ip pim atm** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip pim atm

no debug ip pim atm

Syntax Description

This command has no arguments or keywords.

Examples

The following sample output shows a new group being created and the router toward the RP opening a new VC. Because there are now two groups on this router, there are two VCs open, as reflected by the "current count."

The following is sample output from the **debug ip pim atm** command:

```
Router# debug ip pim atm

Jan 28 19:05:51: PIM-ATM: Max VCs 200, current count 1
Jan 28 19:05:51: PIM-ATM: Send SETUP on ATM2/0 for 239.254.254.253/171.69.214.43
Jan 28 19:05:51: PIM-ATM: Received CONNECT on ATM2/0 for 239.254.254.253, vcd 19
Jan 28 19:06:35: PIM-ATM: Max VCs 200, current count 2
```

Table 95 describes the significant fields in the output.

Table 95 debug ip pim atm Field Descriptions

Field	Description
Jan 28 19:05:51	Current date and time (in hours:minutes:seconds).
PIM-ATM	Indicates what PIM is doing to set up or monitor an ATM connection (vc).
current count	Current number of open virtual circuits.

The resulting **show ip mroute** output follows:

```
Router# show ip mroute 239.254.254.253

IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 239.254.254.253), 00:00:04/00:02:53, RP 171.69.214.50, flags: S
  Incoming interface: Ethernet1/1, RPF nbr 171.69.214.50
  Outgoing interface list:
    ATM2/0, VCD 19, Forward/Sparse-Dense, 00:00:04/00:02:52
```

debug ip pim auto-rp

To display the contents of each Protocol Independent Multicast (PIM) packet used in the automatic discovery of group-to-rendezvous point (RP) mapping and the actions taken on the address-to-RP mapping database, use the **debug ip pim auto-rp** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip pim auto-rp

no debug ip pim auto-rp

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug ip pim auto-rp** command:

```
Router# debug ip pim auto-rp

Auto-RP: Received RP-announce, from 172.16.214.66, RP_cnt 1, holdtime 180 secs
Auto-RP:  update (192.168.248.0/24, RP:172.16.214.66)
Auto-RP: Build RP-Discovery packet
Auto-RP:  Build mapping (192.168.248.0/24, RP:172.16.214.66),
Auto-RP:  Build mapping (192.168.250.0/24, RP:172.16.214.26).
Auto-RP:  Build mapping (192.168.254.0/24, RP:172.16.214.2).
Auto-RP: Send RP-discovery packet (3 RP entries)
Auto-RP: Build RP-Announce packet for 172.16.214.2
Auto-RP:  Build announce entry for (192.168.254.0/24)
Auto-RP: Send RP-Announce packet, IP source 172.16.214.2, ttl 8
```

The first two lines show a packet received from 172.16.214.66 announcing that it is the RP for the groups in 192.168.248.0/24. This announcement contains one RP address and is valid for 180 seconds. The RP-mapping agent then updates its mapping database to include the new information.

```
Auto-RP: Received RP-announce, from 172.16.214.66, RP_cnt 1, holdtime 180 secs
Auto-RP:  update (192.168.248.0/24, RP:172.16.214.66)
```

In the next five lines, the router creates an RP-discovery packet containing three RP mapping entries. The packet is sent to the well-known CISCO-RP-DISCOVERY group address (224.0.1.40).

```
Auto-RP: Build RP-Discovery packet
Auto-RP:  Build mapping (192.168.248.0/24, RP:172.16.214.66),
Auto-RP:  Build mapping (192.168.250.0/24, RP:172.16.214.26).
Auto-RP:  Build mapping (192.168.254.0/24, RP:172.16.214.2).
Auto-RP: Send RP-discovery packet (3 RP entries)
```

The final three lines show the router announcing that it intends to be an RP for the groups in 192.168.254.0/24. Only routers inside the scope ttl 8 receive the advertisement and use the RP for these groups.

```
Auto-RP: Build RP-Announce packet for 172.16.214.2
Auto-RP:  Build announce entry for (192.168.254.0/24)
Auto-RP: Send RP-Announce packet, IP source 172.16.214.2, ttl 8
```

The following is sample output from the **debug ip pim auto-rp** command when a router receives an update. In this example, the packet contains three group-to-RP mappings, which are valid for 180 seconds. The RP-mapping agent then updates its mapping database to include the new information.

```
Router# debug ip pim auto-rp
```

```
Auto-RP: Received RP-discovery, from 172.16.214.17, RP_cnt 3, holdtime 180 secs
Auto-RP:  update (192.168.248.0/24, RP:172.16.214.66)
Auto-RP:  update (192.168.250.0/24, RP:172.16.214.26)
Auto-RP:  update (192.168.254.0/24, RP:172.16172.16.214.2)
```

debug ip policy

To display IP policy routing packet activity, use the **debug ip policy** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip policy [*access-list-name*]

no debug ip policy [*access-list-name*]

Syntax Description

access-list-name	(Optional) The name of the access list. Displays packets permitted by the access list that are policy routed in process level, CEF, and DCEF (with NetFlow enabled or disabled). If no access list is specified, information about all policy-matched and policy-routed packets is displayed.

Command History

Release	Command
12.0(3)T	This command was introduced.

Usage Guidelines

After you configure IP policy routing with the **ip policy** and **route-map** commands, use the **debug ip policy** command to ensure that the IP policy is configured correctly.

Policy routing looks at various parts of the packet and then routes the packet based on certain user-defined attributes in the packet.

The **debug ip policy** command helps you determine what policy routing is following. It displays information about whether a packet matches the criteria, and if so, the resulting routing information for the packet.

Caution Because the **debug ip policy** command generates a substantial amount of output, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Examples

The following is sample output of the **debug ip policy** command:

```
Router# debug ip policy 3

IP: s=30.0.0.1 (Ethernet0/0/1), d=40.0.0.7, len 100,FIB flow policy match
IP: s=30.0.0.1 (Ethernet0/0/1), d=40.0.0.7, len 100,FIB PR flow accelerated!
IP: s=30.0.0.1 (Ethernet0/0/1), d=40.0.0.7, g=10.0.0.8, len 100, FIB policy routed
```

Table 96 describes the significant fields shown in the display.

Table 96 debug ip policy Field Descriptions

Field	Description
IP: s=	IP source address and interface of the packet being routed.
d=	IP destination address of the packet being routed.
len	Length of the packet.
g=	IP gateway address of the packet being routed.

debug ip rgmp

To log debug messages sent by an RGMP-enabled router, use the **debug ip rgmp** privileged EXEC command. To disable RGMP debugging, use the **no** form of this command.

debug ip rgmp [*group-name* | *group-address*]

no debug ip rgmp

Syntax Description

group-name	(Optional) The name of a specific IP multicast group.
group-address	(Optional) The IP address of a specific IP multicast group.

Defaults

Debugging for RGMP is not enabled. If the **debug ip rgmp** command is used without arguments, debugging is enabled for all RGMP message types.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(10)S	This command was introduced.
12.1(1)E	The command was integrated into Cisco IOS Release 12.1(1)E.
12.1(5)T	The command was integrated into Cisco IOS Release 12.1(5)T.

Examples

The following example shows output for the **debug ip rgmp** command:

```
Router# debug ip rgmp

   RGMP: Sending a Hello packet on Ethernet1/0

   RGMP: Sending a Join packet on Ethernet1/0 for group 224.1.2.3

   RGMP: Sending a Leave packet on Ethernet1/0 for group 224.1.2.3

   RGMP: Sending a Bye packet on Ethernet1/0
```

Related Commands

Command	Description
ip rgmp	Enables the RGMP on IEEE 802.3 Ethernet interfaces.
show ip igmp interface	Displays multicast-related information about an interface.

debug ip rip

To display information on RIP routing transactions, use the **debug ip rip** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip rip

no debug ip rip

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug ip rip** command:

Updates received from this source address

Updates sent to these two destination addresses

```
router# debug ip rip
RIP: received update from 10.89.80.28 on Ethernet0
 10.89.95.0 in 1 hops
 10.89.81.0 in 1 hops
 10.89.66.0 in 2 hops
 172.31.0.0 in 16 hops (inaccessible)
 0.0.0.0 in 7 hop
RIP: sending update to 255.255.255.255 via Ethernet0 (10.89.64.31)
 subnet 10.89.94.0, metric 1
 172.31.0.0 in 16 hops (inaccessible)
RIP: sending update to 255.255.255.255 via Serial1 (10.89.94.31)
 subnet 10.89.64.0, metric 1
 subnet 10.89.66.0, metric 3
 172.31.0.0 in 16 hops (inaccessible)
 default 0.0.0.0, metric 8
```

S2550

The output shows that the router being debugged has received updates from one router at source address 160.89.80.28. That router sent information about five destinations in the routing table update. Notice that the fourth destination address in the update—131.108.0.0—is inaccessible because it is more than 15 hops away from the router sending the update. The router being debugged also sent updates, in both cases to broadcast address 255.255.255.255 as the destination.

The second line is an example of a routing table update. It shows how many hops a given Internet address is from the router.

The entries show that the router is sending updates that are similar, except that the number in parentheses is the source address encapsulated into the IP header.

Examples of additional output that the **debug ip rip** command can generate follow.

Entries such as the following appear at startup or when an event occurs such as an interface making a transition or a user manually clearing the routing table:

```
RIP: broadcasting general request on Ethernet0
RIP: broadcasting general request on Ethernet1
```

An entry such as the following is most likely caused by a malformed packet from the sender:

```
RIP: bad version 128 from 160.89.80.43
```

debug ip routing

To display information on Routing Information Protocol (RIP) routing table updates and route cache updates, use the **debug ip routing** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip routing

no debug ip routing

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ip routing

RT: add 172.25.168.0 255.255.255.0 via 172.24.76.30, igrp metric [100/3020]
RT: metric change to 172.25.168.0 via 172.24.76.30, igrp metric [100/3020]
        new metric [100/2930]
IP: cache invalidation from 0x115248 0x1378A, new version 5736
RT: add 172.26.219.0 255.255.255.0 via 172.24.76.30, igrp metric [100/16200]
RT: metric change to 172.26.219.0 via 172.24.76.30, igrp metric [100/16200]
        new metric [100/10816]
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5737
RT: 172.26.219.0 came out of holddown
RT: garbage collecting entry for 172.26.219.0
IP: cache invalidation from 0x115248 0x1378A, new version 5738
RT: add 172.26.219.0 255.255.255.0 via 172.24.76.30, igrp metric [100/10816]
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5739
RT: 172.26.219.0 came out of holddown
RT: garbage collecting entry for 172.26.219.0
IP: cache invalidation from 0x115248 0x1378A, new version 5740
RT: add 172.26.219.0 255.255.255.0 via 172.24.76.30, igrp metric [100/16200]
RT: metric change to 172.26.219.0 via 172.24.76.30, igrp metric [100/16200]
        new metric [100/10816]
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5741
```

In the following lines, a newly created entry has been added to the IP routing table. The "metric change" indicates that this entry existed previously, but its metric changed and the change was reported by means of IGRP. The metric could also be reported via RIP, OSPF, or another IP routing protocol. The numbers inside the brackets report the administrative distance and the actual metric.

```
RT: add 172.25.168.0 255.255.255.0 via 172.24.76.30, igrp metric [100/3020]
RT: metric change to 172.25.168.0 via 172.24.76.30, igrp metric [100/3020]
        new metric [100/2930]
IP: cache invalidation from 0x115248 0x1378A, new version 5736
```

"Cache invalidation" means that the fast-switching cache was invalidated due to a routing table change. "New version" is the version number of the routing table. When the routing table changes, this number is incriminated. The hexadecimal numbers are internal numbers that vary from version to version and software load to software load.

In the following output, the "holddown" and "cache invalidation" lines are displayed. Most of the distance vector routing protocols use "holddown" to avoid typical problems like counting to infinity and routing loops. If you look at the output of the **show ip protocols** command you will see the timer values for "holddown" and "cache invalidation." "Cache invalidation" corresponds to "came out of holddown." "Delete route" is triggered when a better path comes along. It removes the old inferior path.

```
RT: delete route to 172.26.219.0 via 172.24.76.30, igrp metric [100/10816]
RT: no routes to 172.26.219.0, entering holddown
IP: cache invalidation from 0x115248 0x1378A, new version 5737
RT: 172.26.219.0 came out of holddown
```

debug ip rsvp

To display information about Subnetwork Bandwidth Manager (SBM) message processing, the Designated Subnetwork Bandwidth Manager (DSBM) election process, and standard Resource Reservation Protocol (RSVP)-enabled message processing information, use the **debug ip rsvp** privileged EXEC command. To turn off debugging when you no longer want to display the output, use the **no** form of this command.

debug ip rsvp

no debug ip rsvp

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Usage Guidelines

The **debug ip rsvp** command provides information about messages received, minimal detail about the content of these messages, and information about state transitions. To obtain detailed information about RSVP and SBM, use the **debug ip rsvp detail** command.

Command History

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

Examples

The following example enables output of debug information about SBM message processing, the DSBM election process, and RSVP message processing information on router2:

```
Router# debug ip rsvp

RSVP debugging is on
router2#
*Dec 31 16:42:14.635: RSVP: send I_AM_DSBM message from 145.2.2.150
*Dec 31 16:42:14.635: RSVP: IP to 224.0.0.17 length=88 checksum=C788 Ethernet2)
*Dec 31 16:42:19.635: RSVP: send I_AM_DSBM message from 145.2.2.150
*Dec 31 16:42:19.635: RSVP: IP to 224.0.0.17 length=88 checksum=C788 (Ethernet2)
*Dec 31 16:42:20.823: RSVP: PATH message for 145.5.5.202(Ethernet2) from 145.2.2.1
*Dec 31 16:42:22.163: RSVP: send path multicast about 145.5.5.202 on Ethernet2
*Dec 31 16:42:22.163: RSVP: DSBM mgd segment - sending to ALLSBMADDRESS
*Dec 31 16:42:22.163: RSVP: IP to 224.0.0.17 length=212 checksum=DCAB (Ethernet2)
*Dec 31 16:42:23.955: RSVP: Sending RESV message for 145.5.5.202
*Dec 31 16:42:23.955: RSVP: send reservation to 145.2.2.1 about 145.5.5.202
*Dec 31 16:42:23.955: RSVP: IP to 145.2.2.1 length=108 checksum=1420 (Ethernet2)
*Dec 31 16:42:24.443: RSVP: RESV message for 145.5.5.202 (Ethernet2) from 145.2.2.2
*Dec 31 16:42:24.635: RSVP: send I_AM_DSBM message from 145.2.2.150
*Dec 31 16:42:24.635: RSVP: IP to 224.0.0.17 length=88 checksum=43AF (Ethernet2)
```

Related Commands

Command	Description
debug ip rsvp detail	Displays detailed information about RSVP and SBM.
debug ip rsvp sbm	Displays detailed information about the contents of SMB messages only, and SBM and DSBM state transitions.
ip rsvp dsbm-candidate	Configures an interface as a DSBM candidate.
show ip rsvp sbm	Displays information about SBM configured for a specific RSVP-enabled interface or all RSVP-enabled interfaces on the router.

debug ip rsvp detail

To display detailed information about Resource Reservation Protocol (RSVP)-enabled and Subnetwork Bandwidth Manager (SBM) message processing, use the **debug ip rsvp detail** privileged EXEC command. To turn off debugging when you no longer want to display the output, use the **no** form of this command.

debug ip rsvp detail

no debug ip rsvp detail

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Command History

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

Examples

The following example shows the detailed debug information about RSVP and SBM that is available when you enable debug mode through the **debug ip rsvp detail** command:

```
Router# debug ip rsvp detail

RSVP debugging is on
router2#u
*Dec 31 16:44:29.651: RSVP: send I_AM_DSBM message from 145.2.2.150
*Dec 31 16:44:29.651: RSVP: IP to 224.0.0.17 length=88 checksum=43AF
(Ethernet2)
*Dec 31 16:44:29.651: RSVP: version:1 flags:0000 type:I_AM_DSBM cksum:43AF
                        ttl:254 reserved:0 length:88
*Dec 31 16:44:29.651:  DSBM_IP_ADDR       type 1 length 8 : 91020296
*Dec 31 16:44:29.651:  HOP_L2             type 1 length 12: 00E01ECE
*Dec 31 16:44:29.651:                                     : 0F760000
*Dec 31 16:44:29.651:  SBM_PRIORITY       type 1 length 8 : 00000064
*Dec 31 16:44:29.651:  DSBM_TIMERS        type 1 length 8 : 00000F05
*Dec 31 16:44:29.651:  SBM_INFO           type 1 length 44: 00000000
*Dec 31 16:44:29.651:                                     : 00240C02 00000007
*Dec 31 16:44:29.651:                                     : 01000006 7F000005
*Dec 31 16:44:29.651:                                     : 00000000 00000000
*Dec 31 16:44:29.655:                                     : 00000000 00000000
*Dec 31 16:44:29.655:                                     : 00000000
```

Related Commands

Command	Description
debug ip rsvp	Displays information about SBM message processing, the DSBM election process, and RSVP message processing.
debug ip rsvp sbm	Displays detailed information about the contents of SMB messages only, and SBM and DSBM state transitions.
ip rsvp dsbm-candidate	Configures an interface as a DSBM candidate.
show ip rsvp sbm	Displays information about SBM configured for a specific RSVP-enabled interface or all RSVP-enabled interfaces on the router.

debug ip rsvp policy

To display debug messages for RSVP policy processing, use the **debug ip rsvp policy** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug ip rsvp policy

no debug ip rsvp policy

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for RSVP policy processing is not enabled.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

You might find it useful to enable the **debug cops** command when you are using the **debug ip rsvp policy** command. Together, these commands generate a complete record of the policy process.

Examples

The following example uses only the **debug ip rsvp policy** command:

```
router-1# debug ip rsvp policy

RSVP_POLICY debugging is on

02:02:14:RSVP-POLICY:Creating outbound policy IDB entry for Ethernet2/0 (61E6AB38)
02:02:14:RSVP-COPS:COPS query for Path message, 10.31.0.1_44->10.33.0.1_44
02:02:14:RSVP-POLICY:Building incoming Path context
02:02:14:RSVP-POLICY:Building outgoing Path context on Ethernet2/0
02:02:14:RSVP-POLICY:Build REQ message of 216 bytes
02:02:14:RSVP-POLICY:Message sent to PDP
02:02:14:RSVP-COPS:COPS engine called us with reason2, handle 6202A658
02:02:14:RSVP-COPS:Received decision message
02:02:14:RSVP-POLICY:Received decision for Path message
02:02:14:RSVP-POLICY:Accept incoming message
02:02:14:RSVP-POLICY:Send outgoing message to Ethernet2/0
02:02:14:RSVP-POLICY:Replacement policy object for path-in context
02:02:14:RSVP-POLICY:Replacement TSPEC object for path-in context
02:02:14:RSVP-COPS:COPS report for Path message, 10.31.0.1_44->10.33.0.1_44
02:02:14:RSVP-POLICY:Report sent to PDP
02:02:14:RSVP-COPS:COPS report for Path message, 10.31.0.1_44->10.33.0.1_44
```

The following example uses both the **debug ip rsvp policy** and the **debug cops** commands:

```
router-1# debug ip rsvp policy

RSVP_POLICY debugging is on

router-1# debug cops
```

```
COPS debugging is on

02:15:14:RSVP-POLICY:Creating outbound policy IDB entry for Ethernet2/0 (61E6AB38)
02:15:14:RSVP-COPS:COPS query for Path message, 10.31.0.1_44->10.33.0.1_44
02:15:14:RSVP-POLICY:Building incoming Path context
02:15:14:RSVP-POLICY:Building outgoing Path context on Ethernet2/0
02:15:14:RSVP-POLICY:Build REQ message of 216 bytes
02:15:14:COPS:** SENDING MESSAGE **
    COPS HEADER:Version 1, Flags 0, Opcode 1 (REQ), Client-type:1, Length:216
    HANDLE (1/1) object. Length:8.    00 00 22 01
    CONTEXT (2/1) object. Length:8.   R-type:5.     M-type:1
    IN_IF (3/1) object. Length:12.   Address:10.1.2.1.    If_index:4
    OUT_IF (4/1) object. Length:12.   Address:10.33.0.1.    If_index:3
    CLIENT SI (9/1) object. Length:168.   CSI data:
02:15:14: SESSION              type 1 length 12:
02:15:14:      Destination 10.33.0.1, Protocol_Id 17, Don't Police , DstPort 44
02:15:14: HOP                  type 1 length 12:0A010201
02:15:14:                                       :00000000
02:15:14: TIME_VALUES          type 1 length 8 :00007530
02:15:14: SENDER_TEMPLATE      type 1 length 12:
02:15:14:      Source 10.31.0.1, udp_source_port 44
02:15:14: SENDER_TSPEC         type 2 length 36:
02:15:14:      version=0, length in words=7
02:15:14:      Token bucket fragment (service_id=1, length=6 words
02:15:14:          parameter id=127, flags=0, parameter length=5
02:15:14:          average rate=1250 bytes/sec, burst depth=10000 bytes
02:15:14:          peak rate   =1250000 bytes/sec
02:15:14:          min unit=0 bytes, max unit=1514 bytes
02:15:14: ADSPEC               type 2 length 84:
02:15:14: version=0  length in words=19
02:15:14: General Parameters  break bit=0  service length=8
02:15:14:                                            IS Hops:1
02:15:14:                 Minimum Path Bandwidth (bytes/sec):1250000
02:15:14:                        Path Latency (microseconds):0
02:15:14:                                           Path MTU:1500
02:15:14: Guaranteed Service  break bit=0  service length=8
02:15:14:                          Path Delay (microseconds):192000
02:15:14:                         Path Jitter (microseconds):1200
02:15:14:            Path delay since shaping (microseconds):192000
02:15:14:           Path Jitter since shaping (microseconds):1200
02:15:14: Controlled Load Service  break bit=0  service length=0
02:15:14:COPS:Sent 216 bytes on socket,
02:15:14:RSVP-POLICY:Message sent to PDP
02:15:14:COPS:Message event!
02:15:14:COPS:State of TCP is 4
02:15:14:In read function
02:15:14:COPS:Read block of 96 bytes, num=104 (len=104)
02:15:14:COPS:** RECEIVED MESSAGE **
    COPS HEADER:Version 1, Flags 1, Opcode 2 (DEC), Client-type:1, Length:104
    HANDLE (1/1) object. Length:8.    00 00 22 01
    CONTEXT (2/1) object. Length:8.   R-type:1.     M-type:1
    DECISION (6/1) object. Length:8.   COMMAND cmd:1, flags:0
    DECISION (6/3) object. Length:56.   REPLACEMENT  00 10 0E 01 61 62 63 64 65 66 67
    68 69 6A 6B 6C 00 24 0C 02 00
    00 00 07 01 00 00 06 7F 00 00 05 44 9C 40 00 46 1C 40 00 49 98
    96 80 00 00 00 C8 00 00 01 C8
    CONTEXT (2/1) object. Length:8.   R-type:4.     M-type:1
    DECISION (6/1) object. Length:8.   COMMAND cmd:1, flags:0

02:15:14:Notifying client (callback code 2)
02:15:14:RSVP-COPS:COPS engine called us with reason2, handle 6202A104
02:15:14:RSVP-COPS:Received decision message
02:15:14:RSVP-POLICY:Received decision for Path message
02:15:14:RSVP-POLICY:Accept incoming message
```

```
02:15:14:RSVP-POLICY:Send outgoing message to Ethernet2/0
02:15:14:RSVP-POLICY:Replacement policy object for path-in context
02:15:14:RSVP-POLICY:Replacement TSPEC object for path-in context
02:15:14:RSVP-COPS:COPS report for Path message, 10.31.0.1_44->10.33.0.1_44
02:15:14:COPS:** SENDING MESSAGE **
    COPS HEADER:Version 1, Flags 1, Opcode 3 (RPT), Client-type:1, Length:24
    HANDLE (1/1) object. Length:8.    00 00 22 01
    REPORT (12/1) object. Length:8.   REPORT type COMMIT (1)

02:15:14:COPS:Sent 24 bytes on socket,
02:15:14:RSVP-POLICY:Report sent to PDP
02:15:14:Timer for connection entry is zero
02:15:14:RSVP-COPS:COPS report for Path message, 10.31.0.1_44->10.33.0.1_44
```

Related Commands

Command	Description
debug cops	Displays debug messages for COPS processing.

debug ip rsvp sbm

To display detailed information about Subnetwork Bandwidth Manager (SBM) messages only, and SBM and Designated Subnetwork Bandwidth Manager (DSBM) state transitions, use the **debug ip rsvp sbm** privileged EXEC command. To turn off debugging when you no longer want to display the output, use the **no** form of this command.

debug ip rsvp sbm

no debug ip rsvp sbm

Syntax Description

This command has no arguments or keywords.

Defaults

This command is disabled by default.

Usage Guidelines

The **debug ip rsvp sbm** command provides information about messages received, minimal detail about the content of these messages, and information about state transitions.

Command History

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

Examples

The following example shows the detailed debug information about SBM and the SBM and DSBM state transitions that is available when you enable debug mode through the **debug ip rsvp sbm** command:

```
Router# debug ip rsvp sbm

RSVP debugging is on
router2#
*Dec 31 16:45:34.659: RSVP: send I_AM_DSBM message from 145.2.2.150
*Dec 31 16:45:34.659: RSVP: IP to 224.0.0.17 length=88 checksum=9385 (Ethernet2)
*Dec 31 16:45:34.659:  RSVP: version:1 flags:0000 type:I_AM_DSBM cksum:9385
                        ttl:254    reserved:0 length:88
*Dec 31 16:45:34.659:  DSBM_IP_ADDR       type 1 length 8 : 91020296
*Dec 31 16:45:34.659:  HOP_L2             type 1 length 12: 00E01ECE
*Dec 31 16:45:34.659:                                     : 0F760000
*Dec 31 16:45:34.659:  SBM_PRIORITY       type 1 length 8 : 0029B064
*Dec 31 16:45:34.659:  DSBM_TIMERS        type 1 length 8 : 00000F05
*Dec 31 16:45:34.659:  SBM_INFO           type 1 length 44: 00000000
*Dec 31 16:45:34.659:                                     : 00240C02 00000007
*Dec 31 16:45:34.659:                                     : 01000006 7F000005
*Dec 31 16:45:34.659:                                     : 00000000 00000000
*Dec 31 16:45:34.663:                                     : 00000000 00000000
*Dec 31 16:45:34.663:                                     : 00000000
*Dec 31 16:45:34.663:
```

Related Commands

Command	Description
debug ip rsvp	Displays information about SBM message processing, the DSBM election process, and RSVP message processing.
debug ip rsvp detail	Displays detailed information about RSVP and SBM
ip rsvp dsbm-candidate	Configures an interface as a DSBM candidate.

debug ip rsvp traffic-control

To display debug messages for traffic control, use the **debug ip rsvp traffic-control** EXEC command. To disable the **debug ip rsvp traffic-control** command, use the **no** form of this command.

debug ip rsvp traffic-control

no debug ip rsvp traffic-control

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.0	This command was introduced.

Examples

The following is an example of output from the **debug ip rsvp traffic-control** command:

```
Router# debug ip rsvp traffic-control

RSVP debugging is on

Router# show debugging

IP RSVP debugging is on
IP RSVP debugging (Traffic Control events) is on
Router#
03:03:56:RSVP-TC:Attempting to remove QoS for rsb 6268A538
03:03:56:RSVP-TC:tcsb 00001A01 found for rsb 6268A538
03:03:56:RSVP-TC:Deleting tcsb 00001A01
03:04:15:RSVP-TC:Attempting to install QoS for rsb 6268A538
03:04:15:RSVP-TC:Adding new tcsb 00001E01 for rsb 6268A538
03:04:15:RSVP-TC:Assigning WFQ QoS to tcsb 00001E01
03:04:15:RSVP-TC:Consulting policy for tcsb 00001E01
03:04:15:RSVP-TC:Policy granted QoS for tcsb 00001E01
03:04:15:RSVP-TC:Requesting QoS for tcsb 00001E01
03:04:15:RSVP-TC:   ( r = 12500      bytes/s   M = 1514       bytes
03:04:15:RSVP-TC:     b = 1000       bytes     m = 0          bytes )
03:04:15:RSVP-TC:     p = 12500      bytes/s   Service Level = non-priority
03:04:15:RSVP-TC:Allocation succeeded for tcsb 00001E01
```

Related Commands

Command	Description
show debug	Displays active debug output.

debug ip rsvp wfq

To display debug messages for the weighted fair queue (WFQ), use the **debug ip rsvp wfq** EXEC command. To disable the command, use the **no** form of this command.

debug ip rsvp wfq

no debug ip rsvp wfq

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

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

Examples

The following is an example of output from the **debug ip rsvp wfq** command:

```
Router# debug ip rsvp wfq

RSVP debugging is on

Router# show debugging

IP RSVP debugging is on
IP RSVP debugging (Traffic Control events) is on
IP RSVP debugging (WFQ events) is on
Router#
03:03:23:RSVP-TC:Attempting to install QoS for rsb 6268A538
03:03:23:RSVP-TC:Adding new tcsb 00001A01 for rsb 6268A538
03:03:23:RSVP-TC:Assigning WFQ QoS to tcsb 00001A01
03:03:23:RSVP-TC:Consulting policy for tcsb 00001A01
03:03:23:RSVP-TC:Policy granted QoS for tcsb 00001A01
03:03:23:RSVP-TC:Requesting QoS for tcsb 00001A01
03:03:23:RSVP-TC:   ( r = 12500      bytes/s   M = 1514       bytes
03:03:23:RSVP-TC:     b = 1000       bytes     m = 0          bytes )
03:03:23:RSVP-TC:     p = 12500      bytes/s   Service Level = non-priority
03:03:23:RSVP-WFQ:Requesting a RESERVED queue on Et0/1 for tcsb 00001A01
03:03:23:RSVP-WFQ:Queue 265 allocated for tcsb 00001A01
03:03:23:RSVP-TC:Allocation succeeded for tcsb 00001A01
Router#

Router# no debug ip rsvp

RSVP debugging is off
```

Related Commands

Command	Description
show debug	Displays active debug output.

debug ip rtp header-compression

To display events specific to RTP header compression, use the **debug ip rtp header-compression** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug ip rtp header-compression

no debug ip rtp header-compression

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug ip rtp header-compression** command:

```
Router# debug ip rtp header-compression

RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 0, received sequence 0
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 1, received sequence 1
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 2, received sequence 2
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 3, received sequence 3
```

Table 97 describes the significant fields shown in the output.

Table 97 debug ip rtp header-compression Field Descriptions

Field	Description
context0	Compression state for a connection 0.
expected sequence	RTP header compression link sequence (expected).
received sequence	RTP header compression link sequence (actually received).

Related Commands

Command	Description
debug ip rtp packets	Displays a detailed dump of packets specific to RTP header compression.

debug ip rtp packets

To display a detailed dump of packets specific to RTP header compression, use the **debug ip rtp packets** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug ip rtp packets

no debug ip rtp packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ip rtp packets

RTP packet dump:
  IP:  source: 171.68.8.10, destination: 224.2.197.169, id: 0x249B, ttl: 9,
       TOS: 0 prot: 17,
  UDP: source port: 1034, destination port: 27404, checksum: 0xB429,len: 152
  RTP: version: 2, padding: 0, extension: 0, marker: 0,
       payload: 3, ssrc 2369713968,
       sequence: 2468, timestamp: 85187180, csrc count: 0
```

Table 98 describes the significant fields shown in the output.

Table 98 debug ip rtp packets Field Descriptions

Field	Description
id	IP identification.
ttl	IP time to live (TTL).
len	Total UDP length.

Related Commands

Command	Description
debug ip rtp header-compression	Displays events specific to RTP header compression.

debug ip sd

To display all session directory (SD) announcements received, use the **debug ip sd** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip sd

no debug ip sd

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command shows session directory announcements for multicast IP. Use it to observe multicast activity.

Examples

The following is sample output from the **debug ip sd** command:

```
Router# debug ip sd

SD: Announcement from 172.16.58.81 on Serial0.1, 146 bytes
    s=*cisco: CBONE Audio
    i=cisco internal-only audio conference
    o=dino@dino-ss20.cisco.com
    c=224.0.255.1 16 2891478496 2892688096
    m=audio 31372 1700
SD: Announcement from 172.22.246.68 on Serial0.1, 147 bytes
    s=IMS: U.S. Senate
    i=U.S. Senate at http://town.hall.org/radio/live.html
    o=carl@also.radio.com
    c=224.2.252.231 95 0 0
    m=audio 36572 2642
    a=fmt:gsm
```

Table 99 describes the significant fields in the output.

Table 99 debug ip sd Output Descriptions

Field	Description
SD	Session directory event.
Announcement from	Address sending the SD announcement.
on Serial0.1	Interface receiving the announcement.
146 bytes	Size of the announcement event.
s=	Session name being advertised.
i=	Information providing a descriptive name for the session.
o=	Origin of the session, either an IP address or a name.
c=	Connect description showing address and number of hops.
m=	Media description that includes media type, port number, and ID.

Related Commands

Command	Description
debug ip dvmrp	Displays information on DVMRP packets received and sent.
debug ip igmp	Displays IGMP packets received and sent, and IGMP host-related events.
debug ip mbgp dampening	Logs route flap dampening activity related to MBGP.
debug ip mrouting	Displays changes to the IP multicast routing table.
debug ip pim	Displays PIM packets received and sent, and PIM-related events.

debug ip security

To display IP security option processing, use the **debug ip security** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip security

no debug ip security

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug ip security** command displays information for both basic and extended IP security options. For interfaces where **ip security** is configured, each IP packet processed for that interface results in debugging output regardless of whether the packet contains IP security options. IP packets processed for other interfaces that also contain IP security information also trigger debugging output. Some additional IP security debugging information is also controlled by the **debug ip packet** privileged EXEC command.

Caution Because the **debug ip security** command generates a substantial amount of output for every IP packet processed, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Examples

The following is sample output from the **debug ip security** command:

```
Router# debug ip security

IP Security: src 172.24.72.52 dst 172.24.72.53, number of BSO 1
     idb: NULL
     pak: insert (0xFF) 0x0
IP Security: BSO postroute: SECINSERT changed to secret (0x5A) 0x10
IP Security: src 172.24.72.53 dst 172.24.72.52, number of BSO 1
     idb: secret (0x6) 0x10 to secret (0x6) 0x10, no implicit
          def secret (0x6) 0x10
     pak: secret (0x5A) 0x10
IP Security: checking BSO 0x10 against [0x10 0x10]
IP Security: classified BSO as secret (0x5A) 0x10
```

Table 100 describes significant fields shown in the output.

Table 100 debug ip security Field Descriptions

Field	Description
number of BSO	Indicates the number of basic security options found in the packet.
idb	Provides information on the security configuration for the incoming interface.
pak	Provides information on the security classification of the incoming packet.
src	Indicates the source IP address.
dst	Indicates the destination IP address.

The following line indicates that the packet was locally generated, and it has been classified with the internally significant security level “insert” (0xff) and authority information of 0x0:

```
idb: NULL
pak: insert (0xff) 0x0
```

The following line indicates that the packet was received via an interface with dedicated IP security configured. Specifically, the interface is configured at security level “secret” and with authority information of 0x0. The packet itself was classified at level “secret” (0x5a) and authority information of 0x10.

```
idb: secret (0x6) 0x10 to secret (0x6) 0x10, no implicit
     def secret (0x6) 0x10
pak: secret (0x5A) 0x10
```

debug ip slb

To display debug messages for the Cisco IOS Server Load Balancing (SLB) feature, use the **debug ip slb** EXEC command. To stop debug output, use the **no** form of this command.

debug ip slb {**conns** | **dfp** | **icmp** | **reals** | **all**}

no debug ip slb {**conns** | **dfp** | **icmp** | **reals** | **all**}

Syntax Description

conns	Displays debug messages for all connections being handled by Cisco IOS SLB.
dfp	Displays debug messages for the Cisco IOS SLB DFP and DFP agents.
icmp	Displays all Internet Control Message Protocol (ICMP) debug messages for Cisco IOS SLB.
reals	Displays debug messages for all real servers defined to Cisco IOS SLB.
all	Displays all debug messages for Cisco IOS SLB.

Defaults

No default behavior or values.

Command History

Release	Modification
12.0(7)XE	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

See the following caution before using **debug** commands.

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, only use **debug** commands to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, it is best to use **debug** commands during periods of lower network flows and fewer users. Debugging during these periods reduces the effect these commands have on other users on the system.

Examples

The following example configures a debug session to check all IP IOS SLB parameters:

```
Router# debug ip slb all

SLB All debugging is on
Router#
```

The following example stops all debugging:

```
Router# no debug all

All possible debugging has been turned off
Router#
```

The following example shows Cisco IOS SLB DFP debug output:

```
router# debug ip slb dfp

SLB DFP debugging is on
router#
022048 SLB DFP Queue to main queue - type 2 for Agent 161.44.2.3458229
022048 SLB DFP              select_rc = -1   readset = 0
022048 SLB DFP       Sleeping ...
022049 SLB DFP              readset = 0
022049 SLB DFP              select_rc = -1   readset = 0
022049 SLB DFP Processing Q event for Agent 161.44.2.3458229 - OPEN
022049 SLB DFP Queue to conn_proc_q - type 2 for Agent 161.44.2.3458229
022049 SLB DFP              readset = 0
022049 SLB DFP Set SLB_DFP_SIDE_QUEUE
022049 SLB DFP Processing Conn Q event for Agent 161.44.2.3458229 - OPEN
022049 SLB DFP Open to Agent 161.44.2.3458229 succeeded, socket = 0
022049 SLB DFP Agent 161.44.2.3458229 start connect
022049 SLB DFP Connect to Agent 161.44.2.3458229 successful - socket 0


022049 SLB DFP Queue to main queue - type 6 for Agent 161.44.2.3458229
022049 SLB DFP Processing Conn Q unknown MAJOR 80
022049 SLB DFP Reset SLB_DFP_SIDE_QUEUE
022049 SLB DFP              select_rc = -1   readset = 0
022049 SLB DFP       Sleeping ...
022050 SLB DFP              readset = 1
022050 SLB DFP              select_rc = 1   readset = 1
022050 SLB DFP Agent 161.44.2.3458229 fd = 0 readset = 1
022050 SLB DFP Message length 44 from Agent 161.44.2.3458229
022050 SLB DFP Agent 161.44.2.3458229 setting Host 17.17.17.17, Bind ID 1 Weight 1
022050 SLB DFP Agent 161.44.2.3458229 setting Host 34.34.34.34, Bind ID 2 Weight 2
022050 SLB DFP Agent 161.44.2.3458229 setting Host 51.51.51.51, Bind ID 3 Weight 3
022050 SLB DFP Processing Q event for Agent 161.44.2.3458229 - WAKEUP


022050 SLB DFP              readset = 1
022050 SLB DFP              select_rc = 1   readset = 1
022050 SLB DFP Agent 161.44.2.3458229 fd = 0 readset = 1
022050 SLB DFP Message length 64 from Agent 161.44.2.3458229
022050 SLB DFP Agent 161.44.2.3458229 setting Host 17.17.17.17, Bind ID 1 Weight 1
022050 SLB DFP Agent 161.44.2.3458229 setting Host 68.68.68.68, Bind ID 4 Weight 4
022050 SLB DFP Agent 161.44.2.3458229 setting Host 85.85.85.85, Bind ID 5 Weight 5
022050 SLB DFP Agent 161.44.2.3458229 setting Host 17.17.17.17, Bind ID 111 Weight 111
022050 SLB DFP              readset = 1


022115 SLB DFP Queue to main queue - type 5 for Agent 161.44.2.3458229
022115 SLB DFP              select_rc = -1   readset = 0
022115 SLB DFP       Sleeping ...
022116 SLB DFP              readset = 1
022116 SLB DFP              select_rc = -1   readset = 0
022116 SLB DFP Processing Q event for Agent 161.44.2.3458229 - DELETE


022116 SLB DFP Queue to conn_proc_q - type 5 for Agent 161.44.2.3458229
022116 SLB DFP              readset = 1
022116 SLB DFP Set SLB_DFP_SIDE_QUEUE
```

```
022116 SLB DFP Processing Conn Q event for Agent 161.44.2.3458229 - DELETE
022116 SLB DFP Connection to Agent 161.44.2.3458229 closed
022116 SLB DFP Agent 161.44.2.3458229 deleted
022116 SLB DFP Processing Conn Q unknown MAJOR 80
022116 SLB DFP Reset SLB_DFP_SIDE_QUEUE
022116 SLB DFP Set SLB_DFP_SIDE_QUEUE
022116 SLB DFP Reset SLB_DFP_SIDE_QUEUE
```

debug ip socket

To display all state change information for all sockets, use the **debug ip socket** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug ip socket

no debug ip socket

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to collect information on the socket interface. To get more complete information on a socket/TCP port pair, use this command in conjunction with the **debug ip tcp transactions** command.

Because the socket debugging information is state change oriented, you will not see the debug message on a per-packet basis. However, if the connections normally have very short lives (few packet exchanges during the life cycle of a connection), then socket debugging could become expensive because of the state changes involved during connection setup and teardown.

Examples

The following is sample output from the **debug ip socket** output from a server process:

```
Router# debug ip socket

Added socket 0x60B86228 to process 40
SOCKET: set TCP property TCP_PID, socket 0x60B86228, TCB 0x60B85E38
Accepted new socket fd 1, TCB 0x60B85E38
Added socket 0x60B86798 to process 40
SOCKET: set TCP property TCP_PID, socket 0x60B86798, TCB 0x60B877C0
SOCKET: set TCP property TCP_BIT_NOTIFY, socket 0x60B86798, TCB 0x60B877C0
SOCKET: created new socket to TCP, fd 2, TCB 0x60B877C0
SOCKET: bound socket fd 2 to TCB 0x60B877C0
SOCKET: set TCP property TCP_WINDOW_SIZE, socket 0x60B86798, TCB 0x60B877C0
SOCKET: listen on socket fd 2, TCB 0x60B877C0
SOCKET: closing socket 0x60B86228, TCB 0x60B85E38
SOCKET: socket event process: socket 0x60B86228, TCB new state --> FINWAIT1
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING
SOCKET: Removed socket 0x60B86228 from process 40 socket list
```

The following is sample output from the **debug ip socket** command from a client process:

```
Router# debug ip socket

Added socket 0x60B70220 to process 2
SOCKET: set TCP property TCP_PID, socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: set TCP property TCP_BIT_NOTIFY, socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: created new socket to TCP, fd 0, TCB 0x60B6CFDC
SOCKET: socket event process: socket 0x60B70220, TCB new state --> SYNSENT
socket state: SS_ISCONNECTING
SOCKET: socket event process: socket 0x60B70220, TCB new state --> ESTAB
socket state: SS_ISCONNECTING
SOCKET: closing socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: socket event process: socket 0x60B70220, TCB new state --> FINWAIT1
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING
SOCKET: Removed socket 0x60B70220 from process 2 socket list
```

Table 101 describes the significant fields shown in the display.

Table 101 debug ip socket Field Descriptions

Field	Description
Added socket 0x60B86228 process 40	New socket is opened for process 40.
SOCKET	Indicates that this is a SOCKET transaction.
set TCP property TCP_PID	Sets the process ID to the TCP associated with the socket.
socket 0x60B86228, TCB 0x60B85E38	Address for the socket/TCP pair.
set TCP property TCP_BIT_NOTIFY	Sets the method for how the socket wants to be notified for an event.
created new socket to TCP, fd 2	Opened a new socket referenced by file descriptor 2 to TCP.
bound socket fd 2 to TCB	Bound the socket referenced by file descriptor 2 to TCP.
listen on socket fd 2	Indicates which file descriptor the application is listening to.
closing socket	Indicates that the socket is being closed.
socket event process	Processed a state change event occurred in the transport layer.
TCB new state --> FINWAIT1	TCP state machine changed to FINWAIT1. (See the **debug ip tcp transaction** command for more information on TCP state machines.)

Table 101 debug ip socket Field Descriptions (continued)

Field	Description
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING	New SOCKET state flags after the transport event processing. This socket is still connected, but disconnecting is in progress, and it will not send more data to peer. Possible SOCKET state flags follow: • SS_NOFDREF No file descriptor reference for this socket. • SS_ISCONNECTING Socket connecting is in progress. • SS_ISBOUND Socket is bound to TCP. • SS_ISCONNECTED Socket is connected to peer. • SS_ISDISCONNECTING Socket disconnecting is in progress. • SS_CANTSENDMORE Can't send more data to peer. • SS_CANTRCVMORE Can't receive more data from peer. • SS_ISDISCONNECTED Socket is disconnected. Connection is fully closed.
Removed socket 0x60B86228 from process 40 socket list	Connection is closed, and the socket is removed from the process socket list.

Related Commands

Command	Description
debug ip tcp transactions	Displays information on significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip ssh

To display debug messages for Secure Shell (SSH), use the **debug ip ssh** EXEC command. To disable debugging output, use the **no** form of the command.

debug ip ssh

no debug ip ssh

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for SSH is not enabled.

Command History

Release	Modification
12.0(5)S	This command was introduced.
12.1(1)T	This command was integrated into Cisco IOS Release 12.1 T.

Usage Guidelines

Use the **debug ssh** command to ensure normal operation of the SSH server.

Examples

The following example shows the SSH debugging output:

```
Router# debug ssh

00:53:46: SSH0: starting SSH control process
00:53:46: SSH0: Exchanging versions - SSH-1.5-Cisco-1.25

00:53:46: SSH0: client version is - SSH-1.5-1.2.25
00:53:46: SSH0: SSH_SMSG_PUBLIC_KEY message sent
00:53:46: SSH0: SSH_CMSG_SESSION_KEY message received
00:53:47: SSH0: keys exchanged and encryption on
00:53:47: SSH0: authentication request for userid guest
00:53:47: SSH0: authentication successful for jcisco
00:53:47: SSH0: starting exec shell
```

debug ip tcp driver

To display information on TCP driver events; for example, connections opening or closing, or packets being dropped because of full queues, use the **debug ip tcp driver** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip tcp driver

no debug ip tcp driver

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The TCP driver is the process that the router software uses to send packet data over a TCP connection. Remote source-route bridging (RSRB), serial tunneling (STUN), and X.25 switching currently use the TCP driver.

Using the **debug ip tcp driver** command together with the **debug ip tcp driver-pak** command provides the most verbose debugging output concerning TCP driver activity.

Examples

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

```
Router# debug ip tcp driver

TCPDRV359CD8: Active open 172.21.80.26:0 --> 172.21.80.25:1996 OK, lport 36628
TCPDRV359CD8: enable tcp timeouts
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 Abort
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 DoClose tcp abort
```

Table 102 describes the significant fields shown in the display.

Table 102 debug ip tcp driver Field Descriptions

Field	Description
TCPDRV359CD8:	Unique identifier for this instance of TCP driver activity.
Active open 172.21.80.26	Indication that the router at IP address 172.21.80.26 has initiated a connection to another router.
:0	TCP port number the initiator of the connection uses to indicate that any port number can be used to set up a connection.
--> 172.21.80.25	IP address of the remote router to which the connection has been initiated.
:1996	TCP port number that the initiator of the connection is requesting that the remote router use for the connection. (1996 is a private TCP port number reserved in this implementation for RSRB.)
OK,	Indication that the connection has been established. If the connection has not been established, this field and the following field do not appear in this line of output.
lport 36628	TCP port number that has actually been assigned for the initiator to use for this connection.

The following line indicates that the TCP driver user (RSRB, in this case) will allow TCP to drop the connection if excessive retransmissions occur:

```
TCPDRV359CD8: enable tcp timeouts
```

The following line indicates that the TCP driver user (in this case, RSRB) at IP address 172.21.80.26 (and using TCP port number 36628) is requesting that the connection to IP address 172.21.80.25 using TCP port number 1996 be aborted:

```
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 Abort
```

The following line indicates that this connection was in fact closed due to an abnormal termination:

```
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 DoClose tcp abort
```

debug ip tcp driver-pak

To display information on every operation that the TCP driver performs, use the **debug ip tcp driver-pak** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip tcp driver-pak

no debug ip tcp driver-pak

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command turns on a verbose debugging by logging at least one debugging message for every packet sent or received on the TCP driver connection.

The TCP driver is the process that the router software uses to send packet data over a TCP connection. RSRB, serial tunneling (STUN), and X.25 switching currently use the TCP driver.

To observe the context within which certain **debug ip tcp driver-pak** messages occur, turn on this command in conjunction with the **debug ip tcp driver** command.

Caution Because the **debug ip tcp driver-pak** command generates so many messages, use it only on lightly loaded systems. This command not only places a substantial load on the system processor, it also may change the symptoms of any unexpected behavior that occurs.

Examples

The following is sample output from the **debug ip tcp driver-pak** command:

```
Router# debug ip tcp driver-pak

TCPDRV359CD8: send 2E8CD8 (len 26) queued
TCPDRV359CD8: output pak 2E8CD8 (len 26) (26)
TCPDRV359CD8: readf 42 bytes (Thresh 16)
TCPDRV359CD8: readf 26 bytes (Thresh 16)
TCPDRV359CD8: readf 10 bytes (Thresh 10)
TCPDRV359CD8: send 327E40 (len 4502) queued
TCPDRV359CD8: output pak 327E40 (len 4502) (4502)
```

Table 103 describes the significant fields shown in the display.

Table 103 debug ip tcp driver-pak Field Descriptions

Field	Description
TCPDRV359CD8	Unique identifier for this instance of TCP driver activity.
send	Indicates that this event involves the TCP driver sending data.
2E8CD8	Address in memory of the data the TCP driver is sending.
(len 26)	Length of the data (in bytes).
queued	Indicates that the TCP driver user process (in this case, RSRB) has transferred the data to the TCP driver to send.

The following line indicates that the TCP driver has sent the data that it had received from the TCP driver user, as shown in the previous line of output. The last field in the line (26) indicates that the 26 bytes of data were sent out as a single unit.

```
TCPDRV359CD8: output pak 2E8CD8 (len 26) (26)
```

The following line indicates that the TCP driver has received 42 bytes of data from the remote IP address. The TCP driver user (in this case, remote source-route bridging) has established an input threshold of 16 bytes for this connection. (The input threshold instructs the TCP driver to transfer data to the TCP driver user only when at least 16 bytes are present.)

```
TCPDRV359CD8: readf 42 bytes (Thresh 16)
```

debug ip tcp intercept

To display TCP intercept statistics, use the **debug ip tcp intercept** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip tcp intercept

no debug ip tcp intercept

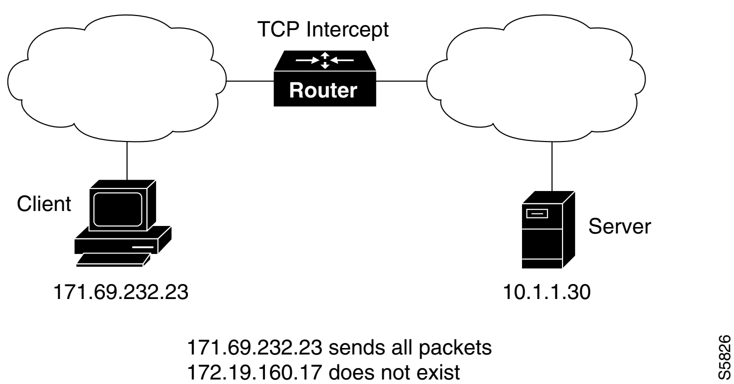
Syntax Description

This command has no arguments or keywords.

Examples

Figure 4 illustrates a scenario in which a router configured with TCP intercept operates between a client and a server.

Figure 4 Example TCP Intercept Environment



The following is sample output from the **debug ip tcp intercept** command:

```
Router# debug ip tcp intercept
```

A connection attempt arrives:

```
INTERCEPT: new connection (172.19.160.17:61774) => (10.1.1.30:23)
INTERCEPT: 172.19.160.17:61774 <- ACK+SYN (10.1.1.30:61774)
```

A second connection attempt arrives:

```
INTERCEPT: new connection (172.19.160.17:62030) => (10.1.1.30:23)
INTERCEPT: 172.19.160.17:62030 <- ACK+SYN (10.1.1.30:62030)
```

The router re-sends to both apparent clients:

```
INTERCEPT: retransmit 2 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 2 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
```

A third connection attempt arrives:

```
INTERCEPT: new connection (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: 171.69.232.23:1048 <- ACK+SYN (10.1.1.30:1048)
```

The router sends more retransmissions trying to establish connections with the apparent clients:

```
INTERCEPT: retransmit 4 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 4 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 2 (171.69.232.23:1048) <- (10.1.1.30:23) SYNRCVD
```

The router establishes the connection with the third client and re-sends to the server:

```
INTERCEPT: 1st half of connection is established (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: (171.69.232.23:1048) SYN -> 10.1.1.30:23
INTERCEPT: retransmit 2 (171.69.232.23:1048) -> (10.1.1.30:23) SYNSENT
```

The server responds; the connection is established:

```
INTERCEPT: 2nd half of connection established (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: (171.69.232.23:1048) ACK -> 10.1.1.30:23
```

The router re-sends to the first two apparent clients, times out, and sends resets:

```
INTERCEPT: retransmit 8 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 8 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 16 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 16 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmitting too long (172.19.160.17:61774) => (10.1.1.30:23) SYNRCVD
INTERCEPT: 172.19.160.17:61774 <- RST (10.1.1.30:23)
INTERCEPT: retransmitting too long (172.19.160.17:62030) => (10.1.1.30:23) SYNRCVD
INTERCEPT: 172.19.160.17:62030 <- RST (10.1.1.30:23)
```

debug ip tcp transactions

To display information on significant TCP transactions such as state changes, retransmissions, and duplicate packets, use the **debug ip tcp transactions** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip tcp transactions

no debug ip tcp transactions

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is particularly useful for debugging a performance problem on a TCP/IP network that you have isolated above the data link layer.

The **debug ip tcp transactions** command displays output for packets the router sends and receives, but does not display output for packets it forwards.

Examples

The following is sample output from the **debug ip tcp transactions** command:

```
Router# debug ip tcp transactions

TCP: sending SYN, seq 168108, ack 88655553
TCP0: Connection to 10.9.0.13:22530, advertising MSS 966
TCP0: state was LISTEN -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: state was SYNSENT -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: Connection to 10.9.0.13:22530, received MSS 956
TCP0: restart retransmission in 5996
TCP0: state was SYNRCVD -> ESTAB [23 -> 10.9.0.13(22530)]
TCP2: restart retransmission in 10689
TCP2: restart retransmission in 10641
TCP2: restart retransmission in 10633
TCP2: restart retransmission in 13384 -> 10.0.0.13(16151)]
TCP0: restart retransmission in 5996 [23 -> 10.0.0.13(16151)]
```

Table 104 describes the significant fields shown in the display.

Table 104 debug ip tcp transactions Field Descriptions

Field	Description
TCP:	Indicates that this is a TCP transaction.
sending SYN	Indicates that a synchronize packet is being sent.
seq 168108	Indicates the sequence number of the data being sent.
ack 88655553	Indicates the sequence number of the data being acknowledged.
TCP0:	Indicates the TTY number (0, in this case) with which this TCP connection is associated.
Connection to 10.9.0.13:22530	Indicates the remote address with which a connection has been established.

Table 104 debug ip tcp transactions Field Descriptions (continued)

Field	Description
advertising MSS 966	Indicates the maximum segment size this side of the TCP connection is offering to the other side.
state was LISTEN -> SYNRCVD	Indicates that the TCP state machine changed state from LISTEN to SYNSENT. Possible TCP states follow: • CLOSED—Connection closed. • CLOSEWAIT—Received a FIN segment. • CLOSING—Received a FIN/ACK segment. • ESTAB—Connection established. • FINWAIT 1—Sent a FIN segment to start closing the connection. • FINWAIT 2—Waiting for a FIN segment. • LASTACK—Sent a FIN segment in response to a received FIN segment. • LISTEN—Listening for a connection request. • SYNRCVD—Received a SYN segment, and responded. • SYNSENT—Sent a SYN segment to start connection negotiation. • TIMEWAIT—Waiting for network to clear segments for this connection before the network no longer recognizes the connection as valid. This must occur before a new connection can be set up.
[23 -> 10.9.0.13(22530)]	The element within these brackets are as follows: • The first field (23) indicates local TCP port. • The second field (10.9.0.13) indicates the destination IP address. • The third field (22530) indicates the destination TCP port.
restart retransmission in 5996	Indicates the number of milliseconds until the next retransmission takes place.

debug ip trigger-authentication

To display information related to automated double authentication, use the **debug ip trigger-authentication** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip trigger-authentication [**verbose**]

no debug ip trigger-authentication [**verbose**]

Syntax Description

verbose	(Optional) Specifies that the complete debugging output be displayed, including information about packets that are blocked before authentication is complete.

Usage Guidelines

Use this command when troubleshooting automated double authentication.

This command displays information about the remote host table. Whenever entries are added, updated, or removed, a new debugging message is displayed.

What is the remote host table? Whenever a remote user needs to be user-authenticated in the second stage of automated double authentication, the local device sends a UDP packet to the host of the remote user. Whenever such a UDP packet is sent, the host IP address of the user is added to a table. If additional UDP packets are sent to the same remote host, a new table entry is not created; instead, the existing entry is updated with a new time stamp. This remote host table contains a cumulative list of host entries; entries are deleted after a timeout period or after you manually clear the table using the **clear ip trigger-authentication** command.

If you include the **verbose** keyword, the debugging output also includes information about packet activity.

Examples

The following is sample output from the **debug ip trigger-authentication** command. In this example, the local device at 172.21.127.186 sends a UDP packet to the remote host at 172.21.127.114. The UDP packet is sent to request the remote user's username and password (or PIN). (The output indicates "New entry added.")

After a timeout period, the local device has not received a valid response from the remote host, so the local device sends another UDP packet. (The output indicates "Time stamp updated.")

Then the remote user is authenticated, and after a length of time (the timeout period) the entry is removed from the remote host table. (The output indicates "remove obsolete entry.")

```
myfirewall# debug ip trigger-authentication

TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.114, qdata=7C2504
              New entry added, timestamp=2940514234
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.114, qdata=7C2504
              Time stamp updated, timestamp=2940514307
TRIGGER_AUTH: remove obsolete entry, remote host=172.21.127.114
```

The following is sample output from the **debug ip trigger-authentication verbose** command. In this example, messages about packet activity are included because of the use of the **verbose** keyword.

You can see many packets that are being blocked at the interface because the user has not yet been double authenticated. These packets will be permitted through the interface only after the user has been double authenticated. (You can see packets being blocked when the output indicates “packet enqueued” then “packet ignored.”)

```
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.113, qdata=69FEEC
              Time stamp updated
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.113, qdata=69FEEC
              Time stamp updated
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
```

debug ip udp

To enable logging of User Datagram Protocol (UDP) packets sent and received, use the **debug ip udp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug ip udp

no debug ip udp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Enter the **debug ip udp** command on the device that should be receiving packets from the host. Check the debugging output to see whether packets are being received from the host.

Caution The **debug ip udp** command can use considerable CPU cycles on the device. Do not enable it if your network is heavily congested.

Examples

The following is sample output from the **debug ip udp** command:

```
Router# debug ip udp
UDP packet debugging is on
Router#

00:18:48: UDP: rcvd src=0.0.0.0(68), dst=255.255.255.255(67), length=584
00:18:48: UDP: sent src=10.1.1.10(67), dst=172.17.110.136(67), length=604
00:18:48: UDP: rcvd src=172.17.110.136(67), dst=10.1.1.10(67), length=308
00:18:48: UDP: sent src=0.0.0.0(67), dst=255.255.255.255(68), length=328
00:18:48: UDP: rcvd src=0.0.0.0(68), dst=255.255.255.255(67), length=584
00:18:48: UDP: sent src=10.1.1.10(67), dst=172.17.110.136(67), length=604
00:18:48: UDP: rcvd src=172.17.110.136(67), dst=10.1.1.10(67), length=308
00:18:50: UDP: sent src=0.0.0.0(67), dst=255.255.255.255(68), length=328
```

debug ip urd

To display debug messages for URL Rendezvous Directory (URD) channel subscription report processing, use the **debug ip urd** EXEC command. To disable debugging of URD reports, use the **no** form of this command.

debug ip urd [*hostname* | *ip-address*]

no debug ip urd

Syntax Description

hostname	(Optional) The domain Name System (DNS) name.
ip-address	(Optional) The IP address.

Defaults

If no host name or IP address is specified, all URD reports are debugged.

Command History

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

Examples

The following is sample output from the **debug ip urd** command:

```
Router# debug ip urd

13:36:25 pdt:URD:Data intercepted from 171.71.225.103
13:36:25 pdt:URD:Enqueued string:
'/cgi-bin/error.pl?group=232.16.16.16&port=32620&source=171.69.214.1&li'
13:36:25 pdt:URD:Matched token:group
13:36:25 pdt:URD:Parsed value:232.16.16.16
13:36:25 pdt:URD:Creating IGMP source state for group 232.16.16.16
```

debug ip wccp events

To display information about significant Web Cache Control Protocol (WCCP) events, use the **debug ip wccp events** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip wccp events

no debug ip wccp events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ip wccp events

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

debug ip wccp packets

To display information about every Web Cache Control Protocol (WCCP) packet received or sent by the router, use the **debug ip wccp packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug ip wccp packets

no debug ip wccp packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug ip wccp packets

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

debug ipx ipxwan

To display debug information for interfaces configured to use IPXWAN, use the **debug ipx ipxwan** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx ipxwan

no debug ipx ipxwan

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug ipx ipxwan** command is useful for verifying the startup negotiations between two routers running the IPX protocol through a WAN. This command produces output only during state changes or startup. During normal operations, no output is produced.

Examples

The following is sample output from the **debug ipx ipxwan** command during link startup:

```
Router# debug ipx ipxwan

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
 state brought up)]
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial1/6666:200 (IPX line
 state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
 state brought up)]

IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 2] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200

IPXWAN: Rcv TIMER_REQ on Serial1/6666:200, NodeID 1234, Seq 1
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Rcv TIMER_RSP on Serial1/6666:200, NodeID 1234, Seq 1, Del 6
IPXWAN: state (Sending Timer Requests -> Master: Sent RIP/SAP) [Serial1/6666:200
 (Received Timer Response as master)]
IPXWAN: Send RIPSAP_INFO_REQ [seq 0] out Serial1/6666:200
IPXWAN: Rcv RIPSAP_INFO_RSP from Serial1/6666:200, NodeID 1234, Seq 0
IPXWAN: state (Master: Sent RIP/SAP -> Master: Connect) [Serial1/6666:200 (Received Router
Info Rsp as Master)]
```

The following line indicates that the interface has initialized:

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
```

The following lines indicate that the startup process failed to receive a timer response, brought the link down, then brought the link up and tried again with a new timer set:

```
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial1/6666:200 (IPX line
 state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
 state brought up)]
```

The following lines indicate that the interface is sending timer requests and waiting for a timer response:

```
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
```

The following lines indicate that the interface has received a timer request from the other end of the link and has sent a timer response. The fourth line shows that the interface has come up as the master on the link.

```
IPXWAN: Rcv TIMER_REQ on Serial1/6666:200, NodeID 1234, Seq 1
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Rcv TIMER_RSP on Serial1/6666:200, NodeID 1234, Seq 1, Del 6
IPXWAN: state (Sending Timer Requests -> Master: Sent RIP/SAP) [Serial1/6666:200
 (Received Timer Response as master)]
```

The following lines indicate that the interface is sending RIP/SAP requests:

```
IPXWAN: Send RIPSAP_INFO_REQ [seq 0] out Serial1/6666:200
IPXWAN: Rcv RIPSAP_INFO_RSP from Serial1/6666:200, NodeID 1234, Seq 0
IPXWAN: state (Master: Sent RIP/SAP -> Master: Connect) [Serial1/6666:200 (Received Router
Info Rsp as Master)]
```

debug ipx nasi

To display information about the NetWare Asynchronous Services Interface (NASI) connections, use the **debug ipx nasi** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx nasi {**packets** | **error** | **activity**}

no debug ipx nasi {**packets** | **error** | **activity**}

Syntax Description

packets	Displays normal operating messages relating to incoming and outgoing NASI packets. This is the default.
error	Displays messages indicating an error or failure in the protocol processing.
activity	Displays messages relating to internal NASI processing of NASI connections. The **activity** option includes all NASI activity such as traffic indication, timer events, and state changes.

Usage Guidelines

Use the **debug ipx nasi** command to display handshaking or negotiating details between the protocol (SPX or NASI) and the other protocols or applications. Use the **packets** option to determine the NASI traffic flow, and use the **error** option as a quick check of failure reasons in NASI connections.

Examples

The following is sample output from the **debug ipx nasi** command using the **packet** and **error** keywords:

```
Router# debug ipx nasi packet

Router# debug ipx nasi error

NASI0: 6E6E Check server info
NASI0: 6E6E sending server-info 4F00   Good response: 43 bytes
NASI0: 7A6E Query Port. Find first
NASI0: FFirst: line 0 DE, port: TTY1-__________ASYNC___^, group: ASYNC___^
NASI0: 7A6E sending Qport find-first response: 300 bytes
NASI0: 7B6E port request. setting up port
NASI: Check-login User: c h r i s
NASI: Check-login PW hash: C7 A6 C5 C7 C4 C0 C5 C3 C4 CC C5 CF C4 C8 C5 CB C4 D4 C5 D7 C4
D0 C5 D3 C4
NASI: Check-login PW: l a b
NASI1: 7B6E sending NCS Good server Data Ack in 0 bytes pkt in 13 size pkt
NASI1: 7B6E sending Preq response: 303 bytes Good
NASI1: 7B6E port request. setting up port
NASI1: 7B6E sending NCS Good server Data Ack in 0 bytes pkt in 13 size pkt
NASI1: 7B6E sending Preq response: 303 bytes Good
NASI1: 7B6E Unknown NASI code 4500 Pkt Size: 13
 45 0 0 FC 0 2 0 20 0 0 FF 1 0
NASI1: 7B6E Flush Rx Buffers
NASI1: 7B6E sending NASI server TTY data: 1 byte in 14 size pkt
NASI1: 7B6E sending NCS Good server Data Ack in 1 bytes pkt in 13 size pkt
```

In the following line, the 0 is the number of the tty to which this NASI connection is attached. TTY 0 is used by all NASI control connections. 6E6E is the associated SPX connection pointer for this NASI connection. “Check server info” is a type of NASI packet that indicates an incoming NASI packet of this type.

```
NASI0: 6E6E Check server info
```

The following message indicates that the router is sending back a “server-info” packet with a positive acknowledgment, and the packet size is 43 bytes:

```
NASI0: 6E6E sending server-info 4F00   Good response: 43 bytes
```

The following line is a NASI packet type. “Find first” and “find next” are NASI packet types.

```
NASI0: 7A6E Query Port. Find first
```

The following line indicates that the outgoing find first packet for the NASI connection 7A6E has line 0 DE, port name TTY1, and general name ASYNC:

```
NASI0: FFirst: line 0 DE, port: TTY1-__________ASYNC___^, group: ASYNC___^
```

The following two lines indicate a received NASI packet for NASI connection on line 1. 7B6E is the NASI connection pointer. The packet code is 4500 and is not recognizable by Cisco devices. The second line is a hexadecimal dump of the packet.

```
NASI1: 7B6E Unknown NASI code 4500 Pkt Size: 13
 45 0 0 FC 0 2 0 20 0 0 FF 1 0
```

Related Commands

Command	Description
debug ipx spx	Displays debugging messages related to the SPX protocol.

debug ipx packet

To display information about packets received, sent, and forwarded, use the **debug ipx packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx packet

no debug ipx packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is useful for learning whether IPX packets are traveling over a router.

Note In order to generate **debug ipx packet** information on all IPX traffic traveling over the router, you must first configure the router so that fast switching is disabled. Use the **no ipx route-cache** command on all interfaces on which you want to observe traffic. If the router is configured for IPX fast switching, only non fast-switched packets will produce output. When the IPX cache is invalidated or cleared, one packet for each destination is displayed as the cache is repopulated.

Examples

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

```
Router# debug ipx packet

IPX: src=160.0260.8c4c.4f22, dst=1.0000.0000.0001, packet received
IPX: src=160.0260.8c4c.4f22, dst=1.0000.0000.0001,gw=183.0000.0c01.5d85,
sending packet
```

The first line indicates that the router receives a packet from a Novell station (address 160.0260.8c4c.4f22); this trace does not indicate the address of the immediate router sending the packet to this router. In the second line, the router forwards the packet toward the Novell server (address 1.0000.0000.0001) through an immediate router (183.0000.0c01.5d85).

Table 105 describes the significant fields shown in the display.

Table 105 debug ipx packet Field Descriptions

Field	Description
IPX	Indicates that this is an IPX packet.
src=160.0260.8c4c.4f22	Source address of the IPX packet. The Novell network number is 160. Its MAC address is 0260.8c4c.4f22.
dst=1.0000.0000.0001	Destination address for the IPX packet. The address 0000.0000.0001 is an internal MAC address, and the network number 1 is the internal network number of a Novell 3.11 server.

Table 105 debug ipx packet Field Descriptions (continued)

Field	Description
packet received	Router received this packet from a Novell station, possibly through an intermediate router.
gw=183.0000.0c01.5d85	Router is sending the packet over to the next hop router; its address of 183.0000.0c01.5d85 was learned from the IPX routing table.
sending packet	Router is attempting to send this packet.

debug ipx routing

To display information on IPX routing packets that the router sends and receives, use the **debug ipx routing** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx routing {**activity** | **events**}

no debug ipx routing {**activity** | **events**}

Syntax Description

activity	Displays messages relating to IPX routing activity.
events	Displays messages relating to IPX routing events.

Usage Guidelines

Normally, a router or server sends out one routing update per minute. Each routing update packet can include up to 50 entries. If many networks exist on the internetwork, the router sends out multiple packets per update. For example, if a router has 120 entries in the routing table, it would send three routing update packets per update. The first routing update packet would include the first 50 entries, the second packet would include the next 50 entries, and the last routing update packet would include the last 20 entries.

Examples

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

```
Router# debug ipx routing

IPXRIP: update from 9999.0260.8c6a.1733
          110801 in 1 hops, delay 2
IPXRIP: sending update to 12FF02:ffff.ffff.ffff via Ethernet 1
          network 555, metric 2, delay 3
          network 1234, metric 3, delay 4
```

Table 106 describes the significant fields in the display.

Table 106 debug ipx routing Field Descriptions

Field	Description
IPXRIP	IPX RIP packet.
update from 9999.0260.8c6a.1733	Routing update packet from an IPX server at address 9999.0260.8c6a.1733.
110801 in 1 hops	Network 110801 is one hop away from the router at address 9999.0260.8c6a.1733.
delay 2	Delay is a time measurement (1/18th second) that the NetWare shell uses to estimate how long to wait for a response from a file server. Also known as ticks.
sending update to 12FF02:ffff.ffff.ffff via Ethernet 1	Router is sending this IPX routing update packet to address 12FF02:ffff.ffff.ffff through Ethernet interface 1.

Table 106 debug ipx routing Field Descriptions (continued)

Field	Description
network 555	Packet includes routing update information for network 555.
metric 2	Network 555 is two metrics (or hops) away from the router.
delay 3	Network 555 is a delay of 3 away from the router. Delay is a measurement that the NetWare shell uses to estimate how long to wait for a response from a file server. Also known as ticks.

Related Commands

Command	Description
debug ipx sap	Displays information about IPX SAP packets.

debug ipx sap

To display information about IPX Service Advertisement Protocol (SAP) packets, use the **debug ipx sap** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx sap [**activity** | **events**]

no debug ipx sap [**activity** | **events**]

Syntax Description

activity	(Optional) Provides more detailed output of SAP packets, including displays of services in SAP packets.
events	(Optional) Limits amount of detailed output for SAP packets to those that contain interesting events.

Usage Guidelines

Normally, a router or server sends out one SAP update per minute. Each SAP packet can include up to seven entries. If many servers are advertising on the network, the router sends out multiple packets per update. For example, if a router has 20 entries in the SAP table, it would send three SAP packets per update. The first SAP would include the first seven entries, the second SAP would include the next seven entries, and the last update would include the last six entries.

Obtain the most meaningful detail by using the **debug ipx sap activity** and the **debug ipx sap events** commands together.

Caution Because the **debug ipx sap** command can generate a substantial amount of output, use it with caution on networks that have many interfaces and large service tables.

Examples

The following is sample output from the **debug ipx sap** command:

```
Router# debug ipx sap

IPXSAP: at 0023F778:
I SAP Response type 0x2 len 160 src:160.0000.0c00.070d dest:160.ffff.ffff.ffff(452)
 type 0x4, "Hello2", 199.0002.0004.0006 (451), 2 hops
 type 0x4, "Hello1", 199.0002.0004.0008 (451), 2 hops
IPXSAP: sending update to 160
IPXSAP: at 00169080:
 O SAP Update type 0x2 len 96 ssoc:0x452 dest:160.ffff.ffff.ffff(452)
IPX: type 0x4, "Magnolia", 42.0000.0000.0001 (451), 2hops
```

The **debug ipx sap** command generates multiple lines of output for each SAP packet—a packet summary message and a service detail message.

The first line displays the internal router memory address of the packet. The technical support staff may use this information in problem debugging.

```
IPXSAP: at 0023F778:
```

Table 107 describes the significant fields shown in the display.

Table 107 debug ipx sap Field Descriptions

Field	Description
I	Indicates whether the router received the SAP packet as input (I) or is sending an update as output (O).
SAP Response type 0x2	Packet type. Format is 0x*n*; possible values for *n* include: • 1—General query • 2—General response • 3—Get Nearest Server request • 4—Get Nearest Server response
len 160	Length of this packet (in bytes).
src: 160.000.0c00.070d	Source address of the packet.
dest:160.ffff.ffff.ffff	IPX network number and broadcast address of the destination IPX network for which the message is intended.
(452)	IPX socket number of the process sending the packet at the source address. This number is always 452, which is the socket number for the SAP process.

Table 108 describes the significant fields shown in the display.

Table 108 debug ipx sap Field Descriptions

Field	Description
type 0x4	Indicates the type of service the server sending the packet provides. Format is 0x*n*. Some of the values for *n* are proprietary to Novell. Those values for *n* that have been published include the following (contact Novell for more information): • 0—Unknown • 1—User • 2—User group • 3—Print queue • 4—File server • 5—Job server • 6—Gateway • 7—Print server • 8—Archive queue • 9—Archive server • A—Job queue • B—Administration • 21—NAS SNA gateway • 24—Remote bridge server • 2D—Time Synchronization VAP • 2E—Dynamic SAP • 47—Advertising print server • 4B—Btrieve VAP 5.0 • 4C—SQL VAP • 7A—TES—NetWare for VMS • 98—NetWare access server • 9A—Named Pipes server • 9E—Portable NetWare—UNIX • 111—Test server • 166—NetWare management • 233—NetWare management agent • 237—NetExplorer NLM • 239—HMI hub • 23A—NetWare LANalyzer agent • 26A—NMS management • FFFF—Wildcard (any SAP service) Contact Novell for more information.

Table 108 debug ipx sap Field Descriptions (continued)

Field	Description
"Hello2"	Name of the server being advertised.
199.0002.0004.0006 (451)	Indicates the network number and address (and socket) of the server generating the SAP packet.
2 hops	Number of hops to the server from the router.

The fifth line of output indicates that the router sent a SAP update to network 160:

```
IPXSAP: sending update to 160
```

The format for **debug ipx sap** output describing a SAP update the router sends is similar to that describing a SAP update the router receives, except that the ssoc: field replaces the src: field, as the following line of output indicates:

```
O SAP Update type 0x2 len 96 ssoc:0x452 dest:160.ffff.ffff.ffff(452)
```

Table 109 describes possible values for the ssoc: field.

Table 109 debug ipx sap Field Descriptions

Field	Description
ssoc:0x452	Indicates the IPX socket number of the process sending the packet at the source address. Possible values include the following: • 451—Network Core Protocol • 452—Service Advertising Protocol • 453—Routing Information Protocol • 455—NetBIOS • 456—Diagnostics • 4000 to 6000—Ephemeral sockets used for interaction with file servers and other network communications

Related Commands

Command	Description
debug ipx routing	Displays information on IPX routing packets that the router sends and receives.

debug ipx spoof

To display information about SPX keepalive and IPX watchdog packets when **ipx watchdog** and **ipx spx-spoof** are configured on the router, use the **debug ipx spoof** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx spoof

no debug ipx spoof

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to troubleshoot connections that use SPX spoofing when SPX keepalive spoofing is enabled.

Examples

The following is sample output from the **debug ipx spoof** command:

```
Router# debug ipx spoof

IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (new) (changed:yes) Last Changed 0
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (new) (changed:yes) Last Changed 0

IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: 80 0 2B8 7104 29 7 7
(early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 42 tc=02, SPX: 80 0 4B8 7004 1D 8 8
(early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 32 tc=02, watchdog
IPX: local:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 32 tc=00, watchdog snet
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
(early)
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
(Last Changed 272 sec)
IPX: local:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, spx keepalive sent 80 0
7104 2B8 7 29 2E
```

The following lines show that SPX packets were seen, but they are not seen for a connection that exists in the SPX table:

```
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (new) (changed:yes) Last Changed 0
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (new) (changed:yes) Last Changed 0
```

The following lines show SPX packets for connections that exist in the SPX table but that SPX idle time has not yet elapsed and spoofing has not started:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: 80 0 2B8 7104 29 7 7
(early)
```

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 42 tc=02, SPX: 80 0 4B8 7004 1D 8 8
(early)
```

The following lines show an IPX watchdog packet and the spoofed reply:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 32 tc=02, watchdog
IPX: local:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 32 tc=00, watchdog sent
```

The following lines show SPX packets that arrived more than two minutes after spoofing started. This situation occurs when the other sides of the SPX table are cleared. When the table is cleared, the routing processes stop spoofing the connection, which allows SPX keepalives from the local side to travel to the remote side and repopulate the SPX table.

```
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
(early)
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (changed:clear) Last Changed 0
```

The following lines show that an SPX keepalive packet came in and was spoofed:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7 7
(Last Changed 272 sec)
IPX: local:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, spx keepalive sent 80 0
7104 2B8 7 29 2E
```

debug ipx spx

To display debugging messages related to the Sequenced Packet Exchange (SPX) protocol, use the **debug ipx spx** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx spx

no debug ipx spx

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug ipx spx** command to display handshaking or negotiating details between the SPX protocol and the other protocols or applications. SPX debugging messages indicate various states of SPX connections such as incoming and outgoing traffic information, timer events, and related processing of SPX connections.

Examples

The following is sample output from the **debug ipx spx** command:

```
Router# debug ipx spx

SPX: Sent an SPX packet
SPX: I Con Src/Dst 776E/20A0 d-strm 0 con-ctl 80
SPX: I Con Src/Dst 776E/20A0 d-strm FE con-ctl 40
SPX: C847C Connection close requested by peer
SPX: Sent an SPX packet
SPX: purge timer fired. Cleaning up C847C
SPX: purging spxcon C847C from conQ
SPX: returning inQ buffers
SPX: returning outQ buffers
SPX: returning unackedQ buffers
SPX: returning spxcon
SPX: I Con Src/Dst 786E/FFFF d-strm 0 con-ctl C0
SPX: new connection request for listening socket
SPX: Sent an SPX packet
SPX: I Con Src/Dst 786E/20B0 d-strm 0 con-ctl 40
SPX: 300 bytes data recvd
SPX: Sent an SPX packet
```

The following line indicates an incoming SPX packet that has a source connection ID of 776E and a destination connection ID of 20A0 (both in hexadecimal). The data stream value in the SPX packet is indicated by d-strm, and the connection control value in the SPX packet is indicated by con-ctl (both in hexadecimal). All data packets received are followed by an SPX debug message indicating the size of the packet. All control packets received are consumed internally.

```
SPX: I Con Src/Dst 776E/20A0 d-strm 0 con-ctl 80
```

The following lines indicate that SPX is attempting to remove an SPX connection that has the address C8476 from its list of connections:

```
SPX: purge timer fired. Cleaning up C847C
SPX: purging spxcon C847C from conQ
```

Related Commands

Command	Description
debug ipx nasi	Displays information about the NASI connections.

debug isdn event

To display ISDN events occurring on the user side (on the router) of the ISDN interface, use the **debug isdn event** privileged EXEC command. The **no** form of this command disables debugging output.

debug isdn event

no debug isdn event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Although the **debug isdn event** and the **debug isdn q931** commands provide similar debug information, the information is displayed in a different format. If you want to see the information in both formats, enable both commands at the same time. The displays will be intermingled.

The ISDN events that can be displayed are Q.931 events (call setup and teardown of ISDN network connections).

Use the **show dialer** command to retrieve information about the status and configuration of the ISDN interface on the router.

Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, see Appendix B, “ISDN Switch Types, Codes, and Values.”

Examples

The following is sample output from the **debug isdn event** command of call setup events for an outgoing call:

```
Router# debug isdn event

ISDN Event: Call to 415555121202
received HOST_PROCEEDING
 Channel ID i = 0x0101
 -------------------
 Channel ID i = 0x89
received HOST_CONNECT
 Channel ID i = 0x0101
ISDN Event: Connected to 415555121202 on B1 at 64 Kb/s
```

The following shows sample **debug isdn event** output of call setup events for an incoming call. The values used for internal purposes are unpacked information elements. The values that follow the ISDN specification are an interpretation of the unpacked information elements. See Appendix B, “ISDN Switch Types, Codes, and Values,” for information about these values.

```
Router# debug isdn event

received HOST_INCOMING_CALL
 Bearer Capability i = 0x080010
 -------------------
 Channel ID i = 0x0101
 Calling Party Number i = 0x0000, ‘415555121202’
 IE out of order or end of ‘private’ IEs --
 Bearer Capability i = 0x8890
```

```
 Channel ID i = 0x89
 Calling Party Number i = 0x0083, '415555121202'
ISDN Event: Received a call from 415555121202 on B1 at 64 Kb/s
ISDN Event: Accepting the call
received HOST_CONNECT
 Channel ID i = 0x0101
ISDN Event: Connected to 415555121202 on B1 at 64 Kb/s
```

The following is sample output from the **debug isdn event** command of call teardown events for a call that has been disconnected by the host side of the connection:

```
Router# debug isdn event

received HOST_DISCONNECT
ISDN Event: Call to 415555121202 was hung up
```

The following is sample output from the **debug isdn event** command of a call teardown event for an outgoing or incoming call that has been disconnected by the ISDN interface on the router side:

```
Router# debug isdn event

ISDN Event: Hangup call to call id 0x8008
```

Table 110 describes the significant fields shown in the display.

Table 110 debug isdn event Field Descriptions

Field	Description
Bearer Capability	Indicates the requested bearer service to be provided by the network. See Table B-4 in the "ISDN Switch Types, Codes, and Values" appendix for detailed information about bearer capability values.
i=	Indicates the information element identifier. The value depends on the field it is associated with. Refer to the ITU-T Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.
Channel ID	Channel Identifier. The values and corresponding channels might be identified in several ways: • Channel ID i=0x0101—Channel B1 • Channel ID i=0x0102—Channel B2 ITU-T Q.931 defines the values and channels as exclusive or preferred: • Channel ID i=0x83—Any B channel • Channel ID i=0x89—Channel B1 (exclusive) • Channel ID i=0x8A—Channel B2 (exclusive) • Channel ID i=0x81—B1 (preferred) • Channel ID i=0x82—B2 (preferred)
Calling Party Number	Identifies the called party. This field is only present in outgoing calls. The Calling Party Number field uses the IA5 character set. Note that it may be replaced by the Keypad facility field.

Table 110 debug isdn event Field Descriptions (continued)

Field	Description
IE out of order or end of 'private' IEs	Indicates that an information element identifier is out of order or there are no more private network information element identifiers to interpret.
Received a call from 415555121202 on B1 at 64 Kb/s	Identifies the origin of the call. This field is present only in incoming calls. Note that the information about the incoming call includes the channel and speed. Whether the channel and speed are displayed depends on the network delivering the calling party number.

The following is sample output from the **debug isdn event** command of a call teardown event for a call that has passed call screening and then has been hung up by the ISDN interface on the far end side:

```
Router# debug isdn event

Jan  3 11:29:52.559: ISDN BR0: RX <-  DISCONNECT pd = 8  callref = 0x81
Jan  3 11:29:52.563:         Cause i = 0x8090 - Normal call clearing
```

The following is sample output from the **debug isdn event** command of a call teardown event for a call that has not passed call screening and has been rejected by the ISDN interface on the router side:

```
Router# debug isdn event

Jan  3 11:32:03.263: ISDN BR0: RX <-  DISCONNECT pd = 8  callref = 0x85
Jan  3 11:32:03.267:         Cause i = 0x8095 - Call rejected
```

The following is sample output from the **debug isdn event** command of a call teardown event for an outgoing call that uses a dialer subaddress:

```
Router# debug isdn event

Jan  3 11:41:48.483: ISDN BR0: Event: Call to 61885:1212 at 64 Kb/s
Jan  3 11:41:48.495: ISDN BR0: TX ->  SETUP pd = 8  callref = 0x04
Jan  3 11:41:48.495:         Bearer Capability i = 0x8890
Jan  3 11:41:48.499:         Channel ID i = 0x83
Jan  3 11:41:48.503:         Called Party Number i = 0x80, '61885'
Jan  3 11:41:48.507:         Called Party SubAddr i = 0x80, 'P1212'
Jan  3 11:41:48.571: ISDN BR0: RX <-  CALL_PROC pd = 8  callref = 0x84
Jan  3 11:41:48.575:         Channel ID i = 0x89
Jan  3 11:41:48.587: ISDN BR0: Event: incoming ces value = 1
Jan  3 11:41:48.587: ISDN BR0: received HOST_PROCEEDING
                        Channel ID i = 0x0101
Jan  3 11:41:48.591:    -------------------
                        Channel ID i = 0x89
Jan  3 11:41:48.731: ISDN BR0: RX <-  CONNECT pd = 8  callref = 0x84
Jan  3 11:41:48.743: ISDN BR0: Event: incoming ces value = 1
Jan  3 11:41:48.743: ISDN BR0: received HOST_CONNECT
                        Channel ID i = 0x0101
Jan  3 11:41:48.747:    -------------------
%LINK-3-UPDOWN: Interface BRI0:1 changed state to up
Jan  3 11:41:48.771: ISDN BR0: Event: Connected to 61885:1212 on B1 at 64 Kb/s
Jan  3 11:41:48.775: ISDN BR0: TX ->  CONNECT_ACK pd = 8  callref = 0x04
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI0:1 is now connected to 61885:1212 goodie
```

The output is similar to the output of **debug isdn q931**. Refer to the **debug isdn q931** command for detailed field descriptions.

The following is sample output from the **debug isdn event** command of call setup events for a successful callback for legacy DDR:

```
Router# debug isdn event

BRI0:Caller id Callback server starting to spanky 81012345678902
: Callback timer expired
BRI0:beginning callback to spanky 81012345678902
BRI0: Attempting to dial 81012345678902
```

The following is sample output from the **debug isdn event** command for a callback that was unsuccessful because the router had no dialer map for the calling number:

```
Router# debug isdn event

BRI0:Caller id 81012345678902 callback - no matching map
```

Table 111 describes the significant fields shown in the display.

Table 111 debug isdn event Field Descriptions for Caller ID Callback and Legacy DDR

Field	Description
BRI0:Caller id Callback server starting to ...	Caller ID callback has started, plus host name and number called. The callback enable timer starts now.
: Callback timer expired	Callback timer has expired; callback can proceed.
BRI0:beginning callback to ... BRI0: Attempting to dial ...	Actions proceeding after the callback timer expired, plus host name and number called.

The following is sample output from the **debug isdn event** command for a callback that was successful when the dialer profiles DDR feature is configured:

```
*Mar  1 00:46:51.827: BR0:1:Caller id 81012345678901 matched to profile delorean
*Mar  1 00:46:51.827: Dialer1:Caller id Callback server starting to delorean
81012345678901
*Mar  1 00:46:54.151: : Callback timer expired
*Mar  1 00:46:54.151: Dialer1:beginning callback to delorean 81012345678901
*Mar  1 00:46:54.155: Freeing callback to delorean 81012345678901
*Mar  1 00:46:54.155: BRI0: Dialing cause Callback return call
*Mar  1 00:46:54.155: BRI0: Attempting to dial 81012345678901
*Mar  1 00:46:54.503: %LINK-3-UPDOWN: Interface BRI0:2, changed state to up
*Mar  1 00:46:54.523: %DIALER-6-BIND: Interface BRI0:2 bound to profile Dialer1
*Mar  1 00:46:55.139: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:2, changed
state to up
*Mar  1 00:46:58.187: %ISDN-6-CONNECT: Interface BRI0:2 is now connected to 81012345678901
delorean
```

Table 112 describes significant fields of call setup events for a successful callback for the sample output from the **debug isdn event** command when the dialer profiles DDR feature is configured.

Table 112 debug isdn event Field Descriptions for Caller ID Callback and Dialer Profiles

Field	Description
BR0:1:Caller id ... matched to profile ...	Interface, channel number, caller ID that are matched, and the profile to bind to the interface.
: Callback timer expired	Callback timer has expired; callback can proceed.
Dialer1:beginning callback to...	Callback process is beginning to the specified number.
Freeing callback to...	Callback has been started to the specified number, and the number has been removed from the callback list.
BRI0: Dialing cause Callback return call BRI0: Attempting to dial	The reason for the call and the number being dialed.
%LINK-3-UPDOWN: Interface BRI0:2, changed state to up	Interface status: up.
%DIALER-6-BIND: Interface BRI0:2 bound to profile Dialer1	Profile bound to the interface.
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:2, changed state to up	Line protocol status: up.
%ISDN-6-CONNECT: Interface BRI0:2 is now connected to ...	Interface is now connected to the specified host and number.

debug isdn q921

To display data link layer (Layer 2) access procedures that are taking place at the router on the D channel (LAPD) of its ISDN interface, use the **debug isdn q921** privileged EXEC command. The **no** form of this command disables debugging output.

debug isdn q921

no debug isdn q921

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The ISDN data link layer interface provided by the router conforms to the user interface specification defined by ITU-T recommendation Q.921. The **debug isdn q921** command output is limited to commands and responses exchanged during peer-to-peer communication carried over the D channel. This debug information does not include data sent over the B channels that is also part of the router's ISDN interface. The peers (data link layer entities and layer management entities on the routers) communicate with each other via an ISDN switch over the D channel.

Note The ISDN switch provides the network interface defined by Q.921. This **debug** command does not display data link layer access procedures taking place within the ISDN network (that is, procedures taking place on the network side of the ISDN connection). See Appendix B, "ISDN Switch Types, Codes, and Values," for a list of the supported ISDN switch types.

A router can be the calling or called party of the ISDN Q.921 data link layer access procedures. If the router is the calling party, the command displays information about an outgoing call. If the router is the called party, the command displays information about an incoming call and the keepalives.

The **debug isdn q921** command can be used with the **debug isdn event** and the **debug isdn q931** commands at the same time. The displays will be intermingled.

Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, see Appendix B, "ISDN Switch Types, Codes, and Values."

Examples

The following is sample output from the **debug isdn q921** command for an outgoing call:

```
Router# debug isdn q921

Jan  3 14:52:24.475: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 5  nr = 2
                              i = 0x08010705040288901801837006803631383835
Jan  3 14:52:24.503: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 6
Jan  3 14:52:24.527: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 2  nr = 6
                              i = 0x08018702180189
Jan  3 14:52:24.535: ISDN BR0: TX ->  RRr sapi = 0  tei = 64  nr = 3
Jan  3 14:52:24.643: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 3  nr = 6
                              i = 0x08018707
Jan  3 14:52:24.655: ISDN BR0: TX ->  RRr sapi = 0  tei = 64  nr = 4
%LINK-3-UPDOWN: Interface BRI0:1, changed state to up
Jan  3 14:52:24.683: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 6  nr = 4
```

```
                              i = 0x0801070F
Jan  3 14:52:24.699: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 7
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI0:1 is now connected to 61885 goodie
Jan  3 14:52:34.415: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 7
Jan  3 14:52:34.419: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 4
```

In the following lines, the seventh and eighth most significant hexadecimal numbers indicate the type of message. 0x05 indicates a Call Setup message, 0x02 indicates a Call Proceeding message, 0x07 indicates a Call Connect message, and 0x0F indicates a Connect Ack message.

```
Jan  3 14:52:24.475: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 5  nr = 2
                              i = 0x08010705040288901801837006803631383835
Jan  3 14:52:24.527: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 2  nr = 6
                              i = 0x08018702180189
Jan  3 14:52:24.643: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 3  nr = 6
                              i = 0x08018707
Jan  3 14:52:24.683: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 6  nr = 4
                              i = 0x0801070F
```

The following is sample output from the **debug isdn q921** command for a startup message on a DMS-100 switch:

```
Router# debug isdn q921

Jan  3 14:47:28.455: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127 0
Jan  3 14:47:30.171: ISDN BR0: TX ->  IDREQ  ri = 31815  ai = 127
Jan  3 14:47:30.219: ISDN BR0: RX <-  IDASSN  ri = 31815  ai = 64
Jan  3 14:47:30.223: ISDN BR0: TX ->  SABMEp sapi = 0  tei = 64
Jan  3 14:47:30.227: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127
Jan  3 14:47:30.235: ISDN BR0: TX ->  IDCKRP  ri = 16568  ai = 64
Jan  3 14:47:30.239: ISDN BR0: RX <-  UAf sapi = 0  tei = 64
Jan  3 14:47:30.247: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 0  nr = 0
                              i = 0x08007B3A03313233
Jan  3 14:47:30.267: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 1
Jan  3 14:47:34.243: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 1  nr = 0
                              i = 0x08007B3A03313233
Jan  3 14:47:34.267: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 2
Jan  3 14:47:43.815: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 2
Jan  3 14:47:43.819: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
Jan  3 14:47:53.819: ISDN BR0: TX ->  RRp sapi = 0  tei = 64 nr = 0
```

The first seven lines of this example indicate a Layer 2 link establishment.

The following lines indicate the message exchanges between the data link layer entity on the local router (user side) and the assignment source point (ASP) on the network side during the TEI assignment procedure. This assumes that the link is down and no TEI currently exists.

```
Jan  3 14:47:30.171: ISDN BR0: TX ->  IDREQ  ri = 31815  ai = 127
Jan  3 14:47:30.219: ISDN BR0: RX <-  IDASSN  ri = 31815  ai = 64
```

At 14:47:30.171, the local router data link layer entity sent an Identity Request message to the network data link layer entity to request a TEI value that can be used in subsequent communication between the peer data link layer entities. The request includes a randomly generated reference number (31815) to differentiate among user devices that request automatic TEI assignment and an action indicator of 127 to indicate that the ASP can assign any TEI value available. The ISDN user interface on the router uses automatic TEI assignment.

At 14:47:30.219, the network data link entity responds to the Identity Request message with an Identity Assigned message. The response includes the reference number (31815) previously sent in the request and TEI value (64) assigned by the ASP.

The following lines indicate the message exchanges between the layer management entity on the network and the layer management entity on the local router (user side) during the TEI check procedure:

```
Jan  3 14:47:30.227: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127
Jan  3 14:47:30.235: ISDN BR0: TX ->  IDCKRP  ri = 16568  ai = 64
```

At 14:47:30.227, the layer management entity on the network sends the Identity Check Request message to the layer management entity on the local router to check whether a TEI is in use. The message includes a reference number that is always 0 and the TEI value to check. In this case, an ai value of 127 indicates that all TEI values should be checked. At 14:47:30.227, the layer management entity on the local router responds with an Identity Check Response message indicating that TEI value 64 is currently in use.

The following lines indicate the messages exchanged between the data link layer entity on the local router (user side) and the data link layer on the network side to place the network side into modulo 128 multiple frame acknowledged operation. Note that the data link layer entity on the network side also can initiate the exchange.

```
Jan  3 14:47:30.223: ISDN BR0: TX ->  SABMEp sapi = 0  tei = 64
Jan  3 14:47:30.239: ISDN BR0: RX <-  UAf sapi = 0  tei = 64
```

At 14:47:30.223, the data link layer entity on the local router sends the SABME command with a SAPI of 0 (call control procedure) for TEI 64. At 14:47:30.239, the first opportunity, the data link layer entity on the network responds with a UA response. This response indicates acceptance of the command. The data link layer entity sending the SABME command may need to send it more than once before receiving a UA response.

The following lines indicate the status of the data link layer entities. Both are ready to receive I frames.

```
Jan  3 14:47:43.815: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 2
Jan  3 14:47:43.819: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
```

These I-frames are typically exchanged every 10 seconds (T203 timer).

The following is sample output from the **debug isdn q921** command for an incoming call. It is an incoming SETUP message that assumes that the Layer 2 link is already established to the other side.

```
Router# debug isdn q921

Jan  3 14:49:22.507: ISDN BR0: TX ->  RRp sapi = 0  tei = 64 nr = 0
Jan  3 14:49:22.523: ISDN BR0: RX <-  RRf sapi = 0  tei = 64  nr = 2
Jan  3 14:49:32.527: ISDN BR0: TX ->  RRp sapi = 0  tei = 64 nr = 0
Jan  3 14:49:32.543: ISDN BR0: RX <-  RRf sapi = 0  tei = 64  nr = 2
Jan  3 14:49:42.067: ISDN BR0: RX <-  RRp sapi = 0  tei = 64 nr = 2
Jan  3 14:49:42.071: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
Jan  3 14:49:47.307: ISDN BR0: RX <-  UI sapi = 0  tei = 127
                              i = 0x08011F05040288901801897006C13631383836
%LINK-3-UPDOWN: Interface BRI0:1, changed state to up
Jan  3 14:49:47.347: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 2  nr = 0
                              i = 0x08019F07180189
Jan  3 14:49:47.367: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 3
Jan  3 14:49:47.383: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 0  nr = 3
                              i = 0x08011F0F180189
Jan  3 14:49:47.391: ISDN BR0: TX ->  RRr sapi = 0  tei = 64  nr = 1
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
```

Table 113 describes the significant fields shown in the display.

Table 113 debug isdn q921 Field Descriptions

Field	Description
Jan 3 14:49:47.391	Indicates the date and time at which the frame was sent from or received by the data link layer entity on the router. The time is maintained by an internal clock.
TX	Indicates that this frame is being sent from the ISDN interface on the local router (user side).
RX	Indicates that this frame is being received by the ISDN interface on the local router from the peer (network side).
IDREQ	Indicates the Identity Request message type sent from the local router to the network (ASP) during the automatic TEI assignment procedure. This message is sent in a UI command frame. The SAPI value for this message type is always 63 (indicating that it is a Layer 2 management procedure) but it is not displayed. The TEI value for this message type is 127 (indicating that it is a broadcast operation).
ri = 31815	Indicates the Reference number used to differentiate between user devices requesting TEI assignment. This value is a randomly generated number from 0 to 65535. The same ri value sent in the IDREQ message should be returned in the corresponding IDASSN message. Note that a Reference number of 0 indicates that the message is sent from the network side management layer entity and a reference number has not been generated.
ai = 127	Indicates the Action indicator used to request that the ASP assign any TEI value. It is always 127 for the broadcast TEI. Note that in some message types, such as IDREM, a specific TEI value is indicated.
IDREM	Indicates the Identity Remove message type sent from the ASP to the user side layer management entity during the TEI removal procedure. This message is sent in a UI command frame. The message includes a reference number that is always 0, because it is not responding to a request from the local router. The ASP sends the Identity Remove message twice to avoid message loss.
IDASSN	Indicates the Identity Assigned message type sent from the ISDN service provider on the network to the local router during the automatic TEI assignment procedure. This message is sent in a UI command frame. The SAPI value for this message type is always 63 (indicating that it is a Layer 2 management procedure). The TEI value for this message type is 127 (indicating it is a broadcast operation).
ai = 64	Indicates the TEI value automatically assigned by the ASP. This TEI value is used by data link layer entities on the local router in subsequent communication with the network. The valid values are in the range from 64 to 126.

Table 113 debug isdn q921 Field Descriptions (continued)

Field	Description
SABME	Indicates the set asynchronous balanced mode extended command. This command places the recipient into modulo 128 multiple frame acknowledged operation. This command also indicates that all exception conditions have been cleared. The SABME command is sent once a second for N200 times (typically three times) until its acceptance is confirmed with a UA response. For a list and brief description of other commands and responses that can be exchanged between the data link layer entities on the local router and the network, see ITU-T Recommendation Q.921.
sapi = 0	Identifies the service access point at which the data link layer entity provides services to Layer 3 or to the management layer. A SAPI with the value 0 indicates it is a call control procedure. Note that the Layer 2 management procedures such as TEI assignment, TEI removal, and TEI checking, which are tracked with the **debug isdn q921** command, do not display the corresponding SAPI value; it is implicit. If the SAPI value were displayed, it would be 63.
tei = 64	Indicates the TEI value automatically assigned by the ASP. This TEI value will be used by data link layer entities on the local router in subsequent communication with the network. The valid values are in the range from 64 to 126.
IDCKRQ	Indicates the Identity Check Request message type sent from the ISDN service provider on the network to the local router during the TEI check procedure. This message is sent in a UI command frame. The ri field is always 0. The ai field for this message contains either a specific TEI value for the local router to check or 127, which indicates that the local router should check all TEI values. For a list and brief description of other message types that can be exchanged between the local router and the ISDN service provider on the network, see Appendix B, "ISDN Switch Types, Codes, and Values."
IDCKRP	Indicates the Identity Check Response message type sent from the local router to the ISDN service provider on the network during the TEI check procedure. This message is sent in a UI command frame in response to the IDCKRQ message. The ri field is a randomly generated number from 0 to 65535. The ai field for this message contains the specific TEI value that has been checked.
UAf	Confirms that the network side has accepted the SABME command previously sent by the local router. The final bit is set to 1.
INFOc	Indicates that this is an Information command. It is used to transfer sequentially numbered frames containing information fields that are provided by Layer 3. The information is transferred across a data-link connection.
INFORMATION pd = 8 callref = (null)	Indicates the information fields provided by Layer 3. The information is sent one frame at a time. If multiple frames need to be sent, several Information commands are sent. The pd value is the protocol discriminator. The value 8 indicates it is call control information. The call reference number is always null for SPID information.

Table 113 debug isdn q921 Field Descriptions (continued)

Field	Description
SPID information i = 0x343135393033383336363031	Indicates the SPID. The local router sends this information to the ISDN switch to indicate the services to which it subscribes. SPIDs are assigned by the service provider and are usually 10-digit telephone numbers followed by optional numbers. Currently, only the DMS-100 switch supports SPIDs, one for each B channel. If SPID information is sent to a switch type other than DMS-100, an error may be displayed in the debug information.
ns = 0	Indicates the send sequence number of sent I frames.
nr = 0	Indicates the expected send sequence number of the next received I frame. At time of transmission, this value should be equal to the value of ns. The value of nr is used to determine whether frames need to be re-sent for recovery.
RRr	Indicates the Receive Ready response for unacknowledged information transfer. The RRr is a response to an INFOc.
RRp	Indicates the Receive Ready command with the poll bit set. The data link layer entity on the user side uses the poll bit in the frame to solicit a response from the peer on the network side.
RRf	Indicates the Receive Ready response with the final bit set. The data link layer entity on the network side uses the final bit in the frame to indicate a response to the poll.
sapi	Indicates the service access point identifier. The SAPI is the point at which data link services are provided to a network layer or management entity. Currently, this field can have the value 0 (for call control procedure) or 63 (for Layer 2 management procedures).
tei	Indicates the terminal endpoint identifier (TEI) that has been assigned automatically by the assignment source point (ASP) (also called the layer management entity on the network side). The valid range is from 64 to 126. The value 127 indicates a broadcast.

debug isdn q931

To display information about call setup and teardown of ISDN network connections (layer 3) between the local router (user side) and the network, use the **debug isdn q931** privileged EXEC command. The **no** form of this command disables debugging output.

debug isdn q931

no debug isdn q931

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The ISDN network layer interface provided by the router conforms to the user interface specification defined by ITU-T recommendation Q.931, supplemented by other specifications such as for switch type VN4. The router tracks only activities that occur on the user side, not the network side, of the network connection. The display information **debug isdn q931** command output is limited to commands and responses exchanged during peer-to-peer communication carried over the D channel. This debug information does not include data sent over the B channels, which are also part of the router's ISDN interface. The peers (network layers) communicate with each other via an ISDN switch over the D channel.

A router can be the calling or called party of the ISDN Q.931 network connection call setup and teardown procedures. If the router is the calling party, the command displays information about an outgoing call. If the router is the called party, the command displays information about an incoming call.

You can use the **debug isdn q931** command with the **debug isdn event** and the **debug isdn q921** commands at the same time. The displays will be intermingled. Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, refer to Appendix B, "ISDN Switch Types, Codes, and Values."

Examples

The following is sample output from the **debug isdn q931** command of a call setup procedure for an outgoing call:

```
Router# debug isdn q931

TX -> SETUP pd = 8 callref = 0x04
 Bearer Capability i = 0x8890
 Channel ID i = 0x83
 Called Party Number i = 0x80, '415555121202'
RX <- CALL_PROC pd = 8 callref = 0x84
 Channel ID i = 0x89
RX <- CONNECT pd = 8 callref = 0x84
TX -> CONNECT_ACK pd = 8 callref = 0x04....
Success rate is 0 percent (0/5)
```

The following is sample output from the **debug isdn q931** command of a call setup procedure for an incoming call:

```
Router# debug isdn q931

RX <- SETUP pd = 8 callref = 0x06
 Bearer Capability i = 0x8890
```

```
 Channel ID i = 0x89
 Calling Party Number i = 0x0083, '81012345678902'
TX -> CONNECT pd = 8 callref = 0x86
RX <- CONNECT_ACK pd = 8 callref = 0x06
```

The following is sample output from the **debug isdn q931** command of a call teardown procedure from the network:

```
Router# debug isdn q931

RX <- DISCONNECT pd = 8 callref = 0x84
 Cause i = 0x8790
 Looking Shift to Codeset 6
 Codeset 6 IE 0x1 1 0x82 '10'
TX -> RELEASE pd = 8 callref = 0x04
 Cause i = 0x8090
RX <- RELEASE_COMP pd = 8 callref = 0x84
```

The following is sample output from the **debug isdn q931** command of a call teardown procedure from the router:

```
Router# debug isdn q931

TX -> DISCONNECT pd = 8 callref = 0x05
 Cause i = 0x879081
RX <- RELEASE pd = 8 callref = 0x85
 Looking Shift to Codeset 6
 Codeset 6 IE 0x1 1 0x82 '10'
TX <- RELEASE_COMP pd = 8 callref = 0x05
```

Table 114 describes the significant fields shown in the display.

Table 114 debug isdn q931 Command Call Setup Procedure Field Descriptions

Field	Description
TX ->	Indicates that this message is being sent from the local router (user side) to the network side of the ISDN interface.
RX <-	Indicates that this message is being received by the user side of the ISDN interface from the network side.
SETUP	Indicates that the SETUP message type has been sent to initiate call establishment between peer network layers. This message can be sent from either the local router or the network.
pd	Indicates the protocol discriminator. The protocol discriminator distinguishes messages for call control over the user-network ISDN interface from other ITU-T-defined messages, including other Q.931messages. The protocol discriminator is 8 for call control messages such as SETUP. For basic-1tr6, the protocol discriminator is 65.
callref	Indicates the call reference number in hexadecimal notation. The value of this field indicates the number of calls made from either the router (outgoing calls) or the network (incoming calls). Note that the originator of the SETUP message sets the high-order bit of the call reference number to 0. The destination of the connection sets the high-order bit to 1 in subsequent call control messages, such as the CONNECT message. For example, callref = 0x04 in the request becomes callref = 0x84 in the response.

Table 114 debug isdn q931 Command Call Setup Procedure Field Descriptions (continued)

Field	Description
Bearer Capability	Indicates the requested bearer service to be provided by the network. Refer to Table B-4 in Appendix B, "ISDN Switch Types, Codes, and Values," for detailed information about bearer capability values.
i =	Indicates the information element identifier. The value depends on the field it is associated with. Refer to the ITU-T Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.
Channel ID	Indicates the channel identifier. The value 83 indicates any channel, 89 indicates the B1 channel, and 8A indicates the B2 channel. For more information about the Channel Identifier, refer to ITU-T Recommendation Q.931.
Called Party Number	Identifies the called party. This field is only present in outgoing SETUP messages. Note that it can be replaced by the Keypad facility field. This field uses the IA5 character set.
Calling Party Number	Identifies the origin of the call. This field is present only in incoming SETUP messages. This field uses the IA5 character set.
CALL_PROC	Indicates the CALL PROCEEDING message; the requested call setup has begun and no more call setup information will be accepted.
CONNECT	Indicates that the called user has accepted the call.
CONNECT_ACK	Indicates that the calling user acknowledges the called user's acceptance of the call.
DISCONNECT	Indicates either that the user side has requested the network to clear an end-to-end connection or that the network has cleared the end-to-end connection.
Cause	Indicates the cause of the disconnect. Refer to Table B-2 and Table B-3 in Appendix B, "ISDN Switch Types, Codes, and Values," for detailed information about DISCONNECT cause codes and RELEASE cause codes.
Looking Shift to Codeset 6	Indicates that the next information elements will be interpreted according to information element identifiers assigned in codeset 6. Codeset 6 means that the information elements are specific to the local network.
Codeset 6 IE 0x1 i = 0x82, '10'	Indicates charging information. This information is specific to the NTT switch type and may not be sent by other switch types.
RELEASE	Indicates that the sending equipment will release the channel and call reference. The recipient of this message should prepare to release the call reference and channel.
RELEASE_COMP	Indicates that the sending equipment has received a RELEASE message and has now released the call reference and channel.

debug isis adj packets

To display information on all adjacency-related activity such as hello packets sent and received and IS-IS adjacencies going up and down, use the **debug isis adj packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug isis adj packets

no debug isis adj packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug isis adj packets

ISIS-Adj: Rec L1 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
ISIS-Adj: Rec L2 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
ISIS-Adj: Rec L1 IIH from 0000.0c00.0c36 (Ethernet1), cir type 3, cir id CCCC.CCCC.CCCC.03
ISIS-Adj: Area mismatch, level 1 IIH on Ethernet1
ISIS-Adj: Sending L1 IIH on Ethernet1
ISIS-Adj: Sending L2 IIH on Ethernet1
ISIS-Adj: Rec L2 IIH from 0000.0c00.0c36 (Ethernet1), cir type 3, cir id BBBB.BBBB.BBBB.03
```

The following line indicates that the router received an IS-IS hello packet (IIH) on Ethernet interface 0 from the Level 1 router (L1) at MAC address 0000.0c00.40af. The circuit type is the interface type: 1—Level 1 only; 2—Level 2 only; 3—Level 1/2.

The circuit ID is what the neighbor interprets as the designated router for the interface.

```
ISIS-Adj: Rec L1 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
```

The following line indicates that the router (configured as a Level 1 router) received on Ethernet interface 1 is an IS-IS hello packet from a Level 1 router in another area, thereby declaring an area mismatch:

```
ISIS-Adj: Area mismatch, level 1 IIH on Ethernet1
```

The following lines indicates that the router (configured as a Level 1/Level 2 router) sent on Ethernet interface 1 is a Level 1 IS-IS hello packet, and then a Level 2 IS-IS packet:

```
ISIS-Adj: Sending L1 IIH on Ethernet1
ISIS-Adj: Sending L2 IIH on Ethernet1
```

debug isis mpls traffic-eng advertisements

To print information about traffic engineering advertisements in ISIS Link-state advertisement (LSA) messages, use the **debug isis mpls traffic-eng advertisements** EXEC command. To disable debugging output, use the **no** form of this command.

debug isis mpls traffic-eng advertisements

[**no debug isis mpls traffic-eng advertisements**

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information about traffic engineering advertisements is printed in ISIS LSA messages:

```
debug isis mpls traffic-eng advertisements

System ID:Router1.00
  Router ID:10.106.0.6
  Link Count:1
    Link[1]
      Neighbor System ID:Router2.00 (P2P link)
      Interface IP address:10.42.0.6
      Neighbor IP Address:10.42.0.10
      Admin. Weight:10
      Physical BW:155520000 bits/sec
      Reservable BW:5000000 bits/sec
      BW unreserved[0]:2000000 bits/sec, BW unreserved[1]:100000 bits/sec
      BW unreserved[2]:100000 bits/sec, BW unreserved[3]:100000 bits/sec
      BW unreserved[4]:100000 bits/sec, BW unreserved[5]:100000 bits/sec
      BW unreserved[6]:100000 bits/sec, BW unreserved[7]:0 bits/sec
      Affinity Bits:0x00000000
```

Table 115 describes the significant fields shown in the display.

Table 115 debug isis mpls traffic-eng advertisements Field Descriptions

Field	Description
System ID	Identification value for the local system in the area.
Router ID	MPLS traffic engineering router ID.

Table 115 debug isis mpls traffic-eng advertisements Field Descriptions (continued)

Field	Description
Link Count	Number of links that MPLS traffic engineering advertised.
Neighbor System ID	Identification value for the remote system in an area.
Interface IP address	IPv4 address of the interface.
Neighbor IP Address	IPv4 address of the neighbor.
Admin. Weight	Administrative weight associated with this link.
Physical BW	Bandwidth capacity of the link (in bits per second).
Reservable BW	Amount of reservable bandwidth on this link.
BW unreserved	Amount of bandwidth that is available for reservation.
Affinity Bits	Attribute flags of the link that are being flooded.

debug isis mpls traffic-eng events

To print information about traffic engineering-related ISIS events, use the **debug isis mpls traffic-eng events** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug isis mpls traffic-eng events

no debug isis mpls traffic-eng events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about traffic engineering-related ISIS events:

```
debug isis mpls traffic-eng events

ISIS-RRR:Send MPLS TE Et4/0/1 Router1.02 adjacency down:address 0.0.0.0
ISIS-RRR:Found interface address 10.1.0.6 Router1.02, building subtlv... 58 bytes
ISIS-RRR:Found interface address 10.42.0.6 Router2.00, building subtlv... 64 bytes
ISIS-RRR:Interface address 0.0.0.0 Router1.00 not found, not building subtlv
ISIS-RRR:LSP Router1.02 changed from 0x606BCD30
ISIS-RRR:Mark LSP Router1.02 changed because TLV contents different, code 16
ISIS-RRR:Received 1 MPLS TE links flood info for system id Router1.00
```

debug isis spf statistics

To display statistical information about building routes between intermediate systems (ISs), use the **debug isis spf statistics** privileged EXEC command. The **no** form of this command disables debugging output.

debug isis spf statistics

no debug isis spf statistics

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The Intermediate System-to-Intermediate System (IS-IS) Interdomain Routing Protocol (IDRP) provides routing between ISs by flooding the network with link-state information. IS-IS provides routing at two levels, intra-area (Level 1) and intra-domain (Level 2). Level 1 routing allows Level 1 ISs to communicate with other Level 1 ISs in the same area. Level 2 routing allows Level 2 ISs to build an interdomain backbone between Level 1 areas by traversing only Level 2 ISs. Level 1 ISs only need to know the path to the nearest Level 2 IS in order to take advantage of the interdomain backbone created by the Level 2 ISs.

The IS-IS protocol uses the SPF routing algorithm to build Level 1 and Level 2 routes. The **debug isis spf statistics** command provides information for determining the time required to place a Level 1 IS or Level 2 IS on the shortest path tree (SPT) using the IS-IS protocol.

Note The SPF algorithm is also called the Dijkstra algorithm, after the creator of the algorithm.

Examples

The following is sample output from the **debug isis spf statistics** command:

```
Router# debug isis spf statistics

ISIS-Stats: Compute L1 SPT, Timestamp 2780.328 seconds
ISIS-Stats: Complete L1 SPT, Compute time 0.004, 1 nodes on SPT
ISIS-Stats: Compute L2 SPT, Timestamp 2780.3336 seconds
ISIS-Stats: Complete L2 SPT, Compute time 0.056, 12 nodes on SPT
```

Table 116 describes the significant fields shown in the display.

Table 116 debug isis spf statistics Field Descriptions

Field	Description
Compute L1 SPT	Indicates that Level 1 ISs are to be added to a Level 1 area.
Timestamp	Indicates the time at which the SPF algorithm was applied. The time is expressed as the number of seconds elapsed since the system was up and configured.
Complete L1 SPT	Indicates that the algorithm has completed for Level 1 routing.
Compute time	Indicates the time required to place the ISs on the SPT.

Table 116 debug isis spf statistics Field Descriptions (continued)

Field	Description
nodes on SPT	Indicates the number of ISs that have been added.
Compute L2 SPT	Indicates that Level 2 ISs are to be added to the domain.
Complete L2 SPT	Indicates that the algorithm has completed for Level 2 routing.

The following lines show the statistical information available for Level 1 ISs:

```
ISIS-Stats: Compute L1 SPT, Timestamp 2780.328 seconds
ISIS-Stats: Complete L1 SPT, Compute time 0.004, 1 nodes on SPT
```

The output indicates that the SPF algorithm was applied 2780.328 seconds after the system was up and configured. Given the existing intra-area topology, 4 milliseconds were required to place one Level 1 IS on the SPT.

The following lines show the statistical information available for Level 2 ISs:

```
ISIS-Stats: Compute L2 SPT, Timestamp 2780.3336 seconds
ISIS-Stats: Complete L2 SPT, Compute time 0.056, 12 nodes on SPT
```

This output indicates that the SPF algorithm was applied 2780.3336 seconds after the system was up and configured. Given the existing intradomain topology, 56 milliseconds were required to place 12 Level 2 ISs on the SPT.

debug isis update-packets

To display various sequence number protocol data units (PDUs) and link-state packets that are detected by a router, use the **debug isis update-packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug isis update-packets

no debug isis update-packets

Syntax Description

This command has no arguments or keywords.

Examples

This router has been configured for IS-IS routing. The following is sample output from thee **debug isis update-packets** command:

```
Router# debug isis update-packets

ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Updating L2 LSP
ISIS-Update: Delete link 888.8800.0181.00 from L2 LSP 1600.8906.4022.00-00, seq E
ISIS-Update: Updating L1 LSP
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Add link 8888.8800.0181.00 to L2 LSP 1600.8906.4022.00-00, new seq 10,
 len 91
ISIS-Update: Sending L2 LSP 1600.8906.4022.00-00, seq 10, ht 1198 on Tunnel0
ISIS-Update: Sending L2 CSNP on Tunnel0
ISIS-Update: Updating L2 LSP
ISIS-Update: Rate limiting L2 LSP 1600.8906.4022.00-00, seq 11 (Tunnel0)
ISIS-Update: Updating L1 LSP
ISIS-Update: Rec L2 LSP 888.8800.0181.00.00-00 (Tunnel0)
ISIS-Update: PSNP entry 1600.8906.4022.00-00, seq 10, ht 1196
```

The following lines indicate that the router has sent a periodic Level 1 and Level 2 complete sequence number PDU on Ethernet interface 0:

```
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
```

The following lines indicate that the network service access point (NSAP) identified as 8888.8800.0181.00 was deleted from the Level 2 LSP 1600.8906.4022.00-00. The sequence number associated with this LSP is 0xE.

```
ISIS-Update: Updating L2 LSP
ISIS-Update: Delete link 888.8800.0181.00 from L2 LSP 1600.8906.4022.00-00, seq E
```

The following lines indicate that the NSAP identified as 8888.8800.0181.00 was added to the Level 2 LSP 1600.8906.4022.00-00. The new sequence number associated with this LSP is 0x10.

```
ISIS-Update: Updating L1 LSP
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Add link 8888.8800.0181.00 to L2 LSP 1600.8906.4022.00-00, new seq 10,
 len 91
```

The following line indicates that the router sent Level 2 LSP 1600.8906.4022.00-00 with sequence number 0x10 on tunnel 0 interface:

```
ISIS-Update: Sending L2 LSP 1600.8906.4022.00-00, seq 10, ht 1198 on Tunnel0
```

The following lines indicates that a Level 2 LSP could not be transmitted because it was recently sent:

```
ISIS-Update: Sending L2 CSNP on Tunnel0
ISIS-Update: Updating L2 LSP
ISIS-Update: Rate limiting L2 LSP 1600.8906.4022.00-00, seq 11 (Tunnel0)
```

The following lines indicate that a Level 2 partial sequence number PDU (PSNP) has been received on tunnel 0 interface:

```
ISIS-Update: Updating L1 LSP
ISIS-Update: Rec L2 PSNP from 8888.8800.0181.00 (Tunnel0)
```

The following line indicates that a Level 2 PSNP with an entry for Level 2 LSP 1600.8906.4022.00-00 has been received. This output is an acknowledgment that a previously sent LSP was received without an error.

```
ISIS-Update: PSNP entry 1600.8906.4022.00-00, seq 10, ht 1196
```

debug kerberos

To display information associated with the Kerberos Authentication Subsystem, use the **debug kerberos** privileged EXEC command. The **no** form of this command disables debugging output.

debug kerberos

no debug kerberos

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Kerberos is a security system that authenticates users and services without passing a cleartext password over the network. Cisco supports Kerberos under the authentication, authorization, and accounting (AAA) security system.

Use the **debug aaa authentication** command to get a high-level view of login activity. When Kerberos is used on the router, you can use the **debug kerberos** command for more detailed debugging information.

Examples

The following is part of the sample output from the **debug aaa authentication** command for a Kerberos login attempt that failed. The information indicates that Kerberos is the authentication method used.

```
Router# debug aaa authentication

AAA/AUTHEN/START (116852612): Method=KRB5
AAA/AUTHEN (116852612): status = GETUSER
AAA/AUTHEN/CONT (116852612): continue_login
AAA/AUTHEN (116852612): status = GETUSER
AAA/AUTHEN (116852612): Method=KRB5
AAA/AUTHEN (116852612): status = GETPASS
AAA/AUTHEN/CONT (116852612): continue_login
AAA/AUTHEN (116852612): status = GETPASS
AAA/AUTHEN (116852612): Method=KRB5
AAA/AUTHEN (116852612): password incorrect
AAA/AUTHEN (116852612): status = FAIL
```

The following is sample output from the **debug kerberos** command for a login attempt that was successful. The information indicates that the router sent a request to the KDC and received a valid credential.

```
Router# debug kerberos

Kerberos: Requesting TGT with expiration date of 820911631
Kerberos: Sent TGT request to KDC
Kerberos: Received TGT reply from KDC
Kerberos: Received valid credential with endtime of 820911631
```

The following is sample output from the **debug kerberos** command for a login attempt that failed. The information indicates that the router sent a request to the KDC and received a reply, but the reply did not contain a valid credential.

```
Router# debug kerberos

Kerberos: Requesting TGT with expiration date of 820911731
Kerberos: Sent TGT request to KDC
```

```
Kerberos: Received TGT reply from KDC
Kerberos: Received invalid credential.
AAA/AUTHEN (425003829): password incorrect
```

The following output shows other failure messages you might see that indicate a configuration problem. The first message indicates that the router failed to find the default Kerberos realm, therefore the process failed to build a message to send to the KDC. The second message indicates that the router failed to retrieve its own IP address. The third message indicates that the router failed to retrieve the current time. The fourth message indicates the router failed to find or create a credentials cache for a user, which is usually caused by low memory availability.

```
Router# debug kerberos

Kerberos: authentication failed when parsing name
Kerberos: authentication failed while getting my address
Kerberos: authentication failed while getting time of day
Kerberos: authentication failed while allocating credentials cache
```

Related Commands

Command	Description
debug aaa authentication	Displays information on accountable events as they occur.

debug l2relay events

To start debugging of Layer 2 Relay events, use the **debug l2relay events** command. To disable debugging output, use the **no** form of the command (SGSN D-node only).

debug l2relay events

no debug l2relay events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

The SGSN module uses the proprietary Layer 2 Relay protocol in conjunction with the intra-Serving GPRS Support Node (iSGSN) protocol for communication between the SGSN-datacom (SGSN-D) and SGSN-telecom (SGSN-T) units that comprise the SGSN.

For debugging purposes, it might also be useful to trace Layer 2 Relay packets. To display information about Layer 2 Relay packets, use the **debug l2relay packets** command.

Normally you will not need to use the **debug l2relay events** or **debug l2relay packets** commands. If problems with the SGSN are encountered, Cisco technical support personnel may request that issue the command.

Caution Because the **debug l2relay events** command generates a substantial amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of Layer 2 Relay events:

```
router# debug l2relay events
```

Related Commands

Command	Description
debug l2relay packets	Displays Layer 2 Relay packets (SGSN D-node only).

debug l2relay packets

To display information about Layer 2 Relay packets, use the **debug l2relay packets** command. To disable debugging output, use the **no** form of the command (SGSN D-node only).

debug l2relay packets

no debug l2relay packets

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

Use the **debug l2relay packets** command to display information about Layer 2 Relay packets.

The SGSN module uses the proprietary Layer 2 Relay protocol in conjunction with the intra-Serving GPRS Support Node (iSGSN) protocol for communication between the SGSN-datacom (SGSN-D) and SGSN-telecom (SGSN-T) units that comprise the SGSN.

For debugging purposes, it might also be useful to trace Layer 2 Relay events. To display information about Layer 2 Relay events, use the **debug l2relay events** command.

Normally you will not need to use the **debug l2relay packets** or **debug l2relay events** command. If problems with the SGSN are encountered, Cisco technical support personnel may request that you issue the command.

Caution Because the **debug l2relay packets** command generates a significant amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of Layer 2 Relay packets:

```
router# debug l2relay packets
```

Related Commands

Command	Description
debug ip igmp	Displays Layer 2 Relay events (SGSN D-node only).

debug lane client

To display information about a LAN Emulation Client (LEC), use the **debug lane client** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane client {**all** | **le-arp** | **mpoa** | **packet** | **signaling** | **state** | **topology**} [**interface** *interface*]

no debug lane client {**all** | **le-arp** | **mpoa** | **packet** | **signaling** | **state** | **topology**} [**interface** *interface*]

Syntax Description

all	Displays all debug information related to the LEC.
le-arp	Displays debug information related to the LANE ARP table.
mpoa	Displays debug information to track the following: • MPOA specific TLV information in le-arp requests/responses • Elan-id and local segment TLV in lane control frames • When a LANE client is bound to an MPC/MPS
packet	Displays debug information about each packet.
signaling	Displays debug information related to client SVCs.
state	Displays debug information when the state changes.
topology	Displays debug information related to the topology of the emulated LAN (ELAN).
interface *interface*	(Optional) Limits the debugging output to messages that relate to a particular interface or subinterface. If you enter this command multiple times with different interfaces, the last interface entered will be the one used to filter the messages.

Defaults

If the interface number is not specified, the default will be the number of all the **mpoa lane** clients.

Command History

Release	Modification
12.0(1)T	This command was introduced.

Usage Guidelines

The **debug lane client all** command can generate a large amount of output. Use a limiting keyword or specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

Sample Displays

The following example shows output for **debug lane client packet** and **debug lane client state** commands for an LEC joining an ELAN named elan1:

```
Router# debug lane client packet
```

```
Router# debug lane client state
```

The LEC listens for signalling calls to its ATM address (Initial State):

```
LEC ATM2/0.1: sending LISTEN
LEC ATM2/0.1:   listen on       39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1: received LISTEN
```

The LEC calls the LAN Emulation Configuration Server (LECS) and attempts to set up the Configure Direct VC (LECS Connect Phase):

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B43.00
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the LECS. The Configure Direct VC is established:

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   vcd             148
```

The LEC sends a CONFIG REQUEST to the LECS on the Configure Direct VC (Configuration Phase):

```
LEC ATM2/0.1: sending LANE_CONFIG_REQ on VCD 148
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC receives a CONFIG RESPONSE from the LECS on the Configure Direct VC:

```
LEC ATM2/0.1: received LANE_CONFIG_RSP on VCD 148
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC releases the Configure Direct VC:

```
LEC ATM2/0.1: sending RELEASE
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   cause code      31
```

The LEC receives a RELEASE_COMPLETE from the LECS:

```
LEC ATM2/0.1: received RELEASE_COMPLETE
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   cause code      16
```

The LEC calls the LAN Emulation Server (LES) and attempts to set up the Control Direct VC (Join/Registration Phase):

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:   callid          0x61167110
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B41.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the LES. The Control Direct VC is established:

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:   callid          0x61167110
LEC ATM2/0.1:   vcd             150
```

The LEC sends a JOIN REQUEST to the LES on the Control Direct VC:

```
LEC ATM2/0.1: sending LANE_JOIN_REQ on VCD 150
LEC ATM2/0.1:   Status          0
LEC ATM2/0.1:   LECID           0
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC receives a SETUP request from the LES to set up the Control Distribute VC:

```
LEC ATM2/0.1: received SETUP
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B41.01
```

The LEC responds to the LES call setup with a CONNECT:

```
LEC ATM2/0.1: sending CONNECT
LEC ATM2/0.1:   callid          0x6114D174
LEC ATM2/0.1:   vcd             151
```

A CONNECT_ACK is received from the ATM switch. The Control Distribute VC is established:

```
LEC ATM2/0.1: received CONNECT_ACK

The LEC receives a JOIN response from the LES on the Control Direct VC.

LEC ATM2/0.1: received LANE_JOIN_RSP on VCD 150
LEC ATM2/0.1:   Status          0
LEC ATM2/0.1:   LECID           1
LEC ATM2/0.1:   SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   LAN Type        2
LEC ATM2/0.1:   Frame size      2
LEC ATM2/0.1:   LAN Name        elan1
LEC ATM2/0.1:   LAN Name size   5
```

The LEC sends an LE ARP request to the LES to obtain the broadcast and unknown server (BUS) ATM NSAP address (BUS connect):

```
LEC ATM2/0.1: sending LANE_ARP_REQ on VCD 150
LEC ATM2/0.1:   SRC MAC address     0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address     39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   TARGET MAC address     ffff.ffff.ffff
LEC ATM2/0.1:   TARGET ATM address  00.000000000000000000000000.000000000000.00
```

The LEC receives its own LE ARP request via the LES over the Control Distribute VC:

```
LEC ATM2/0.1: received LANE_ARP_RSP on VCD 151
LEC ATM2/0.1:   SRC MAC address     0000.0ca0.5b40
LEC ATM2/0.1:   SRC ATM address     39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   TARGET MAC address     ffff.ffff.ffff
LEC ATM2/0.1:   TARGET ATM address  39.020304050607080910111213.00000CA05B42.01
```

The LEC calls the BUS and attempts to set up the Multicast Send VC:

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:   callid          0x6114D354
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B42.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the BUS. The Multicast Send VC is established:

```
LEC ATM2/0.1: received CONNECT
```

```
LEC ATM2/0.1:   callid          0x6114D354
LEC ATM2/0.1:   vcd             153
```

The LEC receives a SETUP request from the BUS to set up the Multicast Forward VC:

```
LEC ATM2/0.1: received SETUP
LEC ATM2/0.1:   callid          0x610D4230
LEC ATM2/0.1:   called party    39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:   calling_party   39.020304050607080910111213.00000CA05B42.01
```

The LEC responds to the BUS call setup with a CONNECT:

```
LEC ATM2/0.1: sending CONNECT
LEC ATM2/0.1:   callid          0x610D4230
LEC ATM2/0.1:   vcd             154
```

A CONNECT_ACK is received from the ATM switch. The Multicast Forward VC is established:

```
LEC ATM2/0.1: received CONNECT_ACK

The LEC moves into the OPERATIONAL state.

%LANE-5-UPDOWN: ATM2/0.1 elan elan1: LE Client changed state to up
```

The following output is from the **show lane client** command after the LEC joins the emulated LAN as shown in the **debug lane client** output:

```
Router# show lane client

LE Client ATM2/0.1  ELAN name: elan1  Admin: up  State: operational
Client ID: 1                 LEC up for 1 minute 2 seconds
Join Attempt: 1
HW Address: 0000.0ca0.5b40   Type: token ring           Max Frame Size: 4544
Ring:1      Bridge:1        ELAN Segment ID: 2048
ATM Address: 39.020304050607080910111213.00000CA05B40.01
 VCD  rxFrames  txFrames  Type       ATM Address
   0         0         0  configure  39.020304050607080910111213.00000CA05B43.00
 142         1         2  direct     39.020304050607080910111213.00000CA05B41.01
 143         1         0  distribute 39.020304050607080910111213.00000CA05B41.01
 145         0         0  send       39.020304050607080910111213.00000CA05B42.01
 146         1         0  forward    39.020304050607080910111213.00000CA05B42.01
```

The following example shows **debug lane client all** command output when an interface with LECS, an LES/BUS, and an LEC is shut down:

```
Router# debug lane client all

LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60E8B474
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_REL_COMP
LEC ATM1/0.2: action A_TEARDOWN_LEC
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2:   callid          0x60EB6160
LEC ATM1/0.2:   cause code      31
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2:   callid          0x60EB7548
LEC ATM1/0.2:   cause code      31
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2:   callid          0x60EB9E48
LEC ATM1/0.2:   cause code      31
LEC ATM1/0.2: sending CANCEL
LEC ATM1/0.2:   ATM address     47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
```

```
LEC ATM1/0.3:   callid          0x60E8D108
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_REL_COMP
LEC ATM1/0.3: action A_TEARDOWN_LEC
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3:   callid          0x60EB66D4
LEC ATM1/0.3:   cause code      31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3:   callid          0x60EB7B8C
LEC ATM1/0.3:   cause code      31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3:   callid          0x60EBA3BC
LEC ATM1/0.3:   cause code      31
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3:   ATM address     47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60EB7548
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:   callid          0x60EB7B8C
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBC458
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_REL_COMP
LEC ATM1/0.1: action A_TEARDOWN_LEC
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1:   callid          0x60EBD30C
LEC ATM1/0.1:   cause code      31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1:   callid          0x60EBDD28
LEC ATM1/0.1:   cause code      31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1:   callid          0x60EBF174
LEC ATM1/0.1:   cause code      31
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1:   ATM address     47.00918100000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBDD28
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60EB6160
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:   callid          0x60EB66D4
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:   callid          0x60EB9E48
LEC ATM1/0.2:   cause code      0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.3: received RELEASE_COMPLETE
```

```
LEC ATM1/0.3:   callid          0x60EBA3BC
LEC ATM1/0.3:   cause code      0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBD30C
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1:   callid          0x60EBF174
LEC ATM1/0.1:   cause code      0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.2: received CANCEL
LEC ATM1/0.2: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.3: received CANCEL
LEC ATM1/0.3: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.1: received CANCEL
LEC ATM1/0.1: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.1: action A_SHUTDOWN_LEC
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1:   ATM address     47.00918100000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
LEC ATM1/0.2: action A_SHUTDOWN_LEC
LEC ATM1/0.2: sending CANCEL
LEC ATM1/0.2:   ATM address     47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
LEC ATM1/0.3: action A_SHUTDOWN_LEC
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3:   ATM address     47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
```

The following output is from the **debug lane client mpoa** command when the **lane** interface is shut down:

```
Router# debug lane client mpoa

Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0.1
Router(config-subif)#shutdown
Router(config-subif)#
00:23:32:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:23:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:23:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#exit
Router(config)#exit
```

The following output is from the **debug lane client mpoa** command when the **lane** interface is started (not shut down):

```
Router# debug lane client mpoa

Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0.1
Router(config-subif)#
Router(config-subif)#
```

```
Router(config-subif)#no shutdown
Router(config-subif)#
00:23:39:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_CONFIG_RSP, num_tlvs 14
00:23:39:LEC ATM1/1/0.1:elan id from LECS set to 300
00:23:39:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_JOIN_RSP, num_tlvs 1
00:23:39:LEC ATM1/1/0.1:elan id from LES set to 300
00:23:39:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:23:39:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
00:23:39:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to up
00:23:39:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:UP
00:25:57:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_ARP_REQ, num_tlvs 1
00:25:57:LEC ATM1/1/0.1:lec_process_dev_type_tlv:    lec 47.0091810000000050E
2097801.00500B306440.02
    type MPS, mpc 00.000000000000000000000000.000000000000.00
    mps 47.0091810000000050E2097801.00500B306444.00, num_mps_mac 1, mac 0050.0b3
0.6440
00:25:57:LEC ATM1/1/0.1:create mpoa_lec
00:25:57:LEC ATM1/1/0.1:new mpoa_lec 0x617E3118
00:25:57:LEC ATM1/1/0.1:lec_process_dev_type_tlv:type MPS, num       _mps_mac
 1
00:2t 5:57:LEC ATM1/1/0.1:lec_add_mps:
    remote lec 47.0091810000000050E2097801.00500B306440.02
    mps 47.0091810000000050E2097801.00500B306444.00 num_mps_mac 1, mac 0050.0b30
.6440
00:25:57:LEC ATM1/1/0.1:mpoa_device_change:lec_nsap 47.0091810000000050E20978
01.00500B306440.02, appl_type 5
     mpoa_nsap 47.0091810000000050E2097801.00500B306444.00, opcode 4
00:25:57:LEC ATM1/1/0.1:lec_add_mps:add mac 0050.0b30.6440, mps_mac 0x617E372
C
00:25:57:LEC ATM1/1/0.1:mpoa_device_change:lec_nsap 47.0091810000000050E20978
01.00500B306440.02, appl_type 5
     mpoa_nsap 47.0091810000000050E2097801.00500B306444.00, opcode 5
00:25:57:LEC ATM1/1/0.1:     mps_mac 0050.0b30.6440
00:25:57:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:25:57:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
Router(config-subif)#exit
Router(config)#exit
```

The following output is from the **debug lane client mpoa** command when the ATM major interface is shut down:

```
Router# debug lane client mpoa

Router# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0
Router(config-if)# shutdown
Router(config-if)#
00:26:28:LANE ATM1/1/0:atm hardware reset
00:26:28:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:26:28:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:28:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:28:%MPOA-5-UPDOWN:MPC mpc2:state changed to down
00:26:28:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
00:26:30:%LINK-5-CHANGED:Interface ATM1/1/0, changed state to administratively
 down
00:26:30:LANE ATM1/1/0:atm hardware reset
00:26:31:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1/0, changed stat
e to down
Router(config-if)#
00:26:31:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
```

```
00:26:32:LANE ATM1/1/0:atm hardware reset
00:26:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:34:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
Router(config-if)# exit
Router(config)# exit
```

The following output is from the **debug lane client mpoa** command when the ATM major interface is started:

```
Router# debug lane client mpoa

Router# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# int atm 1/1/0
Router(config-if)# no shutdown
00:26:32:LANE ATM1/1/0:atm hardware reset
00:26:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:34:%LINK-3-UPDOWN:Interface ATM1/1/0, changed state to down
00:26:34:LANE ATM1/1/0:atm hardware reset
00:26:41:%LINK-3-UPDOWN:Interface ATM1/1/0, changed state to up
00:26:42:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1/0, changed stat
e to up
00:27:10:%LANE-6-INFO:ATM1/1/0:ILMI prefix add event received
00:27:10:LANE ATM1/1/0:prefix add event for 470091810000000050E2097801 ptr=0x6
17BFC0C len=13
00:27:10:    the current first prefix is now:470091810000000050E2097801
00:27:10:%ATMSSCOP-5-SSCOPINIT:- Intf :ATM1/1/0, Event :Rcv End, State :Act
ive.
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0

00:27:10:%LANE-3-NOREGILMI:ATM1/1/0.1 LEC cannot register 47.0091810000000050E
2097801.0050A29AF428.01 with ILMI
00:27:10:%LANE-6-INFO:ATM1/1/0:ILMI prefix add event received
00:27:10:LANE ATM1/1/0:prefix add event for 470091810000000050E2097801 ptr=0x6
17B8E6C len=13
00:27:10:    the current first prefix is now:470091810000000050E2097801
00:27:10:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:27:10:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0

00:27:10:%MPOA-5-UPDOWN:MPC mpc2:state changed to up
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 1

00:27:12:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_CONFIG_RSP, num_tlvs 14
00:27:12:LEC ATM1/1/0.1:elan id from LECS set to 300
00:27:12:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_JOIN_RSP, num_tlvs 1
00:27:12:LEC ATM1/1/0.1:elan id from LES set to 300
00:27:12:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:27:12:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
00:27:12:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to up
00:27:12:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:UP
Router(config-if)#exit
Router(config)#exit
```

Related Commands

Command	Description
debug modem traffic	Displays MPC debug information.
debug mpoa server	Displays information about the MPOA server.

debug lane config

To display information about a LANE configuration server, use the **debug lane config** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane config {**all** | **events** | **packets**}

no debug lane config {**all** | **events** | **packets**}

Syntax Description

all	Displays all debug messages related to the LANE configuration server. The output includes both the **events** and **packets** types of output.
events	Displays only messages related to significant LANE configuration server events.
packets	Displays information on each packet sent or received by the LANE configuration server.

Usage Guidelines

The **debug lane config** output is intended to be used primarily by a Cisco technical support representative.

Examples

The following is sample output from the **debug lane config all** command when an interface with LECS, an LES/BUS, and an LEC is shut down:

```
Router# debug lane config all

LECS EVENT ATM1/0: processing interface down transition
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: elan 'xxx' LES 47.00918100000000613E5A2F01.006070174821.01 callId
0x60CE0F58 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE0F58 cause 31
LECS EVENT ATM1/0: elan 'yyy' LES 47.00918100000000613E5A2F01.006070174821.02 callId
0x60CE2104 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2104 cause 31
LECS EVENT ATM1/0: elan 'zzz' LES 47.00918100000000613E5A2F01.006070174821.03 callId
0x60CE2DC8 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2DC8 cause 31
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE0F58 cause 0
LECS EVENT ATM1/0: call 0x60CE0F58 cleaned up
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2104 cause 0
LECS EVENT ATM1/0: call 0x60CE2104 cleaned up
```

```
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2DC8 cause 0
LECS EVENT ATM1/0: call 0x60CE2DC8 cleaned up
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT: config server 56 killed
```

debug lane finder

To display information about the finder internal state machine, use the **debug lane finder** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane finder

no debug lane finder

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug lane finder** command output is intended to be used primarily by a Cisco technical support representative.

Examples

The following is sample output from the **debug lane finder** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane finder

LECS FINDER ATM1/0.3: user request 1819 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: LECS 47.00918100000000613E5A2F01.006070174823.00 deleted
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1819
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
LECS FINDER ATM1/0.2: user request 1820 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1820
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
LECS FINDER ATM1/0.1: user request 1821 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER ATM1/0: finder state machine started
LECS FINDER ATM1/0: time to perform a getNext on the ILMI
LECS FINDER ATM1/0: ilmi_client_request failed, answering all users
LECS FINDER ATM1/0: answering all requests now
LECS FINDER ATM1/0: responded to user request 1821
LECS FINDER ATM1/0: number of remaining requests still to be processed: 0
```

debug lane server

To display information about a LANE server, use the **debug lane server** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane server [**interface** *interface*]

no debug lane server [**interface** *interface*]

Syntax Description

interface *interface*	(Optional) Limits the debugging output to messages relating to a specific interface or subinterface. If you use this command multiple times with different interfaces, the last interface entered is the one used to filter debug messages.

Usage Guidelines

The **debug lane server** command output is intended to be used primarily by a Cisco technical support representative. The **debug lane server** command can generate a substantial amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following is sample output from the **debug lane server** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane server

LES ATM1/0.1: lsv_lecsAccessSigCB called with callId 0x60CE124C, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.1: disconnected from the master LECS
LES ATM1/0.1: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: lsv_lecsAccessSigCB called with callId 0x60CE29E0, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.2: disconnected from the master LECS
LES ATM1/0.2: should have been connected, will reconnect in 3 seconds
LES ATM1/0.3: lsv_lecsAccessSigCB called with callId 0x60EB1940, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.3: disconnected from the master LECS
LES ATM1/0.3: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: elan yyy client 1 lost control distribute
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 state change Oper -> Term
LES ATM1/0.3: elan zzz client 1 lost control distribute
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 state change Oper -> Term
LES ATM1/0.2: elan yyy client 1 lost MC forward
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 lost MC forward
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 lost control distribute
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 state change Oper -> Term
LES ATM1/0.1: elan xxx client 1 lost MC forward
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 released control direct
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 released control direct
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 MC forward released
```

```
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1: freeing client structures
LES ATM1/0.2: elan yyy client 1 unregistered 0060.7017.4820
LES ATM1/0.2: elan yyy client 1 destroyed
LES ATM1/0.3: elan zzz client 1 MC forward released
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1: freeing client structures
LES ATM1/0.3: elan zzz client 1 unregistered 0060.7017.4820
LES ATM1/0.3: elan zzz client 1 destroyed
LES ATM1/0.1: elan xxx client 1 released control direct
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 MC forward released
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1: freeing client structures
LES ATM1/0.1: elan xxx client 1 unregistered 0060.7017.4820
LES ATM1/0.1: elan xxx client 1 destroyed
LES ATM1/0.1: elan xxx major interface state change
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: shutting down
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: elan xxx: LES/BUS state change operational -> terminating
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy major interface state change
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: shutting down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: elan yyy: LES/BUS state change operational -> terminating
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz major interface state change
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: shutting down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: elan zzz: LES/BUS state change operational -> terminating
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: stopped listening on addresses
LES ATM1/0.1: elan xxx: all clients killed
LES ATM1/0.1: elan xxx: multicast groups killed
LES ATM1/0.1: elan xxx: addresses de-registered from ilmi
LES ATM1/0.1: elan xxx: LES/BUS state change terminating -> down
LES ATM1/0.1: elan xxx: administratively down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: stopped listening on addresses
LES ATM1/0.2: elan yyy: all clients killed
LES ATM1/0.2: elan yyy: multicast groups killed
LES ATM1/0.2: elan yyy: addresses de-registered from ilmi
LES ATM1/0.2: elan yyy: LES/BUS state change terminating -> down
LES ATM1/0.2: elan yyy: administratively down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: stopped listening on addresses
LES ATM1/0.3: elan zzz: all clients killed
LES ATM1/0.3: elan zzz: multicast groups killed
LES ATM1/0.3: elan zzz: addresses de-registered from ilmi
LES ATM1/0.3: elan zzz: LES/BUS state change terminating -> down
LES ATM1/0.3: elan zzz: administratively down
```

```
LES ATM1/0.3: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.2: cleanupLecsAccess: discarding all validation requests
LES ATM1/0.1: cleanupLecsAccess: discarding all validation requests
```

debug lane signaling

To display information about LANE Server (LES) and BUS switched virtual circuits (SVCs), use the **debug lane signaling** privileged EXEC command. The **no** form of this command disables debugging output.

debug lane signaling [**interface** *interface*]

no debug lane signaling [**interface** *interface*]

Syntax Description

interface *interface*	(Optional) Limits the debugging output to messages relating to a specific interface or subinterface. If you use this command multiple times with different interfaces, the last interface entered is the one used to filter debug messages.

Usage Guidelines

The **debug lane signaling** command output is intended to be used primarily by a Cisco technical support representative. The **debug lane signaling** command can generate a substantial amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

Examples

The following is sample output from the **debug lane signaling** command when an interface with LECS, LES/BUS, and LEC is shut down:

```
Router# debug lane signaling

LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB565C cause 0 lv 0x60E8D348
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D348 from mcg 0x60E97E84
LANE SIG ATM1/0.2: timer for lv 0x60E8D348 stopped
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D468 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D3D8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D2B8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB5CA0 cause 0 lv 0x60E8BEF4
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8BEF4 from mcg 0x60E9A37C
LANE SIG ATM1/0.3: timer for lv 0x60E8BEF4 stopped
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8C014 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BF84 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BE64 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB9040 cause 0 lv 0x60E8D468
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D468 from mcg 0x60E97EC8
LANE SIG ATM1/0.2: timer for lv 0x60E8D468 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB97D4 cause 0 lv 0x60E8C014
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8C014 from mcg 0x60E9A3C0
LANE SIG ATM1/0.3: timer for lv 0x60E8C014 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBCEB8 cause 0 lv 0x60EBBAF0
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBAF0 from mcg 0x60E8F51C
LANE SIG ATM1/0.1: timer for lv 0x60EBBAF0 stopped
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBC10 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBB80 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBA60 in state LANE_VCC_CONNECTED
```

```
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBEB00 cause 0 lv 0x60EBBC10
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBC10 from mcg 0x60E8F560
LANE SIG ATM1/0.1: timer for lv 0x60EBBC10 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60E8B174 cause 0 lv 0x60E8D2B8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D2B8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60E8B990 cause 0 lv 0x60E8BE64
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BE64 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB7FE0 cause 0 lv 0x60E8D3D8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D3D8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB8554 cause 0 lv 0x60E8BF84
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BF84 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBB6D4 cause 0 lv 0x60EBBA60
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBA60 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBE24C cause 0 lv 0x60EBBB80
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBB80 stopped
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
```

debug lapb

To display all traffic for interfaces using Link Access Procedure, Balanced (LAPB) encapsulation, use the **debug lapb** privileged EXEC command. The **no** form of this command disables debugging output.

debug lapb

no debug lapb

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command displays information on the X.25 Layer 2 protocol. It is useful to users familiar with the LAPB protocol.

You can use the **debug lapb** command to determine why X.25 interfaces or LAPB connections are going up and down. It is also useful for identifying link problems, as evidenced when the **show interfaces** EXEC command displays a high number of rejects or frame errors over the X.25 link.

Caution Because the **debug lapb** command generates a substantial amount of output, use it when the aggregate of all LAPB traffic on X.25 and LAPB interfaces is fewer than five frames per second.

Examples

The following is sample output from the **debug lapb** command (the numbers 1 through 7 at the top of the display have been added in order to aid documentation):

```
1          2   3    4      5    6    7
Serial0: LAPB I CONNECT (5) IFRAME P 2 1
Serial0: LAPB O REJSENT (2) REJ F 3
Serial0: LAPB O REJSENT (5) IFRAME 0 3
Serial0: LAPB I REJSENT (2) REJ (C) 7
Serial0: LAPB I DISCONNECT (2) SABM P
Serial0: LAPB O CONNECT (2) UA F
Serial0: LAPB O CONNECT (5) IFRAME 0 0
Serial0: LAPB T1 CONNECT 357964 0
```

Each line of output describes a LAPB event. There are two types of LAPB events: frame events (when a frame enters or exits the LAPB) and timer events. In the sample output, the last line describes a timer event; all of the other lines describe frame events. Table 117 describes the first seven fields.

Table 117 debug lapb Field Descriptions

Field	Description
First field (1)	Interface type and unit number reporting the frame event.
Second field (2)	Protocol providing the information.
Third field (3)	Frame event type. Possible values are as follows: • I—Frame input • O—Frame output • T1—T1 timer expired • T3—Interface outage timer expired • T4—Idle link timer expired
Fourth field (4)	State of the protocol when the frame event occurred. Possible values are as follows: • BUSY (RNR frame received) • CONNECT • DISCONNECT • DISCSENT (disconnect sent) • ERROR (FRMR frame sent) • REJSENT (reject frame sent) • SABMSENT (SABM frame sent)
Fifth field (5)	In a frame event, this value is the size of the frame (in bytes). In a timer event, this value is the current timer value (in milliseconds).

Table 117 debug lapb Field Descriptions (continued)

Field	Description
Sixth field (6)	In a frame event, this value is the frame type name. Possible values for frame type names are as follows: • DISC—Disconnect • DM—Disconnect mode • FRMR—Frame reject • IFRAME—Information frame • ILLEGAL—Illegal LAPB frame • REJ—Reject • RNR—Receiver not ready • RR—Receiver ready • SABM—Set asynchronous balanced mode • SABME—Set asynchronous balanced mode, extended • UA—Unnumbered acknowledgment In a T1 timer event, this value is the number of retransmissions already attempted.
Seventh field (7) (This field will not print if the frame control field is required to appear as either a command or a response, and that frame type is correct.)	This field is only present in frame events. It describes the frame type identified by the LAPB address and Poll/Final bit. Possible values are as follows: • (C)—Command frame • (R)—Response frame • P—Command/Poll frame • F—Response/Final frame • /ERR—Command/Response type is invalid for the control field. An ?ERR generally means that the DTE/DCE assignments are not correct for this link. • BAD-ADDR—Address field is neither Command nor Response

A timer event only displays the first six fields of **debug lapb** command output. For frame events, however, the fields that follow the sixth field document the LAPB control information present in the frame. Depending on the value of the frame type name shown in the sixth field, these fields may or may not appear. Descriptions of the fields following the first six fields follow.

After the Poll/Final indicator, depending on the frame type, three different types of LAPB control information can be printed.

For information frames, the value of the N(S) field and the N(R) field will be printed. The N(S) field of an information frame is the sequence number of that frame, so this field will rotate between 0 and 7 for (modulo 8 operation) or 0 and 127 (for modulo 128 operation) for successive outgoing information frames and (under normal circumstances) also will rotate for incoming information frame streams. The N(R) field is a “piggybacked” acknowledgment for the incoming information frame stream; it informs the other end of the link which sequence number is expected next.

RR, RNR, and REJ frames have an N(R) field, so the value of that field is printed. This field has exactly the same significance that it does in an information frame.

For the FRMR frame, the error information is decoded to display the rejected control field, V(R) and V(S) values, the Response/Command flag, and the error flags WXYZ.

In the following example, the output shows an idle link timer action (T4) where the timer expires twice on an idle link, with the value of T4 set to five seconds:

```
Serial2: LAPB T4 CONNECT 255748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
Serial2: LAPB T4 CONNECT 260748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
```

The next example shows an interface outage timer expiration (T3):

```
Serial2: LAPB T3 DISCONNECT 273284
```

The following example output shows an error condition when no DCE to DTE connection exists. Note that if a frame has only one valid type (for example, a SABM can only be a command frame), a received frame that has the wrong frame type will be flagged as a receive error (R/ERR in the following output). This feature makes misconfigured links (DTE-DTE or DCE-DCE) easy to spot. Other, less common errors will be highlighed too, such as a too-short or too-long frame, or an invalid address (neither command nor response).

```
Serial2: LAPB T1 SABMSENT 1026508 1
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
Serial2: LAPB T1 SABMSENT 1029508 2
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
```

The output in the next example shows the router is misconfigured and has a standard (modulo 8) interface connected to an extended (modulo 128) interface. This condition is indicated by the SABM balanced mode and SABME balanced mode extended messages appearing on the same interface.

```
Serial2: LAPB T1 SABMSENT 1428720 0
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
Serial2: LAPB T1 SABMSENT 1431720 1
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
```

debug lapb-ta

To display debug messages for LAPB-TA, use the **debug lapb-ta** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug lapb-ta [**error** | **event** | **traffic**]

no debug lapb-ta [**error** | **event** | **traffic**]

Syntax Description

error	(Optional) Displays LAPB-TA errors.
event	(Optional) Displays LAPB-TA normal events.
traffic	(Optional) Displays LAPB-TA in/out traffic data.

Defaults

Debugging for LAPB-TA is not enabled.

Command History

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

Examples

The following is sample output from the **debug lapb-ta** command with the **error**, **event**, and **traffic** keywords activated:

```
Router# debug lapb-ta error

LAPB-TA error debugging is on
Router# debug lapb-ta event

LAPB-TA event debugging is on
Router# debug lapb-ta traffic

LAPB-TA traffic debugging is on

Mar  9 12:11:36.464:LAPB-TA:Autodetect trying to detect LAPB on
BR3/0:1
Mar  9 12:11:36.464:  sampled pkt: 2 bytes: 1 3F.. match
Mar  9 12:11:36.468:LAPBTA:get_ll_config:BRI3/0:1
Mar  9 12:11:36.468:LAPBTA:line 130 allocated for BR3/0:1
Mar  9 12:11:36.468:LAPBTA:process 79
Mar  9 12:11:36.468:BR3/0:1:LAPB-TA started
Mar  9 12:11:36.468:LAPBTA:service change:LAPB physical layer up,
context 6183E144 interface up, protocol down
Mar  9 12:11:36.468:LAPBTA:service change:, context 6183E144 up
Mar  9 12:11:36.468:LAPB-TA:BR3/0:1, 44 sent
2d14h:%LINEPROTO-5-UPDOWN:Line protocol on Interface BRI3/0:1, changed state to up
2d14h:%ISDN-6-CONNECT:Interface BRI3/0:1 is now connected to 60213
Mar  9 12:11:44.508:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.508:LAPB-TA:BR3/0:1, 3 sent
Mar  9 12:11:44.700:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.700:LAPB-TA:BR3/0:1, 3 sent
Mar  9 12:11:44.840:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:44.840:LAPB-TA:BR3/0:1, 14 sent
Mar  9 12:11:45.852:LAPB-TA:BR3/0:1, 1 rcvd
```

```
Mar  9 12:11:46.160:LAPB-TA:BR3/0:1, 2 rcvd
Mar  9 12:11:47.016:LAPB-TA:BR3/0:1, 1 rcvd
Mar  9 12:11:47.016:LAPB-TA:BR3/0:1, 10 sent
```

debug lat packet

To display information on all LAT events, use the **debug lat packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug lat packet

no debug lat packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For each datagram (packet) received or sent, a message is logged to the console.

Caution This command severely impacts LAT performance and is intended for troubleshooting use only.

Examples

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

```
Router# debug lat packet

LAT: I int=Ethernet0, src=0000.0c01.0509, dst=0900.2b00.000f, type=0, M=0, R=0
LAT: I int=Ethernet0, src=0800.2b11.2d13, dst=0000.0c01.7876, type=A, M=0, R=0
LAT: O dst=0800.2b11.2d13, int=Ethernet0, type= A, M=0, R=0, len= 20, next 0 ref 1
```

The second line of output describes a packet that is input to the router. Table 118 describes the fields in this line.

Table 118 debug lat packet Field Descriptions

Field	Description
LAT:	Indicates that this display shows LAT debugging output.
I	Indicates that this line of output describes a packet that is input to the router (I) or output from the router (O).
int = Ethernet0	Indicates the interface on which the packet event took place.
src = 0800.2b11.2d13	Indicates the source address of the packet.

Table 118 debug lat packet Field Descriptions (continued)

Field	Description
dst=0000.0c01.7876	Indicates the destination address of the packet.
type=A	Indicates the message type (in hexadecimal notation). Possible values are as follows: • 0 = Run Circuit • 1 = Start Circuit • 2 = Stop Circuit • A = Service Announcement • C = Command • D = Status • E = Solicit Information • F = Response Information

The third line of output describes a packet that is output from the router. Table 119 describes the last three fields in this line.

Table 119 debug lat packet Field Descriptions

Field	Description
len= 20	Indicates the length (in hexadecimal notation) of the packet (in bytes).
next 0	Indicates the link on the transmit queue.
ref 1	Indicates the count of packet users.

debug lex rcmd

To debug LAN Extender remote commands, use the **debug lex rcmd** privileged EXEC command. The **no** form of this command disables debugging output.

debug lex rcmd

no debug lex rcmd

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug lex rcmd** command:

```
Router# debug lex rcmd

LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
LEX-RCMD: REJ received
LEX-RCMD: illegal CODE field received in header: <number>
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
LEX-RCMD: Lex0 is not bound to a serial interface
LEX-RCMD: encapsulation failure
LEX-RCMD: timeout for Lex0: "lex priority-group" command
LEX-RCMD: re-transmitting Lex0: "lex priority-group" command
LEX-RCMD: lex_setup_and_send called with invalid parameter
LEX-RCMD: bind occurred on shutdown LEX interface
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
LEX-RCMD: No active Lex interface found for unbind
```

The following output indicates that a LAN Extender remote command packet was received on a serial interface that is not bound to a LAN Extender interface:

```
LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
```

This message can occur for any of the LAN Extender remote commands. Possible causes of this message are as follows:

- FLEX state machine software error
- Serial line momentarily goes down, which is detected by the host but not by FLEX

The following output indicates that a LAN Extender remote command response has been received. The hexadecimal values are for internal use only.

```
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
```

The following output indicates that when the host router originates a LAN Extender remote command to FLEX, it generates an 8-bit identifier that is used to associate a command with its corresponding response:

```
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
```

This message could be displayed for any of the following reasons:

- FLEX was busy at the time that the command arrived and could not send an immediate response. The command timed out on the host router and then FLEX finally sent the response.
- Transmission error.
- Software error.

Possible responses to Config-Request are Config-ACK, Config-NAK, and Config-Rej. The following output shows that some of the options in the Config-Request are not recognizable or are not acceptable to FLEX due to transmission errors or software errors:

```
LEX-RCMD: REJ received
```

The following output shows that a LAN Extender remote command response was received but that the CODE field in the header was incorrect:

```
LEX-RCMD: illegal CODE field received in header: <number>
```

The following output indicates that a LAN Extender remote command response was received but that it had an incorrect length field. This message can occur for any of the LAN Extender remote commands.

```
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
```

The following output shows that a host router was about to send a remote command when the serial link went down:

```
LEX-RCMD: Lex0 is not bound to a serial interface
```

The following output shows that the serial encapsulation routine of the interface failed to encapsulate the remote command datagram because the LEX-NCP was not in the OPEN state. Due to the way the PPP state machine is implemented, it is normal to see a single encapsulation failure for each remote command that gets sent at bind time.

```
LEX-RCMD: encapsulation failure
```

The following output shows that the timer expired for the given remote command without having received a response from the FLEX device. This message can occur for any of the LAN Extender remote commands.

```
 LEX-RCMD: timeout for Lex0: "lex priority-group" command
```

This message could be displayed for any of the following reasons:

- FLEX too busy to respond
- Transmission failure
- Software error

The following output indicates that the host is resending the remote command after a timeout:

```
LEX-RCMD: re-transmitting Lex0: "lex priority-group" command
```

The following output indicates that an illegal parameter was passed to the lex_setup_and_send routine. This message could be displayed for due to a host software error.

```
LEX-RCMD: lex_setup_and_send called with invalid parameter
```

The following output is informational and shows when a bind occurs on a shutdown interface:

```
LEX-RCMD: bind occurred on shutdown LEX interface
```

The following output shows that the LEX-NCP reached the open state and a bind operation was attempted with the FLEX's MAC address, but no free LAN Extender interfaces were found that were configured with that MAC address. This output can occur when the network administrator does not configure a LAN Extender interface with the correct MAC address.

```
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
```

The following output shows that the serial line that was bound to the LAN Extender interface went down and the unbind routine was called, but when the list of active LAN Extender interfaces was searched, the LAN Extender interface corresponding to the serial interface was not found. This output usually occurs because of a host software error.

```
LEX-RCMD: No active Lex interface found for unbind
```

debug list

To filter debugging information on a per-interface or per-access list basis, use the **debug list** privileged EXEC command. The **no** form of this command turns off the list filter.

debug list [*list*] [*interface*]

no debug list [*list*] [*interface*]

Syntax Description

list	(Optional) An access list number in the range from 1100 to 1199.
interface	(Optional) The nterface type. Allowed values are the following: • **channel**—IBM Channel interface • **ethernet**—IEEE 802.3 • **fddi**—ANSI X3T9.5 • **null**—Null interface • **serial**—Serial • **tokenring**—IEEE 802.5 • **tunnel**—Tunnel interface

Usage Guidelines

The **debug list** command is used with other **debug** commands for specific protocols and interfaces to filter the amount of debug information that is displayed. In particular, this command is designed to filter specific physical unit (PU) output from bridging protocols. The **debug list** command is supported with the following commands:

- **debug llc2 errors**
- **debug llc2 packets**
- **debug llc2 state**
- **debug rif**
- **debug sdlc**
- **debug token ring**

Note All **debug** commands that support access list filtering use access lists in the range from 1100 to 1199. The access list numbers shown in the examples are merely samples of valid numbers.

Examples

To use the **debug list** command on only the first of several LLC2 connections, use the **show llc2** command to display the active connections:

```
Router# show llc2

SdllcVirtualRing2008 DTE: 4000.2222.22c7 4000.1111.111c 04 04 state NORMAL
SdllcVirtualRing2008 DTE: 4000.2222.22c8 4000.1111.1120 04 04 state NORMAL
SdllcVirtualRing2008 DTE: 4000.2222.22c1 4000.1111.1104 04 04 state NORMAL
```

Next, configure an extended bridging access list, numbered 1103, for the connection you want to filter:

```
access-list 1103 permit 4000.1111.111c 0000.0000.0000 4000.2222.22c7 0000.0000.0000 0xC 2
eq 0x404
```

The convention for the LLC **debug list** command filtering is to use dmac = 6 bytes, smac = 6 bytes, dsap_offset = 12, and ssap_offset = 13.

Finally, you invoke the following **debug** commands:

```
Router# debug list 1103

Router# debug llc2 packet

LLC2 Packets debugging is on
for access list: 1103
```

To use the **debug list** command for SDLC connections, with the exception of address 04, create access list 1102 to deny the specific address and permit all others:

```
access-list 1102 deny 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000 0xC 1 eq
0x4
access-list 1102 permit 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000
```

The convention is to use dmac = 0.0.0, smac = 0.0.0, and sdlc_frame_offset = 12.

Invoke the following **debug** commands:

```
Router# debug list 1102

Router# debug sdlc

SDLC link debugging is on
for access list: 1102
```

To enable SDLC debugging (or debugging for any of the other supported protocols) for a specific interface rather than for all interfaces on a router, use the following commands:

```
Router# debug list serial 0

Router# debug sdlc

SDLC link debugging is on
for interface: Serial0
```

To enable Token Ring debugging between two MAC address, 0000.3018.4acd and 0000.30e0.8250, configure an extended bridging access list 1106:

```
access-list 1106 permit 0000.3018.4acd 8000.0000.0000 0000.30e0.8250 8000.0000.0000
access-list 1106 permit 0000.30e0.8250 8000.0000.0000 0000.3018.4acd 8000.0000.0000
```

Invoke the following **debug** commands:

```
Router# debug list 1106

Router# debug token ring
Token Ring Interface debugging is on
for access list: 1106
```

To enable RIF debugging for a single MAC address, configure an access list 1109:

```
access-list 1109 permit permit 0000.0000.0000 ffff.ffff.ffff 4000.2222.22c6 0000.0000.0000
```

Invoke the following debug commands:

```
Router# debug list 1109
Router# debug rif
RIF update debugging is on
```

```
for access list: 1109
```

Related Commands

Command	Description
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.
debug rif	Displays information on entries entering and leaving the RIF cache.
debug rtsp	Displays information on SDLC frames received and sent by any router serial interface involved in supporting SDLC end station functions.
debug token ring	Displays messages about Token Ring interface activity.

debug llc2 dynwind

To display changes to the dynamic window over Frame Relay, use the **debug llc2 dynwind** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 dynwind

no debug llc2 dynwind

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug llc2 dynwind** command:

```
Router# debug llc2 dynwind

LLC2/DW: BECN received! event REC_I_CMD, Window size reduced to 4
LLC2/DW: 1 consecutive I-frame(s) received without BECN
LLC2/DW: 2 consecutive I-frame(s) received without BECN
LLC2/DW: 3 consecutive I-frame(s) received without BECN
LLC2/DW: 4 consecutive I-frame(s) received without BECN
LLC2/DW: 5 consecutive I-frame(s) received without BECN
LLC2/DW: Current working window size is 5
```

In this example, the router receives a backward explicit congestion notification (BECN) and reduces the window size to four. After receiving five consecutive I frames without a BECN, the router increases the window size to five.

Related Commands

Command	Description
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 errors

To display Logical Link Control, type 2 (LLC2) protocol error conditions or unexpected input, use the **debug llc2 errors** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 errors

no debug llc2 errors

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug llc2 errors** command from a router ignoring an incorrectly configured device:

```
Router# debug llc2 errors

LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
```

Each line of output contains the remote MAC address, the local MAC address, the remote service access point (SAP), and the local SAP. In this example, the router receives unsolicited RR frames marked as responses.

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 packet

To display all input and output from the Logical Link Control, type 2 (LLC2) protocol stack, use the **debug llc2 packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 packet

no debug llc2 packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command also displays information about some error conditions as well as internal interactions between the Common Link Services (CLS) layer and the LLC2 layer.

Examples

The following is sample output from the **debug llc2 packet** command from the router sending ping data back and forth to another router:

```
Router# debug llc2 packet

LLC: llc2_input
401E54F0:                             10400000              .@..
401E5500: 303A90CF 0006F4E1 2A200404 012B5E    0:.O..ta* ...+
LLC: i REC_RR_CMD N(R)=21 p/f=1
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:              0040 0006F4E1 2A200000        .@..ta* ..
401E5620: 303A90CF 04050129 00               N  0:.O...).       2012
LLC: llc_sendframe
4022E3A0:                       0040 0006F4E1            .@..ta
4022E3B0: 2A200000 303A90CF 04042A28 2C000202  * ..0:.O..*(,...
4022E3C0: 00050B90 A02E0502 FF0003D1 004006C1  .... ......Q.@.A
4022E3D0: D7C9D5C    0.128
 C400130A C1D7D7D5 4BD5F2F0  WIUGD...AWWUKUrp
4022E3E0: F1F30000 011A6071 00010860 D7027000  qs....`q...`W.p.
4022E3F0: 00003B00 1112FF01 03000243 6973636F  ..;........Cisco
4022E400: 20494F53 69                            IOSi
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt I N(S)=21 N(R)=20 p/f=0 size=90
LLC: llc2_input
401E5620:                  10400000 303A90CF          .@..0:.O
401E5630: 0006F4E1 2A200404 282C2C00 02020004  ..ta* ..(,,.....
401E5640: 03902000 1112FF01 03000243 6973636F  .. ........Cisco
401E5650: 20494F53 A0                            IOS
LLC: i REC_I_CMD N(R)=22 N(S)=20 V(R)=20 p/f=0
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_I_CMD (1)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(S)=20 V(R)=20
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(R)=44
LLC: INFO: 0006.f4e1.2a20 0000.303a.90cf 04 04 v(r) 20
```

The first three lines indicate that the router has received some input from the link:

```
LLC: llc2_input
401E54F0:                             10400000              .@..
```

```
401E5500: 303A90CF 0006F4E1 2A200404 012B5E    0:.O..ta* ...+
```

The next line indicates that this input was an RR command with the poll bit set. The other router has received sequence number 21 and is waiting for the final bit.

```
LLC: i REC_RR_CMD N(R)=21 p/f=1
```

The next two lines contain the MAC addresses of the sender and receiver, and the state of the router when it received this frame:

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
```

The next four lines indicate that the router is sending a response with the final bit set:

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:              0040 0006F4E1 2A200000        .@..ta* ..
401E5620: 303A90CF 04050129 00               N  0:.O...).       2012
```

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 state	Displays state transitions of the LLC2 protocol.

debug llc2 state

To display state transitions of the Logical Link Control, type 2 (LLC2) protocol, use the **debug llc2 state** privileged EXEC command. The **no** form of this command disables debugging output.

debug llc2 state

no debug llc2 state

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Refer to the ISO/IEC standard 8802-2 for definitions and explanations of **debug llc2 state** command output.

Examples

The following is sample output from the **debug llc2 state** command when a router disables and enables an interface:

```
Router# debug llc2 state

LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> AWAIT (P_TIMER_EXP)
LLC(rs): 0006.f4e1.2a20 0000.303a.90cf 04 04, AWAIT -> D_CONN (P_TIMER_EXP)
LLC: cleanup 0006.f4e1.2a20 0000.303a.90cf 04 04, UNKNOWN (17)
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, ADM -> SETUP (CONN_REQ)
LLC: normalstate: set_local_busy 0006.f4e1.2a20 0000.303a.90cf 04 04
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> BUSY (SET_LOCAL_BUSY)
LLC: Connection established: 0006.f4e1.2a20 0000.303a.90cf 04 04, success
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, SETUP -> BUSY (SET_LOCAL_BUSY)
LLC: busystate: 0006.f4e1.2a20 0000.303a.90cf 04 04 local busy cleared
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, BUSY -> NORMAL (CLEAR_LOCAL_BUSY)
```

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.
debug llc2 dynwind	Displays changes to the dynamic window over Frame Relay.
debug llc2 errors	Displays LLC2 protocol error conditions or unexpected input.
debug llc2 packet	Displays all input and output from the LLC2 protocol stack.

debug lnm events

To display any unusual events that occur on a Token Ring network, use the **debug lnm events** privileged EXEC command. The **no** form of this command disables debugging output.

debug lnm events

no debug lnm events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Unusual events include stations reporting errors or error thresholds being exceeded.

Examples

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

```
Router# debug lnm events

IBMNM3: Adding 0000.3001.1166 to error list
IBMNM3: Station 0000.3001.1166 going into preweight condition
IBMNM3: Station 0000.3001.1166 going into weight condition
IBMNM3: Removing 0000.3001.1166 from error list
LANMGR0: Beaconing is present on the ring
LANMGR0: Ring is no longer beaconing
IBMNM3: Beaconing, Postmortem Started
IBMNM3: Beaconing, heard from 0000.3000.1234
IBMNM3: Beaconing, Postmortem Next Stage
IBMNM3: Beaconing, Postmortem Finished
```

The following message indicates that station 0000.3001.1166 reported errors and has been added to the list of stations reporting errors. This station is located on Ring 3.

```
IBMNM3: Adding 0000.3001.1166 to error list
```

The following message indicates that station 0000.3001.1166 has passed the “early warning” threshold for error counts:

```
IBMNM3: Station 0000.3001.1166 going into preweight condition
```

The following message indicates that station 0000.3001.1166 is experiencing a severe number of errors:

```
IBMNM3: Station 0000.3001.1166 going into weight condition
```

The following message indicates that the error counts for station 0000.3001.1166 have all decayed to zero, so this station is being removed from the list of stations that have reported errors:

```
IBMNM3: Removing 0000.3001.1166 from error list
```

The following message indicates that Ring 0 has entered failure mode. This ring number is assigned internally.

```
LANMGR0: Beaconing is present on the ring
```

The following message indicates that Ring 0 is no longer in failure mode. This ring number is assigned internally.

```
LANMGR0: Ring is no longer beaconing
```

The following message indicates that the router is beginning its attempt to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. The router attempts to contact stations that were part of the fault domain to detect whether they are still operating on the ring.

```
IBMNM3: Beaconing, Postmortem Started
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It received a response from station 0000.3000.1234, one of the two stations in the fault domain.

```
IBMNM3: Beaconing, heard from 0000.3000.1234
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It is initiating another attempt to contact the two stations in the fault domain.

```
IBMNM3: Beaconing, Postmortem Next Stage
```

The following message indicates that the router has attempted to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It has successfully heard back from both stations that were part of the fault domain.

```
IBMNM3: Beaconing, Postmortem Finished
```

Explanations follow for other messages that the **debug lnm events** command can generate.

The following message indicates that the router is out of memory:

```
LANMGR: memory request failed, find_or_build_station()
```

The following message indicates that Ring 3 is experiencing a large number of errors that cannot be attributed to any individual station:

```
IBMNM3: Non-isolating error threshold exceeded
```

The following message indicates that a station (or stations) on Ring 3 is receiving frames faster than they can be processed:

```
IBMNM3: Adapters experiencing congestion
```

The following message indicates that the beaconing has lasted for over 1 minute and is considered a “permanent” error:

```
IBMNM3: Beaconing, permanent
```

The following message indicates that the beaconing lasted for less than 1 minute. The router is attempting to determine whether either station in the fault domain left the ring.

```
IBMNM: Beaconing, Destination Started
```

In the preceding line of output, the following can replace “Started”: “Next State,” “Finished,” “Timed out,” and “Cannot find station *n*.”

debug lnm llc

To display all communication between the router/bridge and the LAN Network Managers (LNMs) that have connections to it, use the **debug lnm llc** privileged EXEC command. The **no** form of this command disables debugging output.

debug lnm llc

no debug lnm llc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

One line is displayed for each message sent or received.

Examples

The following is sample output from the **debug lnm llc** command:

```
Router# debug lnm llc

IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
IBMNM: Sending LRM LAN Manager Accepted to 1000.5ade.0d8a on link 0.
IBMNM: sending LRM New Reporting Link Established to 1000.5a79.dbf8 on link 1.
IBMNM: Determining new controlling LNM
IBMNM: Sending Report LAN Manager Control Shift to 1000.5ade.0d8a on link 0.
IBMNM: Sending Report LAN Manager Control Shift to 1000.5a79.dbf8 on link 1.

IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
IBMNM: Sending Report Bridge Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Request REM Status from 1000.5ade.0d8a.
IBMNM: Sending Report REM Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Set Bridge Parameters from 1000.5ade.0d8a.
IBMNM: Sending Bridge Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending Bridge Params Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-1-00A, addresses: 0000.3080.2d79 4000.3080.2d7
```

As the output indicates, the **debug lnm llc** command output can vary somewhat in format.

Table 120 describes the significant fields shown in the display.

Table 120 debug lnm llc Field Descriptions

Field	Description
IBMNM:	Displays LLC-level debugging information.
Received	Router received a frame. The other possible value is Sending, to indicate that the router is sending a frame.

Table 120 debug lnm llc Field Descriptions (continued)

Field	Description
LRM	The function of the LLC-level software that is communicating as follows: • CRS—Configuration Report Server • LBS—LAN Bridge Server • LRM—LAN Reporting Manager • REM—Ring Error Monitor • RPS—Ring Parameter Server • RS—Ring Station
Set Reporting Point	Name of the specific frame that the router sent or received. Possible values include the following: • Bridge Counter Report • Bridge Parameters Changed Notification • Bridge Parameters Set • CRS Remove Ring Station • CRS Report NAUN Change • CRS Report Station Information • CRS Request Station Information • CRS Ring Station Removed • LRM LAN Manager Accepted • LRM Set Reporting Point • New Reporting Link Established • REM Forward MAC Frame • REM Parameters Changed Notification • REM Parameters Set • Report Bridge Status • Report LAN Manager Control Shift • Report REM Status • Request Bridge Status • Request REM Status • Set Bridge Parameters • Set REM Parameters
from 1000.5ade.0d8a	If the router has received the frame, this address is the source address of the frame. If the router is sending the frame, this address is the destination address of the frame.

The following message indicates that the lookup for the bridge with which the LAN Manager was requesting to communicate was successful:

```
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
```

The following message indicates that the connection is being opened:

```
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
```

The following message indicates that a LAN Manager has connected or disconnected from an internal bridge and that the router computes which LAN Manager is allowed to change parameters:

```
IBMNM: Determining new controlling LNM
```

The following line of output indicates which bridge in the router is the destination for the frame:

```
IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
```

debug lnm mac

To display all management communication between the router/bridge and all stations on the local Token Rings, use the **debug lnm mac** privileged EXEC command. The **no** form of this command disables debugging output.

debug lnm mac

no debug lnm mac

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

One line is displayed for each message sent or received.

Examples

The following is sample output from the **debug lnm mac** command:

```
Router# debug lnm mac

LANMGR0: RS received request address from 4000.3040.a670.
LANMGR0: RS sending report address to 4000.3040.a670.
LANMGR0: RS received request state from 4000.3040.a670.
LANMGR0: RS sending report state to 4000.3040.a670.
LANMGR0: RS received request attachments from 4000.3040.a670.
LANMGR0: RS sending report attachments to 4000.3040.a670.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: CRS received report NAUN change from 0000.3040.a630.
LANMGR2: RS start watching ring poll.
LANMGR0: CRS received report NAUN change from 0000.3040.a630.
LANMGR0: RS start watching ring poll.
LANMGR2: REM received report soft error from 0000.3040.a630.
LANMGR0: REM received report soft error from 0000.3040.a630.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: RS received AMP from 0000.3040.a630.
LANMGR2: RS received SMP from 0000.3080.2d79.
LANMGR2: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR2: RS start watching ring poll.
LANMGR0: RS received ring purge from 0000.3040.a630.
LANMGR0: RS received AMP from 0000.3040.a630.
LANMGR0: RS received SMP from 0000.3080.2d79.
LANMGR0: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR0: RS start watching ring poll.
LANMGR2: RS received SMP from 1000.5ade.0d8a.
LANMGR2: RPS received request initialization from 1000.5ade.0d8a.
LANMGR2: RPS sending initialize station to 1000.5ade.0d8a.
```

Table 121 describes the significant fields shown in the display.

Table 121 debug lnm mac Field Descriptions

Field	Description
LANMGR0:	Indicates that this line of output displays MAC-level debugging information. 0 indicates the number of the Token Ring interface associated with this line of debugging output.
RS	Indicates which function of the MAC-level software is communicating as follows: • CRS—Configuration Report Server • REM—Ring Error Monitor • RPS—Ring Parameter Server • RS—Ring Station
received	Indicates that the router received a frame. The other possible value is sending, to indicate that the router is sending a frame.
request address	Indicates the name of the specific frame that the router sent or received. Possible values include the following: • AMP • initialize station • report address • report attachments • report nearest active upstream neighbor (NAUN) change • report soft error • report state • request address • request attachments • request initialization • request state • ring purge • SMP
from 4000.3040.a670	Indicates the source address of the frame, if the router has received the frame. If the router is sending the frame, this address is the destination address of the frame.

As the output indicates, all **debug lnm mac** command messages follow the format described in Table 121 except the following:

```
LANMGR2: RS start watching ring poll
LANMGR2: RS stop watching ring poll
```

These messages indicate that the router starts and stops receiving AMP and SMP frames. These frames are used to build a current picture of which stations are on the ring.

debug local-ack state

To display the new and the old state conditions whenever there is a state change in the local acknowledgment state machine, use the **debug local-ack state** privileged EXEC command. The **no** form of this command disables debugging output.

debug local-ack state

no debug local-ack state

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug local-ack state** command:

```
Router# debug local-ack state

LACK_STATE: 2370300, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2370304, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2373816, hashp 2AE628, old state = connected, new state = disconnected
LACK_STATE: 2489548, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2489548, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2490132, hashp 2AE628, old state = connected, new state = awaiting
linkdown response
LACK_STATE: 2490140, hashp 2AE628, old state = awaiting linkdown response,
new state = disconnected
LACK_STATE: 2497640, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2497644, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
```

Table 122 describes the significant fields in the display.

Table 122 debug local-ack state Field Descriptions

Field	Description
LACK_STATE:	Indicates that this packet describes a state change in the local acknowledgment state machine.
2370300	System clock.
hashp 2AE628	Internal control block pointer used by technical support staff for debugging purposes.

Table 122 debug local-ack state Field Descriptions (continued)

Field	Description
old state = disconn	Old state condition in the local acknowledgment state machine. Possible values include the following: • Disconn (disconnected) • awaiting LLC2 open to finish • connected • awaiting linkdown response
new state = awaiting LLC2 open to finish	New state condition in the local acknowledgment state machine. Possible values include the following: • Disconn (disconnected) • awaiting LLC2 open to finish • connected • awaiting linkdown response

debug management event

To monitor the activities of the Event MIB in real time on your routing device, use the **debug management event** command in privileged EXEC mode. To stop output of debug messages to your screen, use the **no** form of this command.

debug management event

no debug management event

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging output is disabled by default.

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

The **debug management event** command prints messages to the screen whenever the Event MIB evaluates a specified trigger. These messages are are given in real-time, and are intendended to be used by technical support engineers for troubleshooting purposes. Definitions for the OID (object identifier) fields can be found in the EVENT-MIB.my file, available for download from the Cisco MIB website on Cisco.com at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

Examples

The following example shows sample output for this command:

```
Router# debug management event

Event Process Bool: Owner aseem, Trigger 01
 Event Bool process: invoke event
 Event Bool process: no wildcarding
Event: OID ifEntry.10.3
Event getValue abs: 69847284
 Event Bool process: Trigger Fired !
 mteSetNotifyObjects:
 Event execOnFiring: sending notification
Event: OID ifEntry.10.1
Event add_objects: Owner , Trigger
Event add_objects: Owner aseem, Trigger sethi
Event Found Owner: aseem
Event Found Name: sethi
Event: OID ifEntry.10.1
 Event: sending trap with 7 OIDs
Event: OID mteHotTrigger.0
Event: OID mteHotTargetName.0
Event: OID mteHotContextName.0
```

```
Event: OID ifEntry.10.3
Event: OID mteHotValue.0
Event: OID ifEntry.10.1
Event: OID ifEntry.10.1
Event mteDoSets: setting oid
 Event mteDoSets: non-wildcarded oid
Event: OID ciscoSyslogMIB.1.2.1.0
Event Thresh Process: Owner aseem, Trigger 01
 Event Thresh process: invoke rising event
 Event Thresh process: invoke falling event
 Event Thresh process: no wildcarding
Event: OID ifEntry.10.3
Event getValue abs: 69847284
Event Existence Process: Owner aseem, Trigger 01
 Event Exist process: invoke event
 Event Exist process: no wildcarding
Event: OID ifEntry.10.3
Event getValue abs: 69847284
 Event Check ExistTrigger for Absent
 Event Check ExistTrigger for Changed
Router# no debug management event
```

Related Commands

Command	Description
show management event	Displays the SNMP Event values that have been configured on your routing device through the use of the Event MIB.

debug mdss

To display the run-time errors and sequence of events for the multicast distributed switching services (MDSS), use the **debug mdss** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug mdss {**all** | **error** | **event**}

no debug mdss {**all** | **error** | **event**}

Syntax Description

all	Displays both errors and sequence of events for MDSS.
error	Displays the run-time errors for MDSS.
event	Displays the run-time sequence of events for MDSS.

Defaults

Debugging is not enabled.

Command History

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

Examples

The following example shows output using the **debug mdss** command with the **all** keyword:

```
Router# debug mdss all

mdss all debugging is on
Router# clear ip mroute *
Router#
01:31:03: MDSS: got MDFS_CLEARALL
01:31:03: MDSS: --> mdss_flush_all_sc
01:31:03: MDSS: enqueue a FE_GLOBAL_DELETE
01:31:03: MDSS: got MDFS_MROUTE_ADD for (0.0.0.0, 224.0.1.40)
01:31:03: MDSS: --> mdss_free_scmdb_cache
01:31:03: MDSS: got MDFS_MROUTE_ADD for (0.0.0.0, 239.255.158.197)
01:31:03: MDSS: got MDFS_MROUTE_ADD for (192.1.21.6, 239.255.158.197)
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan22
01:31:03: MDSS: -- mdss_add_oif
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan22
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan23
01:31:03: MDSS: -- mdss_add_oif
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan
23
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan24
01:31:03: MDSS: -- mdss_add_oif
```

```
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan24
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan25
01:31:03: MDSS: -- mdss_add_oif
01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan25
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan26
01:31:03: MDSS: -- mdss_add_oif

01:31:03: MDSS: enqueue a FE_OIF_ADD (192.1.21.6, 239.255.158.197,
Vlan21) +Vlan26
01:31:03: MDSS: mdb (192.1.21.6, 239.255.158.197) fast_flags |
MCACHE_MTU
01:31:03: MDSS: got a MDFS_MIDB_ADD for (192.1.21.6, 239.255.158.197,u
Vlan21) +Vlan27
```

Related Commands

Command	Description
debug mls rp ip multicast	Displays information relating to MLSP.

debug mgcp

To enable debug traces for errors, events, packets, and the parser, use the **debug mgcp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mgcp [**all** | **errors** | **events** | **packets** | **parser**]

no debug mgcp [**all** | **errors** | **events** | **packets** | **parser**]

Syntax Description

all	(Optional) Debugs errors, events, packets, and the parser for MGCP modules. **Warning** **Using debug mgcp all may severly impact network performance**
errors	(Optional) Debugs errors for MGCP modules.
events	(Optional) Debugs events for MGCP modules.
packets	(Optional) Debugs packets for MGCP modules.
parser	(Optional) Debugs the parser for MGCP modules.

Defaults

Debugging for DRiP packets is not enabled.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced for the Cisco AS5300 access server.
12.1(3)T	The command was modified to display additional information for the gateways.

Examples

The following example illustrates the output for the **debug mgcp all** command with the **all** keyword:

```
Router# debug mgcp all

Router#
20:54:13: MGC stat - 192.168.10.10, total=37, succ=28, failed=8
20:54:13: MGCP Packet received -
CRCX 55560 s0/ds1-0/1 SGCP 1.1
C: 78980
M: sendrecv
L: a:G.726-16


20:54:13: -- mgcp_parse_packet() - call mgcp_parse_header
- mgcp_parse_header()- Request Verb FOUND CRCX
- mgcp_parse_packet() - out mgcp_parse_header
```

```
- SUCCESS: mgcp_parse_packet()-MGCP Header parsing was OK
- mgcp_parse_parameter_lines(), code_str:: 78980, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:C: 78980) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(78980), in_ptr(C: 78980)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[1]
- mgcp_parse_call_id(in_ptr: 78980)
- SUCCESS: mgcp_parse_call_id()-Call ID string(78980) parsing is OK
- mgcp_parse_parameter_lines(), code_str:: sendrecv, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:M: sendrecv) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(sendrecv), in_ptr(M: sendrecv)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[6]
- mgcp_parse_conn_mode(in_ptr: sendrecv)
- mgcp_parse_conn_mode()- tmp_ptr:(sendrecv)
- mgcp_parse_conn_mode(match sendrecv sendrecv
- mgcp_parse_conn_mode(case MODE_SENDRECV)
- SUCCESS: Connection Mode parsing is OK
- mgcp_parse_parameter_lines(), code_str:: a:G.726-16, code_len:2,
str:1640150312
- mgcp_parse_parameter_lines(str:L: a:G.726-16) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(a:G.726-16), in_ptr(L:
a:G.726-16)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[5]
- mgcp_parse_con_opts()
- mgcp_parse_codecs()
- SUCCESS: CODEC strings parsing is OK- SUCCESS: Local Connection option
parsing is OK- mgcp_val_mandatory_parms()

20:54:13: - SUCCESS: mgcp_parse_packet()- END of Parsing
20:54:13: MGCP msg 1

20:54:13: mgcp_search_call_by_endpt: endpt = s0/ds1-0/1, new_call = 1
20:54:13: slot=0,ds1=0,ds0=1

20:54:13: search endpoint - New call=1, callp 61C28130
20:54:13: callp: 61C28130, vdbptr: 0, state: 0
20:54:13: mgcp_remove_old_ack:
20:54:13: mgcp_idle_crcx: get capability
passthru is 3

20:54:13: process_request_ev- callp 61C28130, voice_if 61C281A4

20:54:13: process_detect_ev- callp 61C28130, voice_if 61C281A4
process_signal_ev- callp 61C28130, voice_ifp 61C281A4

20:54:13: mgcp_process_quarantine_mode- callp 61C28130, voice_if 61C281A4

20:54:13: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:54:13: mgcp_xlat_ccapi_error_code - ack_code_tab_index = 0,
20:54:13: No SDP connection info
20:54:13: mgcp_select_codec - LC option, num codec=1, 1st codec=5
20:54:13: mgcp_select_codec - num supprt codec=11
20:54:13: mgcp_select_codec - LC codec list only
20:54:13: codec index=0, bw=16000, codec=5
20:54:13: selected codec=5mgcp_get_pkt_period: voip_codec=2, pkt_period=0, call
adjust_packetization_period
mgcp_get_pkt_period: voip_codec=2, pkt_period=10, after calling
adjust_packetization_period

20:54:13: selected codec 5
20:54:13: IP Precedence=60
```

```
20:54:13: MGCP msg qos value=0mgcp_get_pkt_period: voip_codec=2, pkt_period=0,
call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=2, pkt_period=10, after calling
adjust_packetization_period
mgcp_new_codec_bytes: voip_codec=2, pkt_period=10, codec_bytes=20

20:54:13: callp : 61C28AE8, state : 2, call ID : 40, event : 5, minor evt:
1640137008

20:54:13: MGCPAPP state machine: state = 2, event = 5
20:54:13: mgcp_call_connect: call_id=40, ack will be sent later.
20:54:13: callp : 61C28AE8, new state : 3, call ID : 40

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: MGCP Session Appl: ignore CCAPI event 22, callp 61C28130

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: callp : 61C28130, state : 2, call ID : 39, event : 5, minor evt: 20

20:54:14: MGCPAPP state machine: state = 2, event = 5
20:54:14: callp : 61C28130, new state : 3, call ID : 39

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: callp : 61C28130, state : 3, call ID : 39, event : 6, minor evt: 20

20:54:14: MGCPAPP state machine: state = 3, event = 6
20:54:14: call_id=39, mgcp_ignore_ccapi_ev: ignore 6 for state 3

20:54:14: callp : 61C28130, new state : 3, call ID : 39

20:54:14: MGCP voice mode event

20:54:14: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:54:14: callp : 61C28130, state : 3, call ID : 39, event : 17, minor evt: 0

20:54:14: MGCPAPP state machine: state = 3, event = 17
20:54:14: mgcp_voice_mode_done(): callp 61C28130, major ev 17,
                    minor ev 0mgcp_start_ld_timer: timer already initialized

20:54:14: send_mgcp_create_ack
20:54:14: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:54:14: MGC stat - 1.13.89.3, total=37, succ=29, failed=8
20:54:14: Codec Cnt, 1, first codec 5
20:54:14: First Audio codec, 5, local encoding, 96
20:54:14: -- mgcp_build_packet()-

20:54:14: - mgcp_estimate_msg_buf_length() - 87 bytes needed for header
- mgcp_estimate_msg_buf_length() - 125 bytes needed after checking parameter
lines
- mgcp_estimate_msg_buf_length() - 505 bytes needed  after cheking SDP lines

20:54:14: --- mgcp_build_parameter_lines() ---
- mgcp_build_conn_id()
- SUCCESS: Conn ID string building is OK
- SUCCESS: Building MGCP Parameter lines is OK
- SUCCESS: building sdp owner id (o=) line
- SUCCESS: building sdp session name (s=) line
- SUCCESS: MGCP message building OK
- SUCCESS: END of building
updating lport with 2427

20:54:14: send_mgcp_msg, MGCP Packet sent --->
200 55560
```

```
I: 10

v=0
o=- 78980 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16444 RTP/AVP 96
a=rtpmap:96 G.726-16/8000/1

<---

20:54:14: enqueue_ack: voice_if=61C281A4, ackqhead=0, ackqtail=0,
ackp=61D753E8, msg=61D00010
20:54:14:
mgcp_process_quarantine_after_ack:ack_code=200mgcp_delete_qb_evt_q:cleanup QB
evt q

20:54:14: callp : 61C28130, new state : 4, call ID : 39
```

The following example illustrates the output for the **debug mgcp** command with the **events** keyword:

```
Router# debug mgcp events

Router#
20:51:40: MGC stat - 192.168.10.10, total=27, succ=20, failed=6
20:51:40: MGCP Packet received -
CRCX 55550 s0/ds1-0/1 SGCP 1.1
C: 100
M: sendonly
L: a:G.726-32, s:on


20:51:40: MGCP msg 1

20:51:40: mgcp_search_call_by_endpt: endpt = s0/ds1-0/1, new_call = 1
20:51:40: slot=0,ds1=0,ds0=1

20:51:40: search endpoint - New call=1, callp 61C28130
20:51:40: callp: 61C28130, vdbptr: 0, state: 0
20:51:40: mgcp_remove_old_ack:
20:51:40: mgcp_idle_crcx: get capability
passthru is 3

20:51:40: process_request_ev- callp 61C28130, voice_if 61C281A4

20:51:40: process_detect_ev- callp 61C28130, voice_if 61C281A4
process_signal_ev- callp 61C28130, voice_ifp 61C281A4

20:51:40: mgcp_process_quarantine_mode- callp 61C28130, voice_if 61C281A4

20:51:40: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:51:40: mgcp_xlat_ccapi_error_code - ack_code_tab_index = 0,
20:51:40: No SDP connection info
20:51:40: mgcp_select_codec - LC option, num codec=1, 1st codec=3
20:51:40: mgcp_select_codec - num supprt codec=11
20:51:40: mgcp_select_codec - LC codec list only
20:51:40: codec index=0, bw=32000, codec=3
20:51:40: selected codec=3mgcp_get_pkt_period: voip_codec=4, pkt_period=0, call
adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period
```

```
20:51:40: selected codec 3
20:51:40: IP Precedence=60
20:51:40: MGCP msg qos value=0mgcp_get_pkt_period: voip_codec=4, pkt_period=0,
call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period
mgcp_new_codec_bytes: voip_codec=4, pkt_period=10, codec_bytes=40

20:51:40: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:40: MGCP Session Appl: ignore CCAPI event 22, callp 61C28130

20:51:40: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:40: callp : 61C28130, state : 2, call ID : 31, event : 5, minor evt: 20

20:51:40: MGCPAPP state machine: state = 2, event = 5
20:51:40: mgcp_call_connect: call_id=31, ack will be sent later.
20:51:40: callp : 61C28130, new state : 3, call ID : 31

20:51:40: callp : 61C28AE8, state : 2, call ID : 32, event : 5, minor evt: 0

20:51:40: MGCPAPP state machine: state = 2, event = 5
20:51:40: callp : 61C28AE8, new state : 3, call ID : 32

20:51:40: callp : 61C28AE8, state : 3, call I 32, event : 6, minor evt: 0

20:51:40: MGCPAPP state machine: state = 3, event = 6
20:51:40: call_id=32, mgcp_ignore_ccapi_ev: ignore 6 for state 3

20:51:40: callp : 61C28AE8, new state : 3, call ID : 32

20:51:41: MGCP voice mode event

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61C28130, state : 3, call ID : 31, event : 17, minor evt: 0

20:51:41: MGCPAPP state machine: state = 3, event = 17
20:51:41: mgcp_voice_mode_done(): callp 61C28130, major ev 17,
                    minor ev 0mgcp_start_ld_timer: timer already initialized

20:51:41: send_mgcp_create_ack
20:51:41: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:51:41: MGC stat - 192.168.10.10, total=27, succ=21, failed=6
20:51:41: Codec Cnt, 1, first codec 3
20:51:41: First Audio codec, 3, local encoding, 96updating lport with 2427

20:51:41: send_mgcp_msg, MGCP Packet sent --->
200 55550
I: C

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16434 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:51:41: enqueue_ack: voice_if=61C281A4, ackqhead=0, ackqtail=0,
ackp=61D75384, msg=61C385EC
20:51:41:
```

```
mgcp_process_quarantine_after_ack:ack_code=200mgcp_delete_qb_evt_q:cleanup QB
evt q

20:51:41: callp : 61C28130, new state : 4, call ID : 31

20:51:41: MGC stat - 192.168.10.10, total=28, succ=21, failed=6
20:51:41: MGCP Packet received -
CRCX 55551 s0/ds1-0/2 SGCP 1.1
C: 100
M: sendrecv
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 191.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16434 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:51:41: MGCP msg 1

20:51:41: mgcp_search_call_by_endpt: endpt = s0/ds1-0/2, new_call = 1
20:51:41: slot=0,ds1=0,ds0=2

20:51:41: search endpoint - New call=1, callp 61F62380
20:51:41: callp: 61F62380, vdbptr: 0, state: 0
20:51:41: mgcp_remove_old_ack:
20:51:41: mgcp_idle_crcx: get capability
passthru is 3

20:51:41: process_request_ev- callp 61F62380, voice_if 61CDC9A8

20:51:41: process_detect_ev- callp 61F62380, voice_if 61CDC9A8
process_signal_ev- callp 61F62380, voice_ifp 61CDC9A8

20:51:41: mgcp_process_quarantine_mode- callp 61F62380, voice_if 61CDC9A8

20:51:41: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:51:41: mgcp_xlat_ccapi_error_code - ack_code_tab_index = 0,
20:51:41: get_peer_info, type 1, proto 1, port 16434
20:51:41: mgcp_select_codec - LC option, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - SDP list, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - num supprt codec=11
20:51:41: mgcp_select_codec - peer's pref codec is ok =3
20:51:41:  codec index=100000, bw=1000000, codec=0mgcp_get_pkt_period:
voip_codec=4, pkt_period=0, call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period

20:51:41: selected codec 3
20:51:41: IP Precedence=60
20:51:41: MGCP msg qos value=0mgcp_get_pkt_period: voip_codec=4, pkt_period=0,
call adjust_packetization_period
mgcp_get_pkt_period: voip_codec=4, pkt_period=10, after calling
adjust_packetization_period
mgcp_new_codec_bytes: voip_codec=4, pkt_period=10, codec_bytes=40

20:51:41: callp : 61D4CC1C, state : 2, call ID : 34, event : 5, minor evt:
1643520896

20:51:41: MGCPAPP state machine: state = 2, event = 5
```

```
20:51:41: mgcp_call_connect: call_id=34, ack will be sent later.
20:51:41: callp : 61D4CC1C, new state : 3, call ID : 34

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: MGCP Session Appl: ignore CCAPI event 22, callp 61F62380

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61F62380, state : 2, call ID : 33, event : 5, minor evt: 20

20:51:41: MGCPAPP state machine: state = 2, event = 5
20:51:41: callp : 61F62380, new state : 3, call ID : 33

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61F62380, state : 3, call ID : 33, event : 6, minor evt: 20

20:51:41: MGCPAPP state machine: state = 3, event = 6
20:51:41: call_id=33, mgcp_ignore_ccapi_ev: ignore 6 for state 3

20:51:41: callp : 61F62380, new state : 3, call ID : 33

20:51:41: MGCP voice mode event

20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61F62380, state : 3, call ID : 33, event : 17, minor evt: 0

20:51:41: MGCPAPP state machine: state = 3, event = 17
20:51:41: mgcp_voice_mode_done(): callp 61F62380, major ev 17,
                    minor ev 0mgcp_start_ld_timer: timer already initialized

20:51:41: send_mgcp_create_ack
20:51:41: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:51:41: MGC stat - 192.168.10.10, total=28, succ=22, failed=6
20:51:41: Codec Cnt, 1, first codec 3
20:51:41: First Audio codec, 3, local encoding, 96updating lport with 2427

20:51:41: send_mgcp_msg, MGCP Packet sent --->
200 55551
I: D

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16538 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:51:41: enqueue_ack: voice_if=61CDC9A8, ackqhead=0, ackqtail=0,
ackp=61D71C2C, msg=61CFF448
20:51:41:
mgcp_process_quarantine_after_ack:ack_code=200mgcp_delete_qb_evt_q:cleanup QB
evt q

20:51:41: callp : 61F62380, new state : 4, call ID : 33

20:51:41: MGC stat - 192.168.10.10, total=29, succ=22, failed=6
20:51:41: MGCP Packet received -
MDCX 55552 s0/ds1-0/1 SGCP 1.1
C: 100
I: C
M: sendrecv
```

```
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16538 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:51:41: MGCP msg 1

20:51:41: mgcp_search_call_by_endpt: endpt = s0/ds1-0/1, new_call = 0
20:51:41: slot=0,ds1=0,ds0=1

20:51:41: search endpoint - New call=0, callp 61C28130
20:51:41: callp: 61C28130, vdbptr: 61C290AC, state: 4
20:51:41: mgcp_remove_old_ack:mgcp_modify_connection: callp 61C28130

20:51:41: process_request_ev- callp 61C28130, voice_if 61C281A4

20:51:41: process_detect_ev- callp 61C28130, voice_if 61C281A4
process_signal_ev- callp 61C28130, voice_ifp 61C281A4

20:51:41: mgcp_process_quarantine_mode- callp 61C28130, voice_if 61C281A4

20:51:41: mgcp_process_quarantine_mode- new q mode: process=0, loop=0

20:51:41: mgcp_select_codec - LC option, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - SDP list, num codec=1, 1st codec=3
20:51:41: mgcp_select_codec - num supprt codec=11
20:51:41: mgcp_select_codec - peer's pref codec is ok =3
20:51:41:  codec index=100000, bw=1000000, codec=0
20:51:41: MGCP msg qos value=0
20:51:41: get_peer_info, type 1, proto 1, port 16538
20:51:41: mgcp_modify_connection: peer_addr=10D5902, peer_port=0->16538.
20:51:41: call modify - codec change callp 61C28130, callio 31, await_ev 1
20:51:41: mgcp_modify_connection: conn_mode=3.
20:51:41: mgcp_modify_conference: conf_id=11 callid1=31 callid2=32ccapi
conference already exists

20:51:41: mgcp_modify_connection - rtp change, callp 61C28AE8, callid 32,
await_ev 2
20:51:41: xlate_ccapi_ev - Protocol is SGCP, change pkg=2
20:51:41: callp : 61C28130, state : 4, call ID : 31, event : 16, minor evt:
1640137008

20:51:41: MGCPAPP state machine: state = 4, event = 16
20:51:41: mgcp_call_modified - callp 61C28130, voice_callp 61C28130 voice_if
61C281A4, await_ev 2

20:51:41: callp : 61C28130, new state : 4, call ID : 31

20:51:41: callp : 61C28AE8, state : 4, call ID : 32, event : 16, minor evt: 0

20:51:41: MGCPAPP state machine: state = 4, event = 16
20:51:41: mgcp_call_modified - callp 61C28AE8, voice_callp 61C28130 voice_if
61C281A4, await_ev 1

20:51:41: mgcp_call_modified - SUCCESS
20:51:41: map_mgcp_error_code_to_string error_tab_index  = 0, protocol version:
2
20:51:41: MGC stat - 1.13.89.3, total=29, succ=23, failed=6
```

```
20:51:41: send_mgcp_simple_ackupdating lport with 2427

20:51:41: send_mgcp_msg, MGCP Packet sent --->
200 55552 OK
```

The following example illustrates the output for the **debug mgcp** command with the **packet** keyword:

```
Router# debug mgcp pack
Media Gateway Control Protocol packets debugging is on
Router#
20:50:24: MGCP Packet received -
DLCX 55544 * SGCP 1.1


20:50:24: send_mgcp_msg, MGCP Packet sent --->
250 55544

<---

20:50:31: MGCP Packet received -
CRCX 55545 s0/ds1-0/1 SGCP 1.1
C: 100
M: sendonly
L: a:G.726-32, s:on


20:50:32: send_mgcp_msg, MGCP Packet sent --->
200 55545
I: A

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16468 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:50:32: MGCP Packet received -
CRCX 55546 s0/ds1-0/2 SGCP 1.1
C: 100
M: sendrecv
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16468 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:50:32: send_mgcp_msg, MGCP Packet sent --->
200 55546
I: B

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
```

```
t=0 0
m=audio 16386 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1

<---

20:50:32: MGCP Packet received -
MDCX 55547 s0/ds1-0/1 SGCP 1.1
C: 100
I: A
M: sendrecv
L: a:G.726-32, s:on

v=0
o=- 100 0 IN IP4 192.168.10.9
s=Cisco SDP 0
c=IN IP4 192.168.10.9
t=0 0
m=audio 16386 RTP/AVP 96
a=rtpmap:96 G.726-32/8000/1


20:50:33: send_mgcp_msg, MGCP Packet sent --->
200 55547 OK
```

The following example illustrates the output for the **debug mgcp** command with the **parser** keyword:

```
Router# debug mgcp parser

Router#
20:53:21: -- mgcp_parse_packet() - call mgcp_parse_header
- mgcp_parse_header()- Request Verb FOUND CRCX
- mgcp_parse_packet() - out mgcp_parse_header
- SUCCESS: mgcp_parse_packet()-MGCP Header parsing was OK
- mgcp_parse_parameter_lines(), code_str:: 78980, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:C: 78980) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(78980), in_ptr(C: 78980)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[1]
- mgcp_parse_call_id(in_ptr: 78980)
- SUCCESS: mgcp_parse_call_id()-Call ID string(78980) parsing is OK
- mgcp_parse_parameter_lines(), code_str:: sendrecv, code_len:2, str:1640150312
- mgcp_parse_parameter_lines(str:M: sendrecv) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(sendrecv), in_ptr(M: sendrecv)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[6]
- mgcp_parse_conn_mode(in_ptr: sendrecv)
- mgcp_parse_conn_mode()- tmp_ptr:(sendrecv)
- mgcp_parse_conn_mode(match sendrecv sendrecv
- mgcp_parse_conn_mode(case MODE_SENDRECV)
- SUCCESS: Connection Mode parsing is OK
- mgcp_parse_parameter_lines(), code_str:: a:G.726-16, code_len:2,
str:1640150312
- mgcp_parse_parameter_lines(str:L: a:G.726-16) -num_toks: 19
- mgcp_parse_parameter_lines() check NULL str(a:G.726-16), in_ptr(L:
a:G.726-16)
- mgcp_parse_parameter_lines() return Parse function in
mgcp_parm_rules_array[5]
- mgcp_parse_con_opts()
- mgcp_parse_codecs()
- SUCCESS: CODEC strings parsing is OK- SUCCESS: Local Connection option
parsing is OK- mgcp_val_mandatory_parms()
```

```
20:53:21: - SUCCESS: mgcp_parse_packet()- END of Parsing
20:53:22: -- mgcp_build_packet()-

20:53:22: - mgcp_estimate_msg_buf_length() - 87 bytes needed for header
- mgcp_estimate_msg_buf_length() - 125 bytes needed after checking parameter
lines
- mgcp_estimate_msg_buf_length() - 505 bytes needed  after cheking SDP lines

20:53:22: --- mgcp_build_parameter_lines() ---
- mgcp_build_conn_id()
- SUCCESS: Conn ID string building is OK
- SUCCESS: Building MGCP Parameter lines is OK
- SUCCESS: building sdp owner id (o=) line
- SUCCESS: building sdp session name (s=) line
- SUCCESS: MGCP message building OK
- SUCCESS: END of building
```

Related Commands

Command	Description
mgcp	Initiates the MGCP daemon.

debug mls rp

To display various IPX Multilayer Switching (MLS) debugging elements, use the **debug mls rp** privileged EXEC command. To disable debugging output, use the **no** form of the command.

debug mls rp {**error** | **events** | **ipx** | **locator** | **packets** | **all**}

no debug mls rp {**error** | **events** | **ipx** | **locator** | **packets** | **all**}

Syntax Description

error	Displays MLS error messages.
events	Displays a run-time sequence of events for the Multilayer Switching Protocol (MLSP).
ipx	Displays IPX-related events for MLS, including route purging and changes to access lists and flow masks.
locator	Identifies which switch is switching a particular flow of MLS explorer packets.
packets	Displays packet contents (in verbose and hexadecimal formats) for MLSP messages.
all	Displays all MLS debugging events.

Defaults

Debugging is not enabled.

Command History

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

Examples

The following example shows output using the **debug mls rp ipx** command:

```
Router# debug mls rp ipx

    IPX MLS debugging is on
Router# conf t
    Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# int vlan 22
Router(config-if)# no ipx access-group out
    05:44:37:FCP:flowmask changed to destination
```

Related Commands

Command	Description
debug dss ipx event	Displays debug messages for route change events that affect IPX MLS.

debug mls rp ip multicast

To display information about Multilayer Switching Protocol (MLSP), use the **debug mls rp ip multicast** privileged EXEC command. Use the **no** form of the command to disable debugging output.

debug mls rp ip multicast {**all** | **error** | **events** | **packets**}

no debug mls rp ip multicast {**all** | **error** | **events** | **packets**}

Syntax Description

all	Displays all multicast MLSP debugging information, including errors, events, and packets.
error	Displays error messages related to multicast MLSP.
events	Displays the run-time sequence of events for multicast MLSP.
packets	Displays the contents of MLSP packets.

Defaults

Debugging is not enabled.

Command History

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

Usage Guidelines

Only one of the keywords is required.

Examples

The following example shows output from the **debug mls rp ip multicast** command using the **error** keyword:

```
Router# debug mls rp ip multicast error

mlsm error debugging is on
chtang-7200#
06:06:45: MLSMERR: scb is INACTIVE, free INSTALL_FE
06:06:46: MLSM: --> mlsm_proc_sc_ins_req(10.0.0.1, 224.2.2.3, 10)
```

The following example shows output from the **debug mls rp ip multicast** command using the **event** keyword:

```
Router# debug mls rp ip multicast event

mlsm events debugging is on
Router#
3d23h: MSCP: incoming shortcut flow statistic  from Fa2/0.11
3d23h: MLSM: Flow_stat: (192.1.10.6, 239.255.158.197), byte :537792
packet:8403
3d23h: MLSM: byte delta:7680 packet delta:120, time delta: 10
3d23h: MSCP: incoming shortcut flow statistic  from Fa2/0.11
3d23h: MLSM: Flow_stat: (192.1.10.6, 239.255.158.197), byte :545472
packet:8523
3d23h: MLSM: byte delta:7680 packet delta:120, time delta: 10
3d23h: MSCP: Router transmits keepalive_msg on Fa2/0.11
```

```
3d23h: MSCP: incoming shortcut keepalive ACK  from Fa2/0.11
3d23h: MLSM: Include-list: (192.1.2.1  ->  0.0.0.0)
3d23h: MSCP: incoming shortcut flow statistic  from Fa2/0.11
3d23h: MLSM: Flow_stat: (192.1.10.6, 239.255.158.197), byte :553152
packet:8643
```

The following example shows output from the **debug mls rp ip multicast** command using the **packet** keyword:

```
Router# debug mls rp ip multicast packet

mlsm packets debugging is on
Router#
Router#
Router#
Router#
**23h: MSCP(I): 01 00 0c cc cc cc 00 e0 1e 7c fe 5f 00 30 aa aa
...LLL.`.|~_.0
..23h: MSCP(I): 03 00 00 0c 01 07 01 05 00 28 01 02 0a c7 00 10
.........(...G
 ..23h: MSCP(I): a6 0b b4 ff 00 00 c0 01 0a 06 ef ff 9e c5 00 00
&.4...@...o..E
3d23h: MSCP(I): 00 00 00 09 42 c0 00 00 00 00 00 00 25 0b
....B@......%.
3d23h:
**23h: MSCP(O): 01 00 0c 00 00 00 aa 00 04 00 01 04 00 00 aa aa
......*.......
LL23h: MSCP(O): 03 00 00 0c 00 16 00 00 00 00 01 00 0c cc cc cc
.............L
..23h: MSCP(O): aa 00 04 00 01 04 00 24 aa aa 03 00 00 0c 01 07
*......$**....
..23h: MSCP(O): 01 06 00 1c c0 01 02 01 aa 00 04 00 01 04 00 00
....@...*.....
3d23h: MSCP(O): 00 0b 00 00 00 00 00 00 01 01 0a 62               ...........b

3d23h:
**23h: MSCP(I): 01 00 0c cc cc cc 00 e0 1e 7c fe 5f 00 24 aa aa
...LLL.`.|~_.$
..23h: MSCP(I): 03 00 00 0c 01 07 01 86 00 1c 01 02 0a c7 00 10
.............G
..23h: MSCP(I): a6 0b b4 ff 00 00 00 0b 00 00 c0 01 02 01 00 00
..4.......@...
3d23h: MSCP(I): 00 00
3d23h:
```

Related Commands

Command	Description
debug mdss	Displays information about MDSS.

debug mmoip aaa

To display output relating to AAA services with the Store and Forward Fax feature, use the **debug mmoip aaa** EXEC command. Use the **no** form of this command to disable debugging output.

debug mmoip aaa

no debug mmoip aaa

Usage Guidelines

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

The following output shows how the **debug mmoip aaa** command provides information about AAA on-ramp or off-ramp authentication:

```
router# debug mmoip aaa

5d10h:fax_aaa_begin_authentication:User-Name = mmoip-b.cisco.com
5d10h:fax_aaa_begin_authentication:fax_account_id_origin = GATEWAY_ID
5d10h:fax_aaa_end_authentication_callback:Authentication successful
```

The following output shows how the **debug mmoip aaa** command provides information about AAA off-ramp accounting:

```
router# debug mmoip aaa

5d10h:fax_aaa_start_accounting:User-Name = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:Calling-Station-Id = gmercuri@mail-server.cisco.com
5d10h:fax_aaa_start_accounting:Called-Station-Id = fax=571-0839@mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:fax_account_id_origin = GATEWAY_ID
mmoip-b#ax_aaa_start_accounting:fax_msg_id = <37117AF3.3D98300E@mail-server.cisco.com>
5d10h:fax_aaa_start_accounting:fax_pages = 2
5d10h:fax_aaa_start_accounting:fax_coverpage_flag = TRUE
5d10h:fax_aaa_start_accounting:fax_modem_time = 26/32
5d10h:fax_aaa_start_accounting:fax_connect_speed = 14400bps
5d10h:fax_aaa_start_accounting:fax_recipient_count = 1
5d10h:fax_aaa_start_accounting:fax_auth_status = USER SUCCESS
5d10h:fax_aaa_start_accounting:gateway_id = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:call_type = Fax Send
5d10h:fax_aaa_start_accounting:port_used =  slot:0 modem port:0
5d10h:fax_aaa_do_offramp_accounting tty(6), Stopping accounting

5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveTransmitBytes = 18038
5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveTransmitPackets = 14
```

The following output shows how the **debug mmoip aaa** command provides information about AAA on-ramp accounting:

```
router# debug mmoip aaa

5d10h:fax_aaa_start_accounting:User-Name = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:Calling-Station-Id = FAX=408@mail-from-hostname.com
5d10h:fax_aaa_start_accounting:Called-Station-Id = FAX=5710839@mail-server.cisco.com
5d10h:fax_aaa_start_accounting:fax_account_id_origin = GATEWAY_ID
5d10h:fax_aaa_start_accounting:fax_msg_id = 00391997233216263@mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:fax_pages = 2
5d10h:fax_aaa_start_accounting:fax_modem_time = 22/32
5d10h:fax_aaa_start_accounting:fax_connect_speed = 14400bps
5d10h:fax_aaa_start_accounting:fax_auth_status = USER SUCCESS
5d10h:fax_aaa_start_accounting:email_server_address = 1.14.116.1
5d10h:fax_aaa_start_accounting:email_server_ack_flag = TRUE
5d10h:fax_aaa_start_accounting:gateway_id = mmoip-b.cisco.com
5d10h:fax_aaa_start_accounting:call_type = Fax Receive
5d10h:fax_aaa_start_accounting:port_used = Cisco Powered Fax System slot:1 port:4
5d10h:fax_aaa_do_onramp_accounting tty(5), Stopping accounting

5d10h:fax_aaa_stop_accounting:endb->cact->generic.callActiveTransmitBytes = 26687
5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveReceiveBytes = 18558
5d10h:fax_aaa_stop_accounting:ftdb->cact->generic.callActiveReceivePackets = 14
```

debug modem

To observe modem line activity on an access server, use the **debug modem** privileged EXEC command. The **no** form of this command disables debugging output.

debug modem

no debug modem

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug modem

15:25:51: TTY4: DSR came up
15:25:51: tty4: Modem: IDLE->READY
15:25:51: TTY4: Autoselect started
15:27:51: TTY4: Autoselect failed
15:27:51: TTY4: Line reset
15:27:51: TTY4: Modem: READY->HANGUP
15:27:52: TTY4: dropping DTR, hanging up
15:27:52: tty4: Modem: HANGUP->IDLE
15:27:57: TTY4: restoring DTR
15:27:58: TTY4: DSR came up
```

The output shows when the modem line changes state.

debug modem csm

To debug the Call Switching Module (CSM), used to connect calls on the modem, use the **debug modem csm** privileged EXEC command. The **no** form of this command disables debugging output.

debug modem csm [*slot/port* | **group** *group-number*]

no debug modem csm [*slot/port* | **group** *group-number*]

Syntax Description

slot/port	(Optional) The slot and modem port number.
group *group-number*	(Optional) The modem group.

Usage Guidelines

Use the **debug modem csm** command to troubleshoot call switching problems. With this command, you can trace the complete sequence of switching incoming and outgoing calls.

Examples

The following is sample output from the **debug modem csm** command. In this example, a call enters the modem (incoming) on slot 1, port 0:

```
Router(config)# service timestamps debug uptime

Router(config)# end

Router# debug modem csm

00:04:09: ccpri_ratetoteup bear rate is 10
00:04:09: CSM_MODEM_ALLOCATE: slot 1 and port 0 is allocated.
00:04:09: MODEM_REPORT(0001): DEV_INCALL at slot 1 and port 0
00:04:09: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 0
00:04:11: CSM_RING_INDICATION_PROC: RI is on
00:04:13: CSM_RING_INDICATION_PROC: RI is off
00:04:15: CSM_PROC_IC1_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 0
00:04:15: MODEM_REPORT(0001): DEV_CONNECTED at slot 1 and port 0
00:04:15: CSM_PROC_IC2_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 0
```

The following is sample output from the **debug modem csm** command when call is dialed from the modem into the network (outgoing) from slot 1, port 2:

```
Router# debug modem csm

atdt16665202
00:11:21: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 2
00:11:21: T1_MAIL_FROM_NEAT: DC_READY_RSP: mid = 1, slot = 0, unit = 0
00:11:21: CSM_PROC_OC1_REQUEST_DIGIT: CSM_EVENT_DIGIT_COLLECT_READY at slot 1, port 2
00:11:24: T1_MAIL_FROM_NEAT: DC_FIRST_DIGIT_RSP: mid = 1, slot = 0, unit = 0
00:11:24: CSM_PROC_OC2_COLLECT_1ST_DIGIT: CSM_EVENT_GET_1ST_DIGIT at slot 1, port 2
00:11:27: T1_MAIL_FROM_NEAT: DC_ALL_DIGIT_RSP: mid = 1, slot = 0, unit = 0
00:11:27: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS (16665202) at slot 1,
port 2
00:11:27: ccpri_ratetoteup bear rate is 10
00:11:27: MODEM_REPORT(A000): DEV_CALL_PROC at slot 1 and port 2
00:11:27: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port 2
00:11:31: MODEM_REPORT(A000): DEV_CONNECTED at slot 1 and port 2
00:11:31: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1, port 2
CONNECT 19200/REL - MNP
```

The following is sample output from the **debug modem csm** command for an incoming call:

```
Router# debug modem csm

Router#1.19.36.7 2001
Trying 1.19.36.7, 2001 ... Open
atdt111222333444555666
*Apr  7 12:39:42.475: Mica Modem(1/0): Rcvd Dial String(111222333444555666)
*Apr  7 12:39:42.475: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 0
*Apr  7 12:39:42.479: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_CHANNEL_LOCK at slot 1 and
port 0
*Apr  7 12:39:42.479: CSM_PROC_OC4_DIALING: CSM_EVENT_DSX0_BCHAN_ASSIGNED at slot 1, port
0
*Apr  7 12:39:42.479: Mica Modem(1/0): Configure(0x1)
*Apr  7 12:39:42.479: Mica Modem(1/0): Configure(0x5)
*Apr  7 12:39:42.479: Mica Modem(1/0): Call Setup
*Apr  7 12:39:42.479: neat msg at slot 0: (1/0): Tx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:42.491: neat msg at slot 0: (0/0): Rx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:42.531: VDEV_ALLOCATE: slot 1 and port 3 is allocated.
*Apr  7 12:39:42.531: CSM_RX_CAS_EVENT_FROM_NEAT:(0004):  EVENT_CALL_DIAL_IN at slot 1 and
port 3
*Apr  7 12:39:42.531: CSM_PROC_IDLE: CSM_EVENT_DSX0_CALL at slot 1, port 3
*Apr  7 12:39:42.531: Mica Modem(1/3): Configure(0x0)
*Apr  7 12:39:42.531: Mica Modem(1/3): Configure(0x5)
*Apr  7 12:39:42.531: Mica Modem(1/3): Call Setup
*Apr  7 12:39:42.595: Mica Modem(1/0): State Transition to Call Setup
*Apr  7 12:39:42.655: Mica Modem(1/3): State Transition to Call Setup
*Apr  7 12:39:42.655: Mica Modem(1/3): Went offhook
*Apr  7 12:39:42.655: CSM_PROC_IC1_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 3
*Apr  7 12:39:42.671: neat msg at slot 0: (0/0): Tx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:42.691: neat msg at slot 0: (1/0): Rx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:42.731: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_START_TX_TONE at slot 1
and port 0
*Apr  7 12:39:42.731: CSM_PROC_OC4_DIALING: CSM_EVENT_DSX0_START_TX_TONE at slot 1, port 0
*Apr  7 12:39:42.731: Mica Modem(1/0): Generate digits:called_party_num= len=1
*Apr  7 12:39:42.835: Mica Modem(1/3): Rcvd Digit detected(#)
*Apr  7 12:39:42.835: CSM_PROC_IC2_COLLECT_ADDR_INFO: CSM_EVENT_KP_DIGIT_COLLECTED (DNIS=,
ANI=) at slot 1, port 3
*Apr  7 12:39:42.855: neat msg at slot 0: (0/0): Tx LOOP_OPEN (ABCD=0101)
*Apr  7 12:39:42.871: neat msg at slot 0: (1/0): Rx LOOP_OPEN (ABCD=0101)
*Apr  7 12:39:42.899: Mica Modem(1/0): Rcvd Digits Generated
*Apr  7 12:39:42.911: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_END_TX_TONE at slot 1 and
port 0
*Apr  7 12:39:42.911: CSM_PROC_OC4_DIALING: CSM_EVENT_DSX0_END_TX_TONE at slot 1, port 0
*Apr  7 12:39:42.911: Mica Modem(1/0): Generate digits:called_party_num=A len=1
*Apr  7 12:39:43.019: Mica Modem(1/0): Rcvd Digits Generated
*Apr  7 12:39:43.019: CSM_PROC_OC4_DIALING: CSM_EVENT_TONE_GENERATED at slot 1, port 0
*Apr  7 12:39:43.019: Mica Modem(1/3): Rcvd Digit detected(A)
*Apr  7 12:39:43.335: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_START_TX_TONE at slot 1
and port 0
*Apr  7 12:39:43.335: CSM_PROC_OC4_DIALING: CSM_EVENT_DSX0_START_TX_TONE at slot 1, port 0
*Apr  7 12:39:43.335: Mica Modem(1/0): Generate digits:called_party_num=111222333444555666
len=19
*Apr  7 12:39:43.439: Mica Modem(1/3): Rcvd Digit detected(1)
*Apr  7 12:39:43.559: Mica Modem(1/3): Rcvd Digit detected(1)
*Apr  7 12:39:43.619: Mica Modem(1/3): Rcvd Digit detected(1)
*Apr  7 12:39:43.743: Mica Modem(1/3): Rcvd Digit detected(2)
*Apr  7 12:39:43.859: Mica Modem(1/3): Rcvd Digit detected(2)
*Apr  7 12:39:43.919: Mica Modem(1/3): Rcvd Digit detected(2)
*Apr  7 12:39:44.043: Mica Modem(1/3): Rcvd Digit detected(3)
*Apr  7 12:39:44.163: Mica Modem(1/3): Rcvd Digit detected(3)
*Apr  7 12:39:44.223: Mica Modem(1/3): Rcvd Digit detected(3)
*Apr  7 12:39:44.339: Mica Modem(1/3): Rcvd Digit detected(4)
*Apr  7 12:39:44.459: Mica Modem(1/3): Rcvd Digit detected(4)
```

```
*Apr  7 12:39:44.523: Mica Modem(1/3): Rcvd Digit detected(4)
*Apr  7 12:39:44.639: Mica Modem(1/3): Rcvd Digit detected(5)
*Apr  7 12:39:44.763: Mica Modem(1/3): Rcvd Digit detected(5)
*Apr  7 12:39:44.883: Mica Modem(1/3): Rcvd Digit detected(5)
*Apr  7 12:39:44.943: Mica Modem(1/3): Rcvd Digit detected(6)
*Apr  7 12:39:45.063: Mica Modem(1/3): Rcvd Digit detected(6)
*Apr  7 12:39:45.183: Mica Modem(1/3): Rcvd Digit detected(6)
*Apr  7 12:39:45.243: Mica Modem(1/3): Rcvd Digit detected(B)
*Apr  7 12:39:45.243: CSM_PROC_IC2_COLLECT_ADDR_INFO: CSM_EVENT_DNIS_COLLECTED
(DNIS=111222333444555666, ANI=) at slot 1, port 3
*Apr  7 12:39:45.363: Mica Modem(1/0): Rcvd Digits Generated
*Apr  7 12:39:45.891: neat msg at slot 0: (0/0): Tx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:45.907: neat msg at slot 0: (1/0): Rx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:46.115: neat msg at slot 0: (0/0): Tx LOOP_OPEN (ABCD=0101)
*Apr  7 12:39:46.131: neat msg at slot 0: (1/0): Rx LOOP_OPEN (ABCD=0101)
*Apr  7 12:39:46.175: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_START_TX_TONE at slot 1
and port 0
*Apr  7 12:39:46.175: CSM_PROC_OC4_DIALING: CSM_EVENT_DSX0_START_TX_TONE at slot 1, port 0
*Apr  7 12:39:46.175: Mica Modem(1/0): Generate digits:called_party_num= len=3
*Apr  7 12:39:46.267: Mica Modem(1/3): Rcvd Digit detected(#)
*Apr  7 12:39:46.387: Mica Modem(1/3): Rcvd Digit detected(A)
*Apr  7 12:39:46.447: Mica Modem(1/3): Rcvd Digit detected(B)
*Apr  7 12:39:46.447: CSM_PROC_IC2_COLLECT_ADDR_INFO: CSM_EVENT_ADDR_INFO_COLLECTED
(DNIS=111222333444555666, ANI=) at slot 1, port 3
*Apr  7 12:39:46.507: Mica Modem(1/0): Rcvd Digits Generated
*Apr  7 12:39:46.507: CSM_PROC_OC4_DIALING: CSM_EVENT_ADDR_INFO_COLLECTED at slot 1, port
0
*Apr  7 12:39:47.127: CSM_RX_CAS_EVENT_FROM_NEAT:(0004):  EVENT_CHANNEL_CONNECTED at slot
1 and port 3
*Apr  7 12:39:47.127: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_DSX0_CONNECTED at slot 1,
port 3
*Apr  7 12:39:47.127: Mica Modem(1/3): Link Initiate
*Apr  7 12:39:47.131: neat msg at slot 0: (0/0): Tx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:47.147: neat msg at slot 0: (1/0): Rx LOOP_CLOSURE (ABCD=1101)
*Apr  7 12:39:47.191: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_CHANNEL_CONNECTED at slot
1 and port 0
*Apr  7 12:39:47.191: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_DSX0_CONNECTED at slot 1,
port 0
*Apr  7 12:39:47.191: Mica Modem(1/0): Link Initiate
*Apr  7 12:39:47.227: Mica Modem(1/3): State Transition to Connect
*Apr  7 12:39:47.287: Mica Modem(1/0): State Transition to Connect
*Apr  7 12:39:49.103: Mica Modem(1/0): State Transition to Link
*Apr  7 12:39:52.103: Mica Modem(1/3): State Transition to Link
*Apr  7 12:40:00.927: Mica Modem(1/3): State Transition to Trainup
*Apr  7 12:40:00.991: Mica Modem(1/0): State Transition to Trainup
*Apr  7 12:40:02.615: Mica Modem(1/0): State Transition to EC Negotiating
*Apr  7 12:40:02.615: Mica Modem(1/3): State Transition to EC Negotiating
CONNECT 31200 /V.42/V.42bis
Router>
*Apr  7 12:40:05.983: Mica Modem(1/0): State Transition to Steady State
*Apr  7 12:40:05.983: Mica Modem(1/3): State Transition to Steady State+++
OK
ath
*Apr  7 12:40:09.167: Mica Modem(1/0): State Transition to Steady State Escape
*Apr  7 12:40:10.795: Mica Modem(1/0): State Transition to Terminating
*Apr  7 12:40:10.795: Mica Modem(1/3): State Transition to Terminating
*Apr  7 12:40:11.755: Mica Modem(1/3): State Transition to Idle
*Apr  7 12:40:11.755: Mica Modem(1/3): Went onhook
*Apr  7 12:40:11.755: CSM_PROC_IC5_OC6_CONNECTED: CSM_EVENT_MODEM_ONHOOK at slot 1, port 3
*Apr  7 12:40:11.755: VDEV_DEALLOCATE: slot 1 and port 3 is deallocated
*Apr  7 12:40:11.759: neat msg at slot 0: (0/0): Tx LOOP_OPEN (ABCD=0101)
*Apr 7 12:40:11.767: neat msg at slot 0: (1/0): Rx LOOP_OPEN (ABCD=0101)
*Apr  7 12:40:12.087: neat msg at slot 0: (1/0): Tx LOOP_OPEN (ABCD=0101)
*Apr  7 12:40:12.091: neat msg at slot 0: (0/0): Rx LOOP_OPEN (ABCD=0101)
```

```
*Apr  7 12:40:12.111: CSM_RX_CAS_EVENT_FROM_NEAT:(A001):  EVENT_CALL_IDLE at slot 1 and
port 0
*Apr  7 12:40:12.111: CSM_PROC_IC5_OC6_CONNECTED: CSM_EVENT_DSX0_DISCONNECTED at slot 1,
port 0
*Apr  7 12:40:12.111: Mica Modem(1/0): Link Terminate(0x6)
*Apr  7 12:40:12.779: Mica Modem(1/3): State Transition to Terminating
*Apr  7 12:40:12.839: Mica Modem(1/3): State Transition to Idle
*Apr  7 12:40:13.495: Mica Modem(1/0): State Transition to Idle
*Apr  7 12:40:13.495: Mica Modem(1/0): Went onhook
*Apr  7 12:40:13.495: CSM_PROC_IC6_OC8_DISCONNECTING: CSM_EVENT_MODEM_ONHOOK at slot 1,
port 0
*Apr  7 12:40:13.495: VDEV_DEALLOCATE: slot 1 and port 0 is deallocated
Router#disc
Closing connection to 1.19.36.7 [confirm]
Router#
*Apr  7 12:40:18.783: Mica Modem(1/0): State Transition to Terminating
*Apr  7 12:40:18.843: Mica Modem(1/0): State Transition to Idle
Router#
```

The MICA technologies modem goes through the following internal link states when the call comes in:

- Call Setup
- Off Hook
- Connect
- Link
- Trainup
- EC Negotiation
- Steady State

The following section describes the CSM activity for an incoming call.

When a voice call comes in, CSM is informed of the incoming call. This allocates the modem and sends the Call Setup message to the MICA modem. The Call_Proc message is sent through D channel. The modem sends an offhook message to CSM by sending the state change to Call Setup. The D channel then sends a CONNECT message. When the CONNECT_ACK message is received, the Link initiate message is sent to the MICA modem and it negotiates the connection with the remote modem. In the following debug examples, a modem on slot 1, port 13 is allocated. It goes through its internal states before it is in Steady State and answers the call.

```
Router# debug modem csm

Modem Management Call Switching Module debugging is on
*May 13 15:01:00.609: MODEM_REPORT:dchan_idb=0x60D437F8, call_id=0xE, ces=0x1
   bchan=0x12, event=0x1, cause=0x0
*May 13 15:01:00.609: VDEV_ALLOCATE: slot 1 and port 13 is allocated.
*May 13 15:01:00.609: MODEM_REPORT(000E): DEV_INCALL at slot 1 and port 13
*May 13 15:01:00.609: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 13
*May 13 15:01:00.609: Mica Modem(1/13): Configure(0x0)
*May 13 15:01:00.609: Mica Modem(1/13): Configure(0x0)
*May 13 15:01:00.609: Mica Modem(1/13): Configure(0x6)
*May 13 15:01:00.609: Mica Modem(1/13): Call Setup
*May 13 15:01:00.661: Mica Modem(1/13): State Transition to Call Setup
*May 13 15:01:00.661: Mica Modem(1/13): Went offhook
*May 13 15:01:00.661: CSM_PROC_IC1_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 13
*May 13 15:01:00.661: MODEM_REPORT:dchan_idb=0x60D437F8, call_id=0xE, ces=0x1
   bchan=0x12, event=0x4, cause=0x0
*May 13 15:01:00.661: MODEM_REPORT(000E): DEV_CONNECTED at slot 1 and port 13
*May 13 15:01:00.665: CSM_PROC_IC3_WAIT_FOR_CARRIER:
CSM_EVENT_ISDN_CONNECTED at slot 1, port 13
```

```
*May 13 15:01:00.665: Mica Modem(1/13): Link Initiate
*May 13 15:01:00.693: Mica Modem(1/13): State Transition to Connect
*May 13 15:01:01.109: Mica Modem(1/13): State Transition to Link
*May 13 15:01:09.433: Mica Modem(1/13): State Transition to Trainup
*May 13 15:01:11.541: Mica Modem(1/13): State Transition to EC Negotiating
*May 13 15:01:12.501: Mica Modem(1/13): State Transition to Steady State
```

The following section describes the status of CSM when a call is connected.

The **show modem csm x/y** command is similar to AS5200 access server. For an active incoming analog call, the modem_status and csm_status should be VDEV_STATUS_ACTIVE_CALL and CSM_IC4_CONNECTED, respectively.

```
Router# show modem csm 1/13

MODEM_INFO: slot 1, port 13, unit 0, modem_mask=0x0000, modem_port_offset=0
tty_hwidb=0x60D0BCE0, modem_tty=0x60B6FE7C, oobp_info=0x00000000,
modem_pool=0x60ADC998
modem_status(0x0002):VDEV_STATUS_ACTIVE_CALL.
csm_state(0x0204)=CSM_IC4_CONNECTED, csm_event_proc=0x600C6968, current
call thru PRI line
invalid_event_count=0, wdt_timeout_count=0
wdt_timestamp_started is not activated
wait_for_dialing:False, wait_for_bchan:False
pri_chnl=TDM_PRI_STREAM(s0, u0, c18), modem_chnl=TDM_MODEM_STREAM(s1, c13)
dchan_idb_start_index=0, dchan_idb_index=0, call_id=0x000E, bchan_num=18
csm_event=CSM_EVENT_ISDN_CONNECTED, cause=0x0000
ring_indicator=0, oh_state=0, oh_int_enable=0, modem_reset_reg=0
ring_no_answer=0, ic_failure=0, ic_complete=1
dial_failure=0, oc_failure=0, oc_complete=0
oc_busy=0, oc_no_dial_tone=0, oc_dial_timeout=0
remote_link_disc=0, stat_busyout=0, stat_modem_reset=0
oobp_failure=0
call_duration_started=1d02h, call_duration_ended=00:00:00,
total_call_duration=00:00:00
The calling party phone number = 4085552400
The called party phone number  = 4085551400
total_free_rbs_timeslot = 0, total_busy_rbs_timeslot = 0,
total_dynamic_busy_rbs_timeslot = 0, total_static_busy_rbs_timeslot = 0,
min_free_modem_threshold = 6
```

The following section describes the CSM activity for an outgoing call.

For MICA modems, the dial tone is not required to initiate an outbound call. Unlike in the AS5200, the digit collection step is not required. The dialed digit string is sent to the CSM in the outgoing request to the CSM. CSM signals the D channel to generate an outbound voice call, and the B channel assigned is connected to the modem and the CSM.

The modem is ordered to connect to the remote side with a CONNECT message, and by sending a link initiate message, the modem starts to train.

```
Router# debug modem csm

Modem Management Call Switching Module debugging is on
Router# debug isdn q931
ISDN Q931 packets debugging is on
*May 15 12:48:42.377: Mica Modem(1/0): Rcvd Dial String(5552400)
*May 15 12:48:42.377: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 0
*May 15 12:48:42.377: CSM_PROC_OC3_COLLECT_ALL_DIGIT:
CSM_EVENT_GET_ALL_DIGITS at slot 1, port 0
*May 15 12:48:42.377: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num:
(5552400) at slot 1, port 0
*May 15 12:48:42.381:  process_pri_call making a voice_call.
*May 15 12:48:42.381: ISDN Se0:23: TX ->  SETUP pd = 8  callref = 0x0011
```

```
*May 15 12:48:42.381:         Bearer Capability i = 0x8090A2
*May 15 12:48:42.381:         Channel ID i = 0xE1808397
*May 15 12:48:42.381:         Called Party Number i = 0xA1, '5552400'
*May 15 12:48:42.429: ISDN Se0:23: RX <-  CALL_PROC pd = 8  callref = 0x8011
*May 15 12:48:42.429:         Channel ID i = 0xA98397
*May 15 12:48:42.429: MODEM_REPORT:dchan_idb=0x60D437F8, call_id=0xA011, ces=0x1
   bchan=0x16, event=0x3, cause=0x0
*May 15 12:48:42.429: MODEM_REPORT(A011): DEV_CALL_PROC at slot 1 and port 0
*May 15 12:48:42.429: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED
at slot 1, port 0
*May 15 12:48:42.429: Mica Modem(1/0): Configure(0x1)
*May 15 12:48:42.429: Mica Modem(1/0): Configure(0x0)
*May 15 12:48:42.429: Mica Modem(1/0): Configure(0x6)
*May 15 12:48:42.429: Mica Modem(1/0): Call Setup
*May 15 12:48:42.489: Mica Modem(1/0): State Transition to Call Setup
*May 15 12:48:42.589: ISDN Se0:23: RX <-  ALERTING pd = 8  callref = 0x8011
*May 15 12:48:43.337: ISDN Se0:23: RX <-  CONNECT pd = 8  callref = 0x8011
*May 15 12:48:43.341: MODEM_REPORT:dchan_idb=0x60D437F8, call_id=0xA011, ces=0x1
   bchan=0x16, event=0x4, cause=0x0
*May 15 12:48:43.341: MODEM_REPORT(A011): DEV_CONNECTED at slot 1 and port 0
*May 15 12:48:43.341: CSM_PROC_OC5_WAIT_FOR_CARRIER:
CSM_EVENT_ISDN_CONNECTED at slot 1, port 0
*May 15 12:48:43.341: Mica Modem(1/0): Link Initiate
*May 15 12:48:43.341: ISDN Se0:23: TX ->  CONNECT_ACK pd = 8  callref = 0x0011
*May 15 12:48:43.385: Mica Modem(1/0): State Transition to Connect
*May 15 12:48:43.849: Mica Modem(1/0): State Transition to Link
*May 15 12:48:52.665: Mica Modem(1/0): State Transition to Trainup
*May 15 12:48:54.661: Mica Modem(1/0): State Transition to EC Negotiating
*May 15 12:48:54.917: Mica Modem(1/0): State Transition to Steady State
```

Related Commands

Command	Description
debug modem oob	Creates modem startup messages between the network management software and the modem on the specificed OOB port.
debug modem trace	Performs a call trace on the specified modem, which allows you to determine why calls are terminated.

debug modem dsip

To display output for modem control messages that are received or sent to the router, use the **debug modem dsip** privileged EXEC command. To disable the output, use the **no** form of this command.

debug modem dsip {*tty-range* | **group** | *shelf*/*slot*/*port*}

no debug modem dsip {*tty-range* | **group** | *shelf*/*slot*/*port*}

Syntax Description

tty-range	Modem tty number or range. You can specify a single TTY line number or a range from 0 through the number of modems you have in your Cisco AS5800 access server. Be sure to include a dash (-) between the range values you specify.
group	Modem group information.
shelf/slot/port	Location of the modem by shelf/slot/port numbers for internal modems.

Command History

Release	Modification
11.3(2)AA	This command was introduced.

Usage Guidelines

The **debug modem dsip** command displays each DSIP message that relates to a modem and is sent from or received at the router shelf. This command can be applied to a single modem or a group of modems.

Examples

The following examples show a display of the available **debug modem** command options and **debug modem dsip** command options:

```
Router# debug modem ?

  dsip         Modem DSIP activity
  maintenance  Modem maintenance activity
  oob          Modem out of band activity
  trace        Call Trace Upload
  traffic      Modem data traffic
  <cr>

Router# debug modem dsip ?

  <0-935>  First Modem TTY Number
  group    Modem group information
  x/y/z    Shelf/Slot/Port for Internal Modems
  <cr>
```

The following example indicates that an RTS status message was received from the router shelf, and an ACK message was sent back:

```
Router# debug modem dsip

00:11:02: RSMODEM_SEND-1/2/06: MODEM_RING_INDICATION_MSG cci1 si0 ms0 mm65535,0 dc0
00:11:02: RSMODEM_sRCV-1/2/06:l12,MODEM_CALL_ACK_MSG:
00:11:02: RSMODEM_SEND-1/2/06: MODEM_CALL_ACCEPT_MSG
00:11:11: RSMODEM_sRCV-2:10,MODEM_POLL_MSG: 0 16 0 7 0 146 0 36 21
00:11:18: RSMODEM_sRCV-1/2/06:l12,MODEM_SET_DCD_STATE_MSG: 1
```

```
00:11:19: RSMODEM_SEND-1/2/06: MODEM_RTS_STATUS_MSG 1
00:11:19: RSMODEM_dRCV-2:11258607996,MODEM_RTS_STATUS_MSG: 0 6 0 23 0 0 0 0 0
00:11:23: RSMODEM_sRCV-2:10,MODEM_POLL_MSG: 0 16 0 7 0 146 0 150 21
00:12:31: RSMODEM_sRCV-1/2/06:112,MODEM_SET_DCD_STATE_MSG: 0
00:12:31: RSMODEM_SEND-1/2/06: MODEM_CALL_HANGUP_MSG
00:12:31: RSMODEM_sRCV-1/2/06:112,MODEM_ONHOOK_MSG:
00:12:32: RSMODEM_SEND-1/2/06: MODEM_RTS_STATUS_MSG 1
00:12:32: RSMODEM_SEND-1/2/06: MODEM_SET_DTR_STATE_MSG 0
00:12:32: RSMODEM_dRCV-2:11258659676,MODEM_RTS_STATUS_MSG: 0 6 0 16 0 0 0 0 0
00:12:32: RSMODEM_SEND-1/2/06: MODEM_RTS_STATUS_MSG 1
00:12:32: RSMODEM_dRCV-2:11258600700,MODEM_RTS_STATUS_MSG: 0 6 0 13 0 0 0 0 0
00:12:33: RSMODEM_SEND-1/2/06: MODEM_SET_DTR_STATE_MSG 0
00:12:33: RSMODEM_SEND-1/2/06: MODEM_RTS_STATUS_MSG 1
00:12:33: RSMODEM_dRCV-2:11258662108,MODEM_RTS_STATUS_MSG: 0 6 0 16 0 0 0 0 0
00:12:35: RSMODEM_sRCV-2:10,MODEM_POLL_MSG: 0 16 0 7 0 146 1 34 22
00:12:38: RSMODEM_SEND-1/2/06: MODEM_SET_DTR_STATE_MSG 1
00:12:47: RSMODEM_sRCV-2:10,MODEM_POLL_MSG: 0 16 0 7 0 146 0 12 22
```

Table 123 describes the significant fields shown in the display.

Table 123 debug modem dsip Field Descriptions

Field	Description
RSMODEM_SEND-1/2/06	Router shelf modem shelf sends a MODEM_RING_INDICATION_MSG message.
RSMODEM_sRCV-1/2/06	Router shelf modem received a MODEM_CALL_ACK_MSG message.
MODEM_CALL_ACCEPT_MSG	Router shelf accepts the call.
MODEM_CALL_HANGUP_MSG	Router shelf sends a hangup message.
MODEM_RTS_STATUS_MSG	Request to send message status.

Related Commands

Command	Description
debug modem traffic	Displays output for framed, unframed, and asynchronous data transmission received from the modem cards.
debug dsip	Displays output for DSIP used between the router shelf and the dial shelf.

debug modem oob

To debug the out-of-band port used to poll modem events on the modem, use the **debug modem oob** privileged EXEC command. The **no** form of this command disables debugging output.

debug modem oob [*slot/modem-port* | **group** *group-number*]

no debug modem oob [*slot/modem-port* | **group** *group-number*]

Syntax Description

slot/modem-port	(Optional) The slot and modem port number.
group *group-number*	(Optional) The modem group.

Usage Guidelines

The message types and sequence numbers that appear in the debug output are initiated by the Modem Out-of-Band Protocol and used by service personnel for debugging purposes.

Caution Entering the **debug modem oob** command without specifying a slot and modem number debugs *all* out-of-band ports, which generates a substantial amount of information.

Examples

The following is sample output from the **debug modem oob** command. This example debugs the out-of-band port on modem 2/0, which creates modem startup messages between the network management software and the modem.

```
Router# debug modem oob 2/0

MODEM(2/0): One message sent --Message type:3, Sequence number:0
MODEM(2/0): Modem DC session data reply
MODEM(2/0): One message sent --Message type:83, Sequence number:1
MODEM(2/0): DC session event =
MODEM(2/0): One message sent --Message type:82, Sequence number:2
MODEM(2/0): No status changes since last polled
MODEM(2/0): One message sent --Message type:3, Sequence number:3
MODEM(2/0): Modem DC session data reply
MODEM(2/0): One message sent --Message type:83, Sequence number:4
```

Related Commands

Command	Description
debug modem csm	Debugs the CSM used to connect calls on the modem.
debug modem trace	Performs a call trace on the specified modem, which allows you to determine why calls are terminated.

debug modem trace

To debug a call trace on the modem to determine why calls are terminated, use the **debug modem trace** privileged EXEC command. The **no** form of this command disables debugging output.

debug modem trace [**normal** | **abnormal** | **all**] [*slot/modem-port* | **group** *group-number*]

no debug modem trace [**normal** | **abnormal** | **all**] [*slot/modem-port* | **group** *group-number*]

Syntax Description

normal	(Optional) Uploads the call trace to the syslog server on normal call termination (for example, a local user hangup or a remote user hangup).
abnormal	(Optional) Uploads the call trace to the syslog server on abnormal call termination (for example, any call termination other than normal termination, such as a lost carrier or a watchdog timeout).
all	(Optional) Uploads the call trace on all call terminations including normal and abnormal call termination.
slot/modem-port	(Optional) The slot and modem port number.
group *group-number*	(Optional) The modem group.

Usage Guidelines

The **debug modem trace** command applies only to manageable modems. For additional information, use the **show modem** command.

Examples

The following is sample output from the **debug modem trace abnormal** command:

```
Router# debug modem trace abnormal 1/14

Modem 1/14 Abnormal End of Connection Trace. Caller 123-4567
    Start-up Response: AS5200 Modem, Firmware 1.0
    Control Reply: 0x7C01
    DC session response: brasil firmware 1.0
    RS232 event:
    DSR=On, DCD=On, RI=Off, TST=Off
    changes: RTS=No change, DTR=No change, CTS=No change
    changes: DSR=No change, DCD=No change, RI=No change, TST=No change
    Modem State event: Connected
    Connection event: Speed = 19200, Modulation = VFC
    Direction = Originate, Protocol = reliable/LAPM, Compression = V42bis
    DTR event: DTR On
    Modem Activity event: Data Active
    Modem Analog signal event: TX = -10, RX = -24, Signal to noise = -32
    End connection event: Duration = 10:34-11:43,
    Number of xmit char =    67, Number of rcvd char = 88, Reason: Watchdog Time-out.
```

Related Commands

Command	Description
debug modem csm	Debugs the CSM used to connect calls on the modem.
debug modem oob	Creates modem startup messages between the network management software and the modem on the specificed OOB port.

debug modem traffic

To display output for framed, unframed, and asynchronous data sent received from the modem cards, use the **debug modem traffic** privileged EXEC command. To disable output, use the **no** form of this command.

debug modem traffic

no debug modem traffic

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)AA	This command was introduced.

Usage Guidelines

The **debug modem traffic** command displays output for framed, unframed, and asynchronous data sent or received by the modem cards.

Examples

The following example displays information about unframed or framed data sent to or received from the modem cards:

```
Router# debug modem traffic

MODEM-RAW-TX:modem = 6/5/00, length = 1, data = 0x61, 0xFF, 0x7D, 0x23
MODEM-RAW-RX:modem = 6/5/00, length = 1, data = 0x61, 0x0, 0x0, 0x0
```

The information indicates unframed asynchronous data transmission and reception involving the modem on shelf 6, slot 5, port 00.

The following example displays framed asynchronous data transmission and reception involving the modem on shelf 6, slot 5, port 00:

```
Router# debug modem traffic

MODEM-FRAMED-TX:modem = 6/5/00, length = 8, data = 0xFF, 0x3, 0x82
MODEM-FRAMED-RX:modem = 6/5/00, length = 14, data = 0xFF, 0x3, 0x80
```

Related Commands

Command	Description
debug modem dsip	Displays output for modem control messages that are received or sent to the router.

debug mpls adjacency

To display changes to label switching entries in the adjacency database, use the **debug mpls adjacency** EXEC command. The **no** form of this command disables debugging output.

debug mpls adjacency

no debug mpls adjacency

Usage Guidelines

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Usage Guidelines

Use the **debug mpls adjacency** command to monitor when entries are updated in or added to the adjacency database.

Examples

The following is sample output generated by the **debug mpls adjacency** command:

```
Router# debug mpls adjacency

TAG ADJ: add 10.10.0.1, Ethernet0/0/0
TAG ADJ: update 10.10.0.1, Ethernet0/0/0
```

Table 124 describes the significant fields shown in the sample display above.

Table 124 debug mpls adjacency Command Field Description

Field	Description
add	Adding an entry to the database.
update	Updating the MAC address for an existing entry.
10.10.0.1	Address of neighbor TSR.
Ethernet0/0/0	Connecting interface.

debug mpls ldp backoff

To display information about the label distribution protocol (LDP) backoff mechanism parameters, use the **debug mpls ldp backoff** command in privileged EXEC mode. To disable this feature, use the **no** form of this command.

debug mpls ldp backoff

no debug mpls ldp backoff

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(10)ST	This command was introduced.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.
12.1(8a)E	This command was integrated into Cisco IOS Release 12.1(8a)E.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.

Usage Guidelines

Use this command to monitor backoff parameters configured for LDP sessions.

Examples

The following shows sample output from the **debug mpls ldp backoff** command:

```
Router# debug mpls ldp backoff

LDP session establishment backoff debugging is on

Router#

Jan 6 22:31:13.012: ldp: Backoff peer ok: 12.12.12.12:0; backing off; threshold/count 8/6
Jan 6 22:31:13.824: ldp: Backoff peer ok: 12.12.12.12:1; backing off; threshold/count 8/6
Jan 6 22:31:17.848: ldp: Backoff peer ok: 12.12.12.12:0; backing off; threshold/count 8/6
Jan 6 22:31:18.220: ldp: Backoff peer ok: 12.12.12.12:1; backing off; threshold/count 8/6
Jan 6 22:31:21.908: ldp: Backoff peer ok: 12.12.12.12:0; backing off; threshold/count 8/6
Jan 6 22:31:22.980: ldp: Backoff peer ok: 12.12.12.12:1; backing off; threshold/count 8/6
Jan 6 22:31:25.724: ldp: Backoff peer ok: 12.12.12.12:0; backing off; threshold/count 8/7
Jan 6 22:31:26.944: ldp: Backoff peer ok: 12.12.12.12:1; backing off; threshold/count 8/7
Jan 6 22:31:30.140: ldp: Backoff peer ok: 12.12.12.12:0; backing off; threshold/count 8/7
Jan 6 22:31:31.932: ldp: Backoff peer ok: 12.12.12.12:1; backing off; threshold/count 8/7
Jan 6 22:31:35.028: ldp: Backoff peer ok: 12.12.12.12:0; backing off; threshold/count 8/7
Jan 6 22:31:35.788: ldp: Backoff peer ok: 12.12.12.12:1; backing off; threshold/count 8/7
```

```
Jan 6 22:31:39.332: ldp: Update backoff rec: 12.12.12.12:0, threshold = 8, tbl ents 2
Jan 6 22:31:39.640: ldp: Update backoff rec: 12.12.12.12:1, threshold = 8, tbl ents 2
```

Table 125 describes the significant fields shown in the display.

Table 125 debug mpls ldp backoff Field Descriptions

Field	Description
ldp	Identifies the Label Distribution Protocol.
Backoff peer ok: a.b.c.d:n	Identifies the LDP peer for which a session is being delayed because of a failure to establish a session due to incompatible configuration.
backing off;	Indicates that a session setup attempt failed and the LSR is delaying its next attempt (that is, is backing off).
threshold/count x/y	Identifies a set threshold (x) and a count (y) that represents the time that has passed since the last attempt to set up a session with the peer. The count is incremented every 15 seconds until it reaches the threshold. When the count equals the threshold, a fresh attempt is made to set up an LDP session with the peer.
Update backoff rec	Indicates that the backoff period is over and that it is time for another attempt to set up an LDP session.
threshold = x	Indicates the backoff time of x*15 seconds, for the next LDP session attempt with the peer.
tbl ents 2	Indicates unsuccessful attempts to set up an LDP session with two different LDP peers. In this example, attempts to set up sessions with LDP peers 12.12.12.12:0 and 12.12.12.12:1 are failing.

Related Commands

Command	Description
mpls ldp backoff	Configures session setup delay parameters for the LDP backoff mechanism.
show mpls ldp backoff	Displays information about the configured session setup backoff parameters and any potential LDP peers with which session setup attempts are being throttled.

debug mpls events

To display information about significant MPLS events, use the **debug mpls events** privileged EXEC command. Use the **no** form of this command to disable this feature.

debug mpls events

no debug mpls events

Syntax Description

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

Use this command to monitor significant MPLS events. For this Cisco IOS release, the only events reported by this command are changes to the MPLS router ID.

Examples

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

```
Router# debug mpls events

MPLS events debugging is on

TAGSW: Unbound IP address, 155.0.0.55, from Router ID
TAGSW: Bound IP address, 199.44.44.55, to Router ID
```

debug mpls lfib cef

To print detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed, use the **debug mpls lfib cef** EXEC command. The **no** form of this command disables debugging.

debug mpls lfib cef

no debug mpls lfib cef

Syntax Description

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Usage Guidelines

Several lines of output are produced for each route placed into the LFIB. If your router has thousands of labeled routes, be careful about issuing this command. When label switching is first enabled, each of these routes is placed into the LFIB, and several lines of output are displayed for each route.

Examples

The following is sample output displayed when you enter the **debug mpls lfib cef** command:

```
Router# debug mpls lfib cef

Cisco Express Forwarding related TFIB services debugging is on

tagcon: tc_ip_rtlookup fail on 10.0.0.0/8:subnet_lookup failed
TFIB: route tag chg 10.7.0.7/32,idx=1,inc=Withdrn,outg=Withdrn,enabled=0x2
TFIB: fib complete delete: prefix=10.7.0.7/32,inc tag=26,delete_info=1
TFIB: deactivate tag rew for 10.7.0.7/32,index=0
TFIB: set fib rew: pfx 10.7.0.7/32,index=0,add=0,tag_rew->adj=Ethernet2/3
TFIB: resolve tag rew,prefix=10.7.0.7/32,no tag_info,no parent
TFIB: fib scanner start:needed:1,unres:0,mac:0,loadinfo:0
TFIB: resolve tag rew,prefix=10.7.0.7/32,no tag_info,no parent
TFIB: fib upd loadinf 10.100.100.100/32,tag=Tun_hd,fib no loadin,tfib no loadin
TFIB: fib check cleanup for 10.100.100.100/32,index=0,return_value=0
TFIB: fib_scanner_end
TFIB: create dynamic entry for 10.11.0.11/32
TFIB: call find_route_tags,dist_method=1,next_hop=10.93.0.11,Et2/3
TFIB: route tag chg 10.11.0.11/32,idx=0,inc=26,outg=Unkn,enabled=0x3
TFIB: create tag info 10.11.0.11/32,inc tag=26,has no info
TFIB: resolve tag rew,prefix=10.11.0.11/32,has tag_info,no parent
```

```
TFIB: finish fib res 10.11.0.11/32:index 0,parent outg tag no parent
TFIB: fib upd loadinf 10.11.0.11/32,tag=26,fib no loadin,tfib no loadin
TFIB: set fib rew: pfx 10.11.0.11/32,index=0,add=1,tag_rew->adj=Ethernet2/3
tagcon: route_tag_change for: 10.250.0.97/32
        intag 33, outtag 28, nexthop tsr 10.11.0.11:0
TFIB: route tag chg 10.250.0.97/32,idx=0,inc=33,outg=28,enabled=0x3
TFIB: deactivate tag rew for 10.250.0.97/32,index=0
TFIB: set fib rew: pfx 10.250.0.97/32,index=0,add=0,tag_rew->adj=Ethernet2/3
TFIB: create tag info 10.250.0.97/32,inc tag=33,has old info
On VIP:
TFIB: route tag chg 10.13.72.13/32,idx=0,inc=34,outg=Withdrn,enabled=0x3
TFIB: deactivate tag rew for 10.13.72.13/32,index=0
TFIB: set fib rew: pfx 10.13.72.13/32,index=0,add=0,tag_rew->adj=
TFIB: create tag info 10.13.72.13/32,inc tag=34,has old info
TFIB: resolve tag rew,prefix=10.13.72.13/32,has tag_info,no parent
TFIB: finish fib res 10.13.72.13/32:index 0,parent outg tag no parent
TFIB: set fib rew: pfx 10.100.100.100/32,index=0,add=0,tag_rew->adj=
TFIB: create tag info 10.100.100.100/32,inc tag=37,has old info
TFIB: resolve tag rew,prefix=10.100.100.100/32,has tag_info,no parent
TFIB: finish fib res 10.100.100.100/32:index 0,parent outg tag no parent
TFIB: fib upd loadinf 10.100.100.100/32,tag=37,fib no loadin,tfib no loadin
```

Table 126 lists the significant fields shown in the display.

See Table 128 for a description of special labels that appear in the output of this debug command.

Table 126 debug mpls lfib cef Field Descriptions

Field	Description
tagcon	The name of the subsystem issuing the debug output (Label Control).
LFIB	The name of the subsystem issuing the debug output.
tc_ip_rtlookup fail on x.y.w.z/m: subnet_lookup failed	The destination with IP address and mask shown is not in the routing table.
route tag chg x.y.w.z/m	Request to create the LFIB entry for the specified prefix/mask.
idx=-1	The index within the FIB entry of the path whose LFIB entry is being created. The parameter –1 means all paths for this FIB entry.
inc=s	Incoming label of the entry being processed.
outg=s	Outgoing label of the entry being processed.
enabled=0xn	Bit mask indicating the types of label switching currently enabled: • 0x1 = dynamic • 0x2 = TSP tunnels • 0x3 = both
fib complete delete	Indicates that the FIB entry is being deleted.
prefix=x.y.w.z/m	A destination prefix.
delete_info=1	Indicates that label_info is also being deleted.
deactivate tag rew for x.y.w.z/m	Indicates that label rewrite for specified prefix is being deleted.
index=n	Index of path in the FIB entry being processed.
set fib rew: pfx x.y.w.z/m	Indicates that label rewrite is being installed or deleted from the FIB entry for the specified destination for label imposition purposes.

Table 126 debug mpls lfib cef Field Descriptions (continued)

Field	Description
add=0	Indicates that label rewrite is being deleted from the FIB (no longer imposing labels).
tag_rew->adj=s	Adjacency of label rewrite for label imposition.
resolve tag rew,prefix=x.y.w.z/m	Indicates that the FIB route to the specified prefix is being resolved.
no tag_info	Indicates that there is no label_info for the destination (destination not labeled).
no parent	Indicates that the route is not recursive.
fib scanner start	Indicates that the periodic scan of the FIB has started.
needed:1	Indicates that the LFIB needs the FIB to be scanned.
unres:n	Indicates the number of unresolved TFIB entries.
mac:n	Indicates the number of TFIB entries missing MAC strings.
loadinfo:n	Indicates whether the nonrecursive accounting state has changed and whether the loadinfo information in the LFIB needs to be adjusted.
fib upd loadinf x.y.w.z/m	Indicates that a check for nonrecursive accounting is being made and that the LFIB loadinfo information for the specified prefix is being updated.
tag=s	Incoming label of entry.
fib no loadin	Indicates that the corresponding FIB entry has no loadinfo.
tfib no loadin	Indicates that the LFIB entry has no loadinfo.
fib check cleanup for x.y.w.z/m	Indicates that a check is being made on the LFIB entry for the specified destination to determine if rewrite needs to be removed from the LFIB.
return_value=x	If *x* is 0, indicates that no change has occurred in the LFIB entry. If *x* is 1, there was a change.
fib_scanner_end	Indicates that the FIB scan has come to an end.
create dynamic entry for x.y.w.z/m	Indicates that the LFIB has been enabled and that an LFIB entry is being created for the specified destination.
call find_route_tags	Indicates that the labels for that destination are being requested.
dist_method=n	Identifies the label distribution method—TDP, TC-ATM, and so on.
next_hop=x.y.z.w	Identifies the next hop for the destination.
interface name	Identifies the outgoing interface for the destination.
create tag info	Indicates that a label_info data structure is being created for the destination.
has no info	Indicates that the destination does not already have label_info.
finish fib re x.y.z.w/m	Indicates that the LFIB entry for the specified route is being completed.
parent outg tag s	If recursive, specifies the outgoing label of the route through which it is recursive (the parent). If not recursive, *s* = “no parent.”
tagcon: route_tag_change for: x.y.z.w/m	Indicates that label control is notifying LFIB that labels are available for the specified destination.
intag s	Identifies the incoming label for the destination.

Table 126 debug mpls lfib cef Field Descriptions (continued)

Field	Description
outtag s	Identifies the outgoing label for the destination.
nexthop tsr x.y.z.w.i	Identifies the TDP ID of the next hop that sent the tag.

Related Commands

Command	Description
debug mpls lfib cef	Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed.
debug mpls lfib lsp	Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed.
debug mpls lfib state	Traces what happens when label switching is enabled or disabled.
debug mpls lfib struct	Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label rewrites, and label_info data.

debug mpls lfib enc

To print detailed information about label encapsulations while label rewrites are created or updated and placed in the label forwarding information base (LFIB), use the **debug mpls lfib enc** privileged EXEC command. The **no** form of this command disables debugging output.

debug mpls lfib enc

no debug mpls lfib enc

Syntax Description

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Usage Guidelines

Several lines of output are produced for each route placed into the LFIB. If your router has thousands of labeled routes, issue this command with care. When label switching is first enabled, each of these routes is placed into the LFIB and a label encapsulation is created. The command output shows you on which adjacency the label rewrite is being created and the labels assigned.

Examples

The following is an example of output generated when you issue the **debug mpls lfib enc** command. This example shows the encapsulations for three routes that have been created and placed into the LFIB.

```
Router# debug mpls lfib enc

TFIB: finish res:inc tag=28,outg=Imp_null,next_hop=10.93.72.13,Ethernet4/0/3
TFIB: update_mac, mac_length = 14,addr=10.93.72.13,idb=Ethernet4/0/3
TFIB: get ip adj: addr=10.93.72.13,is_p2p=0,fibidb=Ethernet4/0/3,linktype=7
TFIB: get tag adj: addr=10.93.72.13,is_p2p=0,fibidb=Ethernet4/0/3,linktype=79
TFIB: encaps:inc=28,outg=Imp_null,idb:Ethernet4/0/3,sizes 14,14,1504,type 0
TFIB: finish res:inc tag=30,outg=27,next_hop=10.93.72.13,Ethernet4/0/3
TFIB: get ip adj: addr=10.93.72.13,is_p2p=0,fibidb=Ethernet4/0/3,linktype=7
TFIB: get tag adj: addr=10.93.72.13,is_p2p=0,fibidb=Ethernet4/0/3,linktype=79
TFIB: encaps:inc=30,outg=27,idb:Ethernet4/0/3,sizes 14,18,1500,type 0
TFIB: finish res:inc tag=30,outg=10,next_hop=0.0.0.0,ATM0/0.1
TFIB: get ip adj: addr=0.0.0.0,is_p2p=1,fibidb=ATM0/0.1,linktype=7
TFIB: get tag adj: addr=0.0.0.0,is_p2p=1,fibidb=ATM0/0.1,linktype=79
TFIB: encaps:inc=30,outg=10,idb:ATM0/0,sizes 4,8,4470,type 1
```

Table 127 describes the significant fields shown in the display.

Table 127 debug mpls lfib enc Field Descriptions

Field	Description
TFIB	Identifies the source of the message as the LFIB subsystem.
finish res	Identifies that the LFIB resolution is being finished.
inc tag=x or inc=x	An incoming (local) label for the LFIB entry is being created. Labels can be numbers or special values.
outg=y	An outgoing (remote) label for the LFIB entry is being created.
next_hop=a.b.c.d	IP address of the next hop for the destination.
interface	The outgoing interface through which a packet will be sent.
get ip adj	Identifies that the IP adjacency to use in the LFIB entry is being determined.
get tag adj	Identifies that the label switching adjacency to use for the LFIB entry is being determined.
addr = a.b.c.d	The IP address of the adjacency.
is_p2p=x	If *x* is 1, this is a point-to-point adjacency. If *x* is 0, it is not.
fibidb = s	Indicates the interface of the adjacency.
linktype = x	The link type of the adjacency, as follows: • 7 = LINK_IP • 79 = LINK_TAG
sizes x,y,z	Indicates the following values: • x = length of macstring • y = length of tag encapsulation • z = tag MTU
type = x	Tag encapsulation type, as follows: • 0 = normal • 1 = TCATM • 2 = TSP tunnel
idb:s	Indicates the outgoing interface.
update_mac	Indicates that the macstring of the adjacency is being updated.

Table 128 describes the special labels, which sometimes appear in the debug output, and their meanings.

Table 128 Special Labels Appearing in debug Command Output

Special Label	Meaning
Unassn—Inital value	No label assigned yet.
Unused	This destination does not have a label (for example, a BGP route).
Withdrn	The label for this destination has been withdrawn.
Unkn	This destination should have a label, but it is not yet known.
Get_res	A recursive route that will get a label when resolved.

Table 128 Special Labels Appearing in debug Command Output (continued)

Special Label	Meaning
Exp_null	Explicit null label—used over TC-ATM.
Imp_null	Implicit null label—for directly connected routes.
Tun_hd	Identifies head of TSP tunnel.

Related Commands

Command	Description
debug mpls lfib cef	Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed.
debug mpls lfib lsp	Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed.
debug mpls lfib state	Traces what happens when label switching is enabled or disabled.
debug mpls lfib struct	Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label rewrites, and label_info data.

debug mpls lfib lsp

To print detailed information about label rewrites being created and deleted as LSP tunnels are added or removed, use the **debug mpls lfib lsp** EXEC command. The **no** form of this command disables debugging output.

debug mpls lfib lsp

no debug mpls lfib lsp

Syntax Description

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Examples

The following is sample output generated from the **debug mpls lfib lsp** command:

```
Router# debug mpls lfib lsp

TSP-tunnel related TFIB services debugging is on

TFIB: tagtun,next hop=10.93.72.13,inc=35,outg=1,idb=Et4/0/3
TFIB: tsptunnel:next hop=10.93.72.13,inc=35,outg=Imp_null,if_number=7
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun tag chg linec,fiblc=0,in tg=35,o tg=1,if=7,nh=10.93.72.13
TFIB: tagtun,next hop=10.92.0.7,inc=36,outg=1,idb=Et4/0/2
TFIB: tsptunnel:next hop=10.92.0.7,inc=36,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=36,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun tag chg linec,fiblc=0,in tg=36,o tg=1,if=6,nh=10.92.0.7
TFIB: tagtun_delete, inc = 36
tagtun tag del linec,itag=12
TFIB: tagtun_delete, inc = 35
tagtun tag del linec,itag=12
TFIB: tagtun,next hop=10.92.0.7,inc=35,outg=1,idb=Et4/0/2
TFIB: tsptunnel:next hop=10.92.0.7,inc=35,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun tag chg linec,fiblc=0,in tg=35,o tg=1,if=6,nh=10.92.0.7

On VIP:
TFIB: tagtun chg msg,in tg=35,o tg=1,nh=10.93.72.13,if=7
TFIB: tsptunnel:next hop=10.93.72.13,inc=35,outg=Imp_null,if_number=7
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=36,o tg=1,nh=10.92.0.7,if=6
TFIB: tsptunnel:next hop=10.92.0.7,inc=36,outg=Imp_null,if_number=6
```

```
TFIB: tsptun update loadinfo:tag=36,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=35,o tg=1,nh=10.93.72.13,if=7
TFIB: tsptunnel:next hop=10.93.72.13,inc=35,outg=Imp_null,if_number=7
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=36,o tg=1,nh=10.92.0.7,if=6
TFIB: tsptunnel:next hop=10.92.0.7,inc=36,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=36,loadinfo_reqd=0,no new loadinfo,no old loadinfo
TFIB: tagtun chg msg,in tg=35,o tg=1,nh=10.92.0.7,if=6
TFIB: tsptunnel:next hop=10.92.0.7,inc=35,outg=Imp_null,if_number=6
TFIB: tsptun update loadinfo:tag=35,loadinfo_reqd=0,no new loadinfo,no old loadinfo
```

Table 129 describes the significant fields in the sample display shown above.

Table 129 debug mpls lfib lsp Field Descriptions

Field	Description
tagtun	Name of routine entered.
next hop=x.y.z.w	Next hop for the tunnel being created.
inc=x	Incoming label for this hop of the tunnel being created.
outg=x	Outgoing label (1 means Implicit Null label).
idb=s	Outgoing interface for the tunnel being created.
if_number=7	Interface number of the outgoing interface.
tsptunnel	Name of the routine entered.
tsptun update loadinfo	The procedure being performed.
tag=x	Incoming label of the LFIB slot whose loadinfo is being updated.
loadinfo_reqd=x	Indicates whether a loadinfo is expected for this entry (non-recursive accounting is on).
no new loadinfo	No change required in loadinfo.
no old loadinfo	No previous loadinfo available.
tagtun tag chg linec	Line card is being informed of the TSP tunnel.
fiblc=x	Indicates which line card is being informed (0 means all).
in tg=x	Indicates the incoming label of new TSP tunnel.
o tg=x	Indicates the outgoing label of new TSP tunnel.
if=x	Indicates the outgoing interface number.
nh=x.y.w.z	Indicates the next hop IP address.
tagtun_delete	Indicates that a procedure is being performed: delete a TSP tunnel.
tagtun tag del linec	Informs the line card of the TSP tunnel deletion.
tagtun chg msg	Indicates that the line card has received a message to create a TSP tunnel.

Related Commands

Command	Description
debug mpls lfib cef	Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed.
debug mpls lfib state	Traces what happens when label switching is enabled or disabled.
debug mpls lfib struct	Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label rewrites, and label_info data.

debug mpls lfib state

To trace what happens when label switching is enabled or disabled, use the **debug mpls lfib state** EXEC command. The **no** form of this command disables debugging output.

debug mpls lfib state

no debug mpls lfib state

Syntax Description

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Usage Guidelines

Use this command when you wish to trace what happens to the LFIB when you issue the **mpls ip** or the **mpls tsp-tunnel** command.

Examples

The following is sample output generated from the **debug mpls lfib state** command:

```
Router# debug mpls lfib state

TFIB enable/disable state debugging is on
TFIB: Upd tag sb 6(status:0xC1,tmtu:1500,VPI:1-1 VC=0/32,et:0/0/0),lc 0x0
TFIB: intf status chg: idb=Et4/0/2,status=0xC1,oldstatus=0xC3
TFIB: interface dyntag change,change in state to Ethernet4/0/2
TFIB: enable entered, table exists,enabler type=0x2
TFIB: enable, TFIB already enabled, types now 0x3,returning
TFIB: enable entered, table exists,enabler type=0x1
TFIB: disable entered, table exists,type=0x1

TFIB: cleanup: tfib[32] still non-0

On linecard only:

TFIB: disable lc msg recvd, type=0x1
TFIB: Ethernet4/0/1 fibidb subblock message received
TFIB: enable lc msg recvd, type=0x1
TFIB: Tunnel301 set encapfix to 0x6016A97C
```

Table 130 describes the significant fields shown in the display.

Table 130 debug mpls lfib state Field Descriptions

Field	Description
LFIB	Identifies the source of the message as the LFIB subsystem.
Upd tag sb x	Indicates that the status of the "*x*th" label switching sub-block is being updated, where *x* is the interface number. There is a label switching sub-block for each interface on which label switching has been enabled.
(status:0xC1,tmtu:1500, VPI:1-1VC=0/32, et:0/0/0),lc 0x0)	Identifies the values of the fields in the label switching sub-block, as follows: • status byte • maximum transmission unit (*tmtu*) • range of ATM VPs • control VP • control VC (if this is a TC-ATM interface) • encapsulation type (*et*) • encapsulation information • tunnel interface number (*lc*) • line card number to which the update message is being sent (0 means all line cards)
intf status chg	Indicates that there was an interface status change.
idb=Et4/0/2	Identifies the interface whose status changed.
status=0xC1	Indicates the new status bits in the label switching sub-block of the idb.
oldstatus=0xC3	Indicates the old status bits before the change.
interface dyntag change, change in state to Ethernet4/0/2	Indicates that there was a change in the dynamic label status for the particular interface.
enable entered	Indicates that the code that enables the LFIB was invoked.
TFIB already enabled	Indicates that the LFIB was already enabled when this call was made.
table exists	Indicates that an LFIB table had already been allocated in a previous call.
cleanup: tfib[x] still non-0	Indicates that the LFIB is being deleted, but that slot *x* is still active.
disable lc mesg recvd, type=0x1	Indicates that a message to disable label switching type 1 (dynamic) was received by the line card.
disable entered, table exists,type=0x1	Indicates that a call to disable dynamic label switching was issued.
Ethernet4/0/1 fibidb subblock message received	Indicates that a message giving fibidb status change was received on the line card.
enable lc msg recvd,type=0x1	Indicates that the line card received a message to enable label switching type 1 (dynamic).

Table 130 debug mpls lfib state Field Descriptions (continued)

Field	Description
Tunnel301 set encapfix to 0x6016A97C	Shows that fibidb Tunnel301 on the line card received an encapsulation fixup.
types now 0x3, returning	Shows the value of the bitmask indicating the type of label switching enabled on the interface, as follows: • 0x1—means dynamic label switching • 0x2—means tsp-tunnels • 0x3—means both

Related Commands

Command	Description
debug mpls lfib cef	Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed.
debug mpls lfib lsp	Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed.
debug mpls lfib state	Traces what happens when label switching is enabled or disabled.
debug mpls lfib struct	Traces the allocation and freeing of LFIB-related data structures, including the LFIB itself, label rewrites, and label_info data.

debug mpls lfib struct

To trace the allocation and freeing of LFIB-related data structures, such as the LFIB itself, label rewrites, and label_info data, use the **debug mpls lfib struct** EXEC command. The **no** form of this command disables debugging output.

debug mpls lfib struct

no debug mpls lfib struct

Syntax Description

This command has no keywords or arguments.

Defaults

This command has no default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Examples

The following is sample output generated from the **debug mpls lfib struct** command:

```
Router# debug mpls lfib struct

TFIB data structure changes debugging is on

TFIB: delete tag rew, incoming tag 32
TFIB: remove from tfib,inc tag=32
TFIB: set loadinfo,tag=32,no old loadinfo,no new loadinfo
TFIB: TFIB not in use.  Checking for entries.
TFIB: cleanup: tfib[0] still non-0
TFIB: remove from tfib,inc tag=Tun_hd
TFIB: set loadinfo,tag=Exp_null,no old loadinfo,no new loadinfo
TFIB: TFIB freed.
TFIB: enable, TFIB allocated, size 4024 bytes, maxtag = 500
TFIB: create tag rewrite: inc Tun_hd,outg Unkn
TFIB: add to tfib at Tun_hd, first in circular list, mac=0,enc=0
TFIB: delete tag rew, incoming tag Tun_hd
TFIB: remove from tfib,inc tag=Tun_hd
TFIB: set loadinfo,tag=Exp_null,no old loadinfo,no new loadinfo
TFIB: create tag rewrite: inc Tun_hd,outg Unkn
TFIB: add to tfib at Tun_hd, first in circular list, mac=0,enc=0
TFIB: create tag rewrite: inc 26,outg Unkn
TFIB: add to tfib at 26, first in circular list, mac=0,enc=0
TFIB: add to tfib at 27, added to circular list, mac=0,enc=0
TFIB: delete tag rew, incoming tag Tun_hd
TFIB: remove from tfib,inc tag=Tun_hd
TFIB: set loadinfo,tag=Exp_null,no old loadinfo,no new loadinfo
TFIB: add to tfib at 29, added to circular list, mac=4,enc=8
```

```
TFIB: delete tag rew, incoming tag 29
TFIB: remove from tfib,inc tag=29
```

Table 131 describes the significant fields shown in the display.

Table 131 debug mpls lfib struct Field Descriptions

Field	Description
TFIB	The subsystem issuing the message.
delete tag rew	A label rewrite is being freed.
remove from tfib	A label rewrite is being removed from the LFIB.
inc tag=s	The incoming label of the entry being processed.
set loadinfo	The loadinfo field in the LFIB entry is being set (used for nonrecursive accounting).
tag=s	The incoming label of the entry being processed.
no old loadinfo	The LFIB entry did not have a loadinfo before.
no new loadinfo	The LFIB entry should not have a loadinfo now.
TFIB not in use. Checking for entries.	Label switching has been disabled and the LFIB is being freed up.
cleanup: tfib[x] still non-0	The LFIB is being checked for any entries in use, and entry *x* is the lowest numbered slot still in use.
TFIB freed	The LFIB table has been freed.
enable, TFIB allocated, size x bytes, maxtag = y	Label switching has been enabled and an LFIB of *x* bytes has been allocated. The largest legal label is *y*.
create tag rewrite	A label rewrite is being created.
inc s	The incoming label.
outg s	The outgoing label.
add to tfib at s	A label rewrite has been placed in the LFIB at slots.
first in circular list	This LFIB slot had been empty and this is the first rewrite in the list.
mac=0,enc=0	Length of the mac string and total encapsulation length, including labels.
added to circular list	A label rewrite is being added to an LFIB slot that already had an entry. This rewrite is being inserted in the circular list.

Related Commands

Command	Description
debug mpls lfib cef	Prints detailed information about label rewrites being created, resolved, and deactivated as CEF routes are added, changed, or removed.
debug mpls lfib lsp	Prints detailed information about label rewrites being created and deleted as LSP tunnels are added or removed.
debug mpls lfib state	Traces what happens when label switching is enabled or disabled.

debug mpls packets

To display labeled packets switched by the host router, use the **debug mpls packets** EXEC command. The **no** form of this command disables debugging output.

debug mpls packets [*interface*]

no debug mpls packets [*interface*]

Syntax Description

interface	(Optional.) The interface or subinterface name.

Defaults

Displays all labeled packets regardless of interface.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.1CT	This command was introduced.
12.1(3)T	This command was modified to reflect new MPLS IETF terminology and CLI command syntax.

Usage Guidelines

The optional *interface* parameter restricts the display to only those packets received or sent on the indicated interface.

Note Use this command with care because it generates output for every packet processed. Furthermore, enabling this command causes fast and distributed label switching to be disabled for the selected interfaces. To avoid adversely affecting other system activity, use this command only when traffic on the network is at a minimum.

Examples

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

```
Router# debug mpls packets

TAG: Hs3/0: recvd: CoS=0, TTL=254, Tag(s)=27
TAG: Hs0/0: xmit: (no tag)

TAG: Hs0/0: recvd: CoS=0, TTL=254, Tag(s)=30
TAG: Hs3/0: xmit: CoS=0, TTL=253, Tag(s)=27
```

Table 132 describes the significant fields shown in the display.

Table 132 debug mpls packets Field Descriptions

Field	Description
Hs0/0	The identifier for the interface on which the packet was received or sent.
recvd	Packet received.
xmit	Packet transmitted.
CoS	Class of Service field from the packet label header.
TTL	Time to live field from the packet label header.
(no tag)	Last label popped off the packet and were sent unlabeled.
Tag(s)	A list of labels on the packet, ordered from the top of the stack to the bottom.

Related Commands

Command	Description
show mpls forwarding-table	Displays the contents of the MPLS forwarding table.

debug mpls traffic-eng areas

To print information about traffic engineering area configuration change events, use the **debug mpls traffic-eng areas** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng areas

no debug mpls traffic-eng areas

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about traffic engineering area configuration change events:

```
debug mpls traffic-eng areas

TE-AREAS:isis level-1:up event
TE-PCALC_LSA:isis level-1
```

debug mpls traffic-eng autoroute

To print information about automatic routing over traffic engineering tunnels, use the **debug mpls traffic-eng autoroute** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng autoroute

no debug mpls traffic-eng autoroute

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about automatic routing over traffic engineering tunnels:

```
debug mpls traffic-eng autoroute

TE-Auto:announcement that destination 0001.0000.0003.00 has 1 tunnels
    Tunnel1 (traffic share 333, nexthop 10.112.0.12)
```

debug mpls traffic-eng link-management admission-control

To print information about traffic engineering LSP admission control on traffic engineering interfaces, use the **debug mpls traffic-eng link-management admission-control** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management admission-control [**detail**] [*aclnum*]

no debug mpls traffic-eng link-management admission-control [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information only for those LSPs that match the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword and the *aclnum* argument were added.

Examples

In the following example, information is printed about traffic engineering LSP admission control on traffic engineering interfaces:

```
debug mpls traffic-eng link-management admission-control

TE-LM-ADMIT:tunnel 10.106.0.6 1_10002:created [total 4]
TE-LM-ADMIT:tunnel 10.106.0.6 1_10002: "None" -> "New"
TE-LM-ADMIT:tunnel 10.106.0.6 1_10002: "New" -> "Admitting 2nd Path Leg"
TE-LM-ADMIT:tunnel 10.106.0.6 1_10002: "Admitting 2nd Path Leg" -> "Path Admitted"
TE-LM-ADMIT:Admission control has granted Path query for 10.106.0.6 1_10002 (10.112.0.12)
on link Ethernet4/0/1 [reason 0]
TE-LM-ADMIT:tunnel 10.106.0.6 1_10002: "Path Admitted" -> "Admitting 1st Resv Leg"
TE-LM-ADMIT:tunnel 10.106.0.6 1_10002: "Admitting 1st Resv Leg" -> "Resv Admitted"
TE-LM-ADMIT:Admission control has granted Resv query for 10.106.0.6 1_10002 (10.112.0.12)
on link Ethernet4/0/1 [reason 0]
```

debug mpls traffic-eng link-management advertisements

To print information about resource advertisements for traffic engineering interfaces, use the **debug mpls traffic-eng link-management advertisements** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management advertisements [**detail**] [*aclnum*]

no debug mpls traffic-eng link-management advertisements [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword was added.

Examples

In the following example, detailed debugging information is printed about resource advertisements for traffic engineering interfaces:

```
debug mpls traffic-eng link-management advertisements detail

TE-LM-ADV:area isis level-1:IGP announcement:link Et4/0/1:info changed
TE-LM-ADV:area isis level-1:IGP msg:link Et4/0/1:includes subnet type (2), described nbrs
(1)
TE-LM-ADV:area isis level-1:IGP announcement:link Et4/0/1:info changed
TE-LM-ADV:area isis level-1:IGP msg:link Et4/0/1:includes subnet type (2), described nbrs
(1)
TE-LM-ADV:LSA:Flooding manager received message:link information change (Et4/0/1)
TE-LM-ADV:area isis level-1:*** Flooding node information ***
  System Information::
    Flooding Protocol:   ISIS
  Header Information::
    IGP System ID:       0001.0000.0001.00
    MPLS TE Router ID:   10.106.0.6
    Flooded Links:       1
  Link ID:: 0
    Link IP Address:     10.1.0.6
    IGP Neighbor:        ID 0001.0000.0001.02
    Admin. Weight:       10
    Physical Bandwidth:  10000 kbits/sec
    Max Reservable BW:   5000 kbits/sec
    Downstream::
      Reservable Bandwidth[0]:       5000 kbits/sec
```

```
          Reservable Bandwidth[1]:        2000 kbits/sec
          Reservable Bandwidth[2]:        2000 kbits/sec
          Reservable Bandwidth[3]:        2000 kbits/sec
          Reservable Bandwidth[4]:        2000 kbits/sec
          Reservable Bandwidth[5]:        2000 kbits/sec
          Reservable Bandwidth[6]:        2000 kbits/sec
Attribute Flags: 0x00000000
```

Table 133 describes the significant fields shown in the display.

Table 133 debug isis mpls traffic-eng link-management advertisements Field Descriptions

Field	Description
Flooding Protocol	IGB that is flooding information for this area.
IGP System ID	Identification that IGP flooding uses in this area to identify this node.
MPLS TE Router ID	MPLS traffic engineering router ID.
Flooded Links	Number of links that are flooded in this area.
Link ID	Index of the link that is being described.
Link IP Address	Local IP address of this link.
IGP Neighbor	IGP neighbor on this link.
Admin. Weight	Administrative weight associated with this link.
Physical Bandwidth	Link’s bandwidth capacity (in kbps).
Max Reservable BW	Maximum amount of bandwidth that is currently available for reservation at this priority.
Reservable Bandwidth	Amount of bandwidth that is available for reservation.
Attribute Flags	Attribute flags of the link being flooded.

debug mpls traffic-eng link-management bandwidth-allocation

To print detailed information about bandwidth allocation for traffic engineering LSPs, use the **debug mpls traffic-eng link-management bandwidth-allocation** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management bandwidth-allocation [**detail**] [*aclnum*]

no debug mpls traffic-eng link-management bandwidth-allocation [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information only for those LSPs that match the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword and the *aclnum* argument were added.

Examples

In the following example, information is printed about bandwidth allocation for traffic engineering LSPs:

```
debug mpls traffic-eng link-management bandwidth-allocation
TE-LM-BW:tunnel 10.106.0.6 1_10002:requesting Downstream bw hold (3000000 bps [S]) on link
Et4/0/1
TE-LM-BW:tunnel 10.106.0.6 1_10002:Downstream bw hold request succeeded
TE-LM-BW:tunnel 10.106.0.6 1_10002:requesting Downstream bw lock (3000000 bps [S]) on link
Et4/0/1
TE-LM-BW:tunnel 10.106.0.6 1_10002:Downstream bw lock request succeeded×_„Rs
```

Related Commands

Command	Description
debug mpls traffic-eng link-management admission-control	Prints information about traffic engineering LSP admission control on traffic engineering interfaces.
debug mpls traffic-eng link-management errors	Prints information about errors encountered during any traffic engineering link management procedure.

debug mpls traffic-eng link-management errors

To print information about errors encountered during any traffic engineering link management procedure, use the **debug mpls traffic-eng link-management errors** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management errors [**detail**]

no debug mpls traffic-eng link-management errors [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, detailed debugging information is printed about errors encountered during a traffic engineering link management procedure:

```
debug mpls traffic-eng link-management errors detail

00:04:48 TE-LM-ROUTING: link Et1/1/1: neighbor 0010.0000.0012.01: add to IP peer db failed
```

Related Commands

Command	Description
debug mpls traffic-eng link-management admission-control	Prints information about traffic engineering LSP admission control on traffic engineering interfaces.
debug mpls traffic-eng link-management advertisements	Prints information about resource advertisements for traffic engineering interfaces.
debug mpls traffic-eng link-management bandwidth-allocation	Prints information about bandwidth allocation for traffic engineering LSPs.
debug mpls traffic-eng link-management events	Prints information about traffic engineering link management system events.
debug mpls traffic-eng link-management igp-neighbors	Prints information about changes to the link management databases of IGP neighbors.
debug mpls traffic-eng link-management links	Prints information about traffic engineering link management interface events.

debug mpls traffic-eng link-management events

To print information about traffic engineering link management system events, use the **debug mpls traffic-eng link-management events** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management events [**detail**]

no debug mpls traffic-eng link-management events [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword was added.

Examples

In the following example, detailed debugging information is printed about traffic engineering link management system events:

```
debug mpls traffic-eng link-management events detail

TE-LM-EVENTS:stopping MPLS TE Link Management process
TE-LM-EVENTS:MPLS TE Link Management process dying now
```

debug mpls traffic-eng link-management igp-neighbors

To print information about changes to the link management database of IGP neighbors, use the **debug mpls traffic eng link-management igp-neighbors** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management igp-neighbors [**detail**]

no debug mpls traffic-eng link-management igp-neighbors [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword was added.

Examples

In the following example, detailed debugging information is printed about changes to the link management database of IGP neighbors:

```
debug mpls traffic-eng link-management igp-neighbors detail

TE-LM-NBR:link AT0/0.2:neighbor 0001.0000.0002.00:created (isis level-1, 10.42.0.10,
Up)[total 2]
```

Related Commands

Command	Description
debug mpls traffic-eng link-management events	Prints information about traffic engineering-related ISIS events.

debug mpls traffic-eng link-management links

To print information about traffic engineering link management interface events, use the **debug mpls traffic-eng link-management links** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management links [**detail**]

no debug mpls traffic-eng link-management links [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword was added.

Examples

In the following example, detailed debugging information is printed about traffic engineering link management interface events:

```
debug mpls traffic-eng link-management links detail

TE-LM-LINKS:link AT0/0.2:RSVP enabled
TE-LM-LINKS:link AT0/0.2:increasing RSVP bandwidth from 0 to 5000000
TE-LM-LINKS:link AT0/0.2:created [total 2]
TE-LM-LINKS:Binding MPLS TE LM Admission Control as the RSVP Policy Server on ATM0/0.2
TE-LM-LINKS:Bind attempt succeeded
TE-LM-LINKS:link AT0/0.2:LSP tunnels enabled
```

debug mpls traffic-eng link-management preemption

To print information about traffic engineering LSP preemption, use the **debug mpls traffic-eng link-management preemption** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng link-management preemption [**detail**]

no debug mpls traffic-eng link-management preemption [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, detailed debugging information is printed about traffic engineering LSP preemption:

```
debug mpls traffic-eng link-management preemption detail

TE-LM-BW:preempting Downstream bandwidth, 1000000, for tunnel 10.106.0.6 2_2
TE-LM-BW:building preemption list to get bandwidth, 1000000, for tunnel 10.106.0.6 2_2
(priority 0)
TE-LM-BW:added bandwidth, 3000000, from tunnel 10.106.0.6 1_2 (pri 1) to preemption list
TE-LM-BW:preemption list build to get bw, 1000000, succeeded (3000000)
TE-LM-BW:preempting bandwidth, 1000000, using plist with 1 tunnels
TE-LM-BW:tunnel 10.106.0.6 1_2:being preempted on AT0/0.2 by 10.106.0.6 2_2
TE-LM-BW:preemption of Downstream bandwidth, 1000000, succeeded
```

debug mpls traffic-eng link-management routing

To print information about traffic engineering link management routing resolutions that can be performed to help RSVP interpret explicit route objects, use the **debug mpls traffic-eng link-management routing** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug mpls traffic-eng link-management routing [**detail**]

no debug mpls traffic-eng link-management routing [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword was added.

Examples

In the following example, detailed debugging information is printed about traffic engineering link management routing resolutions that can be performed to help RSVP interpret explicit route objects:

```
debug mpls traffic-eng link-management routing detail

TE-LM-ROUTING:route options to 10.42.0.10:building list (w/ nhop matching)
TE-LM-ROUTING:route options to 10.42.0.10:adding {AT0/0.2, 10.42.0.10}
TE-LM-ROUTING:route options to 10.42.0.10:completed list has 1 links
```

Related Commands

Command	Description
debug ip rsvp	Prints information about RSVP signalling events.

debug mpls traffic-eng load-balancing

To print information about unequal cost load balancing over traffic engineering tunnels, use the **debug mpls traffic-eng load-balancing** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng load-balancing

no debug mpls traffic-eng load-balancing

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about unequal cost load balancing over traffic engineering tunnels:

```
debug mpls traffic-eng load-balancing

TE-Load:10.210.0.0/16, 2 routes, loadbalancing based on MPLS TE bandwidth
TE-Load:10.200.0.0/16, 2 routes, loadbalancing based on MPLS TE bandwidth
```

debug mpls traffic-eng path

To print information about traffic engineering path calculation, use the **debug mpls traffic-eng path** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng path {*num* | **lookup** | **spf** | **verify**}

no debug mpls traffic-eng path {*num* | **lookup** | **spf** | **verify**}

Syntax Description

num	Prints path calculation information only for the local tunneling interface with unit number *num*.
lookup	Prints information for path lookups.
spf	Prints information for shortest path first (SPF) calculations.
verify	Prints information for path verifications.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about the calculation of the traffic engineering path:

```
debug mpls traffic-eng path lookup

TE-PCALC:Tunnel1000 Path Setup to 10.110.0.10:FULL_PATH
TE-PCALC:bw 0, min_bw 0, metric:0
TE-PCALC:setup_pri 0, hold_pri 0
TE-PCALC:affinity_bits 0x0, affinity_mask 0xFFFF
TE-PCALC_PATH:create_path_hoplist:ip addr 10.42.0.6 unknown.
```

debug mpls traffic-eng topology change

To print information about traffic engineering topology change events, use the **debug mpls traffic-eng topology change** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng topology change

no debug mpls traffic-eng topology change

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about traffic engineering topology change events:

```
debug mpls traffic-eng topology change

TE-PCALC_LSA:NODE_CHANGE_UPDATE isis level-1
        link flags:LINK_CHANGE_BW
        system_id:0001.0000.0001.00, my_ip_address:10.42.0.6
        nbr_system_id:0001.0000.0002.00, nbr_ip_address 10.42.0.10
```

debug mpls traffic-eng topology lsa

To print information about traffic engineering topology link state advertisement (LSA) events, use the **debug mpls traffic-eng topology lsa** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng topology lsa

no debug mpls traffic-eng topology lsa

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information is printed about traffic engineering topology LSA events:

```
debug mpls traffic-eng topology lsa

TE-PCALC_LSA:node_lsa_add:Received a LSA:flags 0x1 !

IGP Id:0001.0000.0001.00, MPLS TE Id:10.106.0.6 is VALID has 2 links (frag_id 0)
      link[0 ]:Nbr IGP Id:0001.0000.0001.02
          frag_id 0, Intf Address:0.0.0.0
          admin_weight:10, attribute_flags:0x0

      link[1 ]:Nbr IGP Id:0001.0000.0002.00
          frag_id 0, Intf Address:10.42.0.6, Nbr Intf Address:10.42.0.10
          admin_weight:100, attribute_flags:0x0
TE-PCALC_LSA:(isis level-1):Received lsa:

IGP Id:0001.0000.0001.00, MPLS TE Id:10.106.0.6 Router Node  id 8
      link[0 ]:Nbr IGP Id:0001.0000.0002.00, nbr_node_id:9, gen:114
          frag_id 0, Intf Address:10.42.0.6, Nbr Intf Address:10.42.0.10
          admin_weight:100, attribute_flags:0x0
          physical_bw:155520 (kbps), max_reservable_bw:5000 (kbps)
               allocated_bw   reservable_bw      allocated_bw   reservable_bw
               ------------   -------------      ------------   -------------
        bw[0]:0             5000        bw[1]:3000          2000
        bw[2]:0             2000        bw[3]:0             2000
        bw[4]:0             2000        bw[5]:0             2000
        bw[6]:0             2000        bw[7]:0             2000
```

debug mpls traffic-eng tunnels errors

To print information about errors encountered during any traffic engineering tunnel management procedure, use the **debug mpls traffic-eng tunnels errors** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels errors [**detail**]

no debug mpls traffic-eng tunnels errors [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, detailed debugging information is printed about errors encountered during a traffic engineering tunnel management procedure:

```
debug mpls traffic-eng tunnels errors

00:04:14: LSP-TUNNEL-SIG: Tunnel10012[1]: path verification failed (unprotected) [Can't
use link 10.12.4.4 on node 10.0.0.4]
```

debug mpls traffic-eng tunnels events

To print information about traffic engineering tunnel management system events, use the **debug mpls traffic-eng tunnels events** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels events [**detail**]

no debug mpls traffic-eng tunnels events [**detail**]

Syntax Description

detail	(Optional) Prints detailed debugging information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword was added.

Examples

In the following example, detailed debugging information is printed about traffic engineering tunnel management system events:

```
debug mpls traffic-eng tunnels events detail

LSP-TUNNEL:received event:interface admin. down [Ethernet4/0/1]
LSP-TUNNEL:posting action(s) to all-tunnels:
              check static LSPs
LSP-TUNNEL:scheduling pending actions on all-tunnels
LSP-TUNNEL:applying actions to all-tunnels, as follows:
              check static LSPs
```

debug mpls traffic-eng tunnels labels

To print information about MPLS label management for traffic engineering tunnels, use the **debug mpls traffic-eng tunnels labels** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels labels [**detail**] [*aclnum*]

no debug mpls traffic-eng tunnels labels [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information only about traffic engineering tunnels that match the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword and the *aclnum* argument were added.

Examples

In the following example, detailed debugging information is printed about MPLS label management for traffic engineering tunnels:

```
debug mpls traffic-eng tunnels labels detail

LSP-TUNNEL-LABELS:tunnel 10.106.0.6 1 [2]:fabric PROGRAM request
LSP-TUNNEL-LABELS:tunnel 10.106.0.6 1 [2]:programming label 16 on output interface
ATM0/0.2
LSP-TUNNEL-LABELS:descriptor 71FA64:continuing "Program" request
LSP-TUNNEL-LABELS:descriptor 71FA64:set "Interface Point Out State" to, allocated
LSP-TUNNEL-LABELS:# of resource points held for "default" interfaces:2
LSP-TUNNEL-LABELS:descriptor 71FA64:set "Fabric State" to, enabled
LSP-TUNNEL-LABELS:descriptor 71FA64:set "Fabric Kind" to, default (LFIB)
LSP-TUNNEL-LABELS:descriptor 71FA64:set "Fabric State" to, set
LSP-TUNNEL-LABELS:tunnel 10.106.0.6 1 [2]:fabric PROGRAM reply
```

To restrict output to information about a single tunnel, you can configure an access list and supply it to the **debug** command. Configure the access list as follows:

```
Router(config-ext-nacl)# permit udp host scr_address host dst_address eq tun intfc
```

For example, if tunnel 10012 has destination 10.0.0.11 and source 10.0.0.4, as determined by **show mpls traffic-eng tunnels** command, the following access list could be configured and added to the **debug** command:

```
Router(config-ext-nacl)# permit udp host 10.0.0.4 10.0.0.11 eq 10012
```

debug mpls traffic-eng tunnels reoptimize

To print information about traffic engineering tunnel re-optimizations, use the **debug mpls traffic-eng tunnels reoptimize** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels reoptimize [**detail**] [*aclnum*]

no debug mpls traffic-eng tunnels reoptimize [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information about only those traffic engineering tunnel reoptimizations that match the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword and the *aclnum* argument were added.

Examples

In the following example, detailed debugging information is printed about traffic engineering tunnel re-optimizations that match access list number 101:

```
debug mpls traffic-eng tunnels reoptimize detail 101

LSP-TUNNEL-REOPT:Tunnel1 curr option 2 (0x6175CF8C), activate new option 2
LSP-TUNNEL-REOPT:Tunnel1 new path:option 2 [10002], weight 20
LSP-TUNNEL-REOPT:Tunnel1 old path:option 2 [2], weight 110
LSP-TUNNEL-REOPT:Tunnel1 [10002] set as reopt
LSP-TUNNEL-REOPT:Tunnel1 path option 2 [10002] installing as current
LSP-TUNNEL-REOPT:Tunnel1 [2] removed as current
LSP-TUNNEL-REOPT:Tunnel1 [2] set to delayed clean
LSP-TUNNEL-REOPT:Tunnel1 [10002] removed as reopt
LSP-TUNNEL-REOPT:Tunnel1 [10002] set to current
```

debug mpls traffic-eng tunnels signalling

To print information about traffic engineering tunnel signalling operations, use the **debug mpls traffic-eng tunnels signalling** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels signalling [**detail**] [*aclnum*]

no debug mpls traffic-eng tunnels signalling [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information about only those traffic engineering tunnel signalling operations that match the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword and the aclnum argument were added.

Examples

In the following example, detailed debugging information is printed about traffic engineering tunnel signalling operations that match access list number 101:

```
debug mpls traffic-eng tunnels signalling detail 101

LSP-TUNNEL-SIG:tunnel Tunnel1 [2]:RSVP head-end open
LSP-TUNNEL-SIG:tunnel Tunnel1 [2]:received Path NHOP CHANGE
LSP-TUNNEL-SIG:Tunnel1 [2]:first hop change:0.0.0.0 --> 10.1.0.10
LSP-TUNNEL-SIG:received ADD RESV request for tunnel 10.106.0.6 1 [2]
LSP-TUNNEL-SIG:tunnel 10.106.0.6 1 [2]:path next hop is 10.1.0.10 (Et4/0/1)
LSP-TUNNEL-SIG:Tunnel1 [2] notified of new label information
LSP-TUNNEL-SIG:sending ADD RESV reply for tunnel 10.106.0.6 1 [2]
```

debug mpls traffic-eng tunnels state

To print information about state maintenance for traffic engineering tunnels, use the **debug mpls traffic-eng tunnels state** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels state [**detail**] [*aclnum*]

no debug mpls traffic-eng tunnels state [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information about state maintenance for traffic engineering tunnels that match the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, detailed debugging information is printed about state maintenance for traffic engineering tunnels that match access list number 99:

```
debug mpls traffic-eng tunnels state detail 99

LSP-TUNNEL:tunnel 10.106.0.6 1 [2]: "Connected" -> "Disconnected"
LSP-TUNNEL:Tunnel1 received event:LSP has gone down
LSP-TUNNEL:tunnel 10.106.0.6 1 [2]: "Disconnected" -> "Dead"
LSP-TUNNEL-SIG:Tunnel1:changing state from up to down
LSP-TUNNEL:tunnel 10.106.0.6 1 [2]: "Dead" -> "Connected"
```

debug mpls traffic-eng tunnels timers

To print information about traffic engineering tunnel timer management, use the **debug mpls traffic-eng tunnels timers** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug mpls traffic-eng tunnels timers [**detail**] [*aclnum*]

no debug mpls traffic-eng tunnels timers [**detail**] [*aclnum*]

Syntax Description

detail	(Optional) Prints detailed debugging information.
aclnum	(Optional) Uses the specified access list to filter the debugging information. Prints information about traffic engineering tunnel timer management that matches the access list.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.05(S)	This command was introduced.
12.1(3)T	The **detail** keyword and the *aclnum* argument were added.

Examples

In the following example, detailed debugging information is printed about traffic engineering tunnel timer management:

```
debug mpls traffic-eng tunnels timers detail

LSP-TUNNEL-TIMER:timer fired for Action Scheduler
LSP-TUNNEL-TIMER:timer fired for Tunnel Head Checkup
```

debug mpoa client

To display MPC debug information, use the **debug mpoa client** privileged EXEC command. The **no** form of this command disables debugging output.

debug mpoa client {**all** | **data** | **egress** | **general** | **ingress** | **keep-alives** | **platform-specific**} [**name** *mpc-name*]

no debug mpoa client {**all** | **data** | **egress** | **general** | **ingress** | **keep-alives** | **platform-specific**} [**name** *mpc-name*]

Syntax Description

all	Displays debugging information for all MPC activity.
data	Displays debugging information for data plane activity only. This option applies only to routers.
egress	Displays debugging information for egress functionality only.
general	Displays general debugging information only.
ingress	Displays debugging information for ingress functionality only.
keep-alives	Displays debugging information for keep-alive activity only.
platform-specific	Displays debugging information for specific platforms only. This option applies only to the Catalyst 5000 series ATM module.
name *mpc-name*	Specifies the name of the MPC with the specified name.

Defaults

The default is debugging turned on for all MPCs.

Command History

Release	Modification
11.3	This command was introduced.

Examples

The following shows how to turn on debugging for the MPC ip_mpc:

```
ATM# debug mpoa client all name ip_mpc
```

Related Commands

Command	Description
debug mpoa server	Displays information about the MPOA server.

debug mpoa server

To display information about the MPOA server, use the **debug mpoa server** privileged EXEC command. The **no** form of this command disables debugging output.

debug mpoa server [**name** *mps-name*]

no debug mpoa server [**name** *mps-name*]

Syntax Description

name *mps-name*	(Optional) Specifies the name of a MPOA server.

Command History

Release	Modification
11.3	This command was introduced.

Usage Guidelines

The **debug mpo server** command optionally limits the output only to the specified MPS.

Examples

The following turns on debugging only for the MPS named ip_mps:

```
Router# debug mpoa server name ip_mps
```

Related Commands

Command	Description
debug modem traffic	Displays MPC debug information.

debug mspi receive

To display debug messages for mail Service Provider Interface (SPI) receive, use the **debug mspi receive** EXEC command. To disable the debug messages, use the **no** form of this command.

debug mspi receive

no debug mspi receive

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug mspi receive** command.

```
Router# debug mspi receive

Jan  1 05:09:33.890: mspi_tel_num_trans: from: Radhika,
ph#in: fax=5271714 ph#dial: 5271714
Jan  1 05:09:33.890:  incoming destPat(5271714), matched(7), tag(22)
Jan  1 05:09:33.890:  out destPat(5......), tag(20), dgt strip enabled
Jan  1 05:09:33.890: mspi_off_new_rcpt: envlp_to [fax=5271714@rpadmana.cisco.com], 30
Jan  1 05:09:33.890:  tel_numb_dial: 5271714, subaddr:[], cover page
Jan  1 05:09:39.122: mspi_offramp_rfc822_header: msgType=0
Jan  1 05:09:39.122:  envlp_from: [Radhika], 8
Jan  1 05:09:39.122: mspi_off_put_buff: ignore mime type=1, st=CONNECTING, len=0
Jan  1 05:09:39.122: moff_save_buffer: cid=0x1F, mime=9, len=4
Jan  1 05:09:39.122:  offramp disabled receiving!
Dec 31 21:09:44.078: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 5271714
Jan  1 05:09:52.154: mspi_bridge: cid=0x1F, dst cid=0x22, data dir=OFFRAMP, conf dir=DEST
Jan  1 05:09:52.154: mspi_offramp_send_buffer: cid=0x1F, mime=9
Jan  1 05:09:52.154:  buffer with only CR/LF - set buff_len=0
Jan  1 05:09:52.154: mspi_offramp_send_buffer: cid=0x1F, mime=9 rx BUFF_END_OF_PART,
offramp rcpt enabled
Jan  1 05:09:54.126: mspi_offramp_send_buffer: cid=0x1F, mime=11
Jan  1 05:09:54.134: mspi_offramp_send_buffer: cid=0x1F, mime=11
```

Related Commands

Command	Description
debug mspi send	Displays debug messages for mail SPI send.

debug mspi send

To display debug messages for mail Service Provider Interface (SPI) send, use the **debug mspi send** EXEC command. To disable the debug messages, use the **no** form of this command.

debug mspi send

no debug mspi send

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI	This command was introduced on the Cisco AS5300 access server.

Examples

The following example displays output from the **debug mspi send** command.

```
Router# debug mspi send

*Oct 16 08:40:27.515: mspi_bridge: cid=0x21, dst cid=0x26, data dir=OFFRAMP, conf dir=DEST
*Oct 16 08:40:29.143: mspi_setup_req: for cid=0x27
*Oct 16 08:40:29.147:   envelope_from=5??????@fax.cisco.com
*Oct 16 08:40:29.147:   envelope_to=ilyau@cisco.com
*Oct 16 08:40:30.147: mspi_chk_connect: cid=0x27, cnt=0,
*Oct 16 08:40:30.147:  SMTP connected to the server !
*Oct 16 08:40:30.147: mspi_bridge: cid=0x27, dst cid=0x28, data dir=ONRAMP, conf dir=SRC
*Oct 16 08:40:38.995: mspi_xmit: cid=0x27, st=CONFERENCED, src_cid=0x28, buf cnt=0
```

Related Commands

Command	Description
debug mspi receive	Displays debug messages for mail SPI receive.

debug mta receive all

To show output relating to the activity on the SMTP server, use the **debug mta receive all** EXEC command. Use the **no** form of this command to disable debugging output.

debug mta receive all

no debug mta receive all

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

The following example shows the messages exchanged (for example, the handshake) between the e-mail server and the off-ramp gateway.

```
Router# debug mta receive all

Jan  1 05:07:41.314: esmtp_server_work: calling helo
Jan  1 05:07:43.354: esmtp_server_work: calling mail
Jan  1 05:07:45.386: esmtp_server_work: calling rcpt
Jan  1 05:07:47.426: esmtp_server_work: calling data
Jan  1 05:07:49.514: (S)R: 'Content-Type: multipart/mixed;
boundary="------------11F7CD9D2EB3E8B8D5627C62"'
Jan  1 05:07:49.514: (S)R: ''
Jan  1 05:07:49.514: esmtp_server_engine_new_part:
Jan  1 05:07:49.514: (S)R: 'Content-Type: text/plain; charset=us-ascii'
Jan  1 05:07:49.514: (S)R: 'Content-Transfer-Encoding: 7bit'
Jan  1 05:07:49.514: (S)R: ''
Jan  1 05:07:49.514: esmtp_server_engine_new_part:
Jan  1 05:07:49.514: esmtp_server_work: freeing temp header
Jan  1 05:07:49.514: (S)R: 'Content-Type: image/tiff; name="DevTest.8.1610.tif"'
Jan  1 05:07:49.514: (S)R: 'Content-Transfer-Encoding: base64'
Jan  1 05:07:49.514: (S)R: 'Content-Disposition: inline; filename="DevTest.8.1610.tif"'
Jan  1 05:07:49.514: (S)R: ''
Jan  1 05:07:49.514: esmtp_server_engine_update_recipient_status: status=6
Jan  1 05:07:49.514: esmtp_server_engine_new_part:
Jan  1 05:07:49.518: esmtp_server_work: freeing temp header
Jan  1 05:08:03.014: esmtp_server_engine_update_recipient_status: status=7
Jan  1 05:08:04.822: esmtp_server_engine_update_recipient_status: status=6
Jan  1 05:08:33.042: esmtp_server_engine_update_recipient_status: status=7
Jan  1 05:08:34.906: esmtp_server_engine_getline: Unexpected end of file on socket 1
Jan  1 05:08:34.906: esmtp_server_work: error occured with ctx=0x61FFF710, socket=1
```

Related Commands

Command	Description
debug mta send all	Displays output for all of the on-ramp client connections.

debug mta send all

To display output for all of the on-ramp client connections, use the **debug mta send all** EXEC command. Use the **no** form of this command to disable debugging output.

debug mta send all

no debug mta send all

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

The following example shows the messages exchanged (for example, the handshake) between the e-mail server and the on-ramp gateway.

```
Router# debug mta send all

*Oct 16 09:04:13.055: esmtp_client_engine_open: from=5??????@fax.cisco.com,
to=ilyau@cisco.com
*Oct 16 09:04:13.055: esmtp_client_engine_add_headers: from_comment=
*Oct 16 09:04:13.111: esmtp_client_work: socket 0 attempting to connect to IP address
171.71.154.56
*Oct 16 09:04:13.111: esmtp_client_work: socket 0 readable for first time
*Oct 16 09:04:13.135: esmtp_client_work: socket 0 readable for first time
*Oct 16 09:04:13.135: (C)R: 220 quisp.cisco.com ESMTP Sendmail 8.8.4-Cisco.1/8.6.5 ready
at Wed, 27 Sep 2000 11:45:46 -0700 (PDT)
*Oct 16 09:04:13.135: (C)S: EHLO mmoip-c.cisco.com
*Oct 16 09:04:13.183: (C)R: 250-quisp.cisco.com Hello [172.22.95.16], pleased to meet you
*Oct 16 09:04:13.183: (C)R: 250-EXPN
*Oct 16 09:04:13.183: (C)R: 250-VERB
```

Related Commands

Command	Description
debug mta receive all	Displays output for all of the off-ramp client connections.
debug mta send rcpt-to	Displays output for a specific on-ramp SMTP client connection during an e-mail transmission.

debug mta send rcpt-to

To display output for a specific on-ramp SMTP client connection during an e-mail transmission, use the **debug mta send rcpt-to** EXEC command. Use the **no** form of this command to disable debugging output.

debug mta send rcpt-to *string*

[no] debug mta send rcpt-to *string*

Syntax Description

string	Specifies the e-mail address.

Defaults

Disabled

Command History

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

Examples

The following example shows debugging information displayed when the **debug mmoip send email** command has been enabled and the SMTP client is sending an e-mail message.

```
Router# debug mta send all
All email send debugging is on
Router# debug mmoip send email ilyau@company.com
Router# socket 0 attempting to connect to IP address 172.69.95.82
socket 0 readable for first time - let's try to read it
R:220 quisp.cisco.com ESMTP Sendmail 8.8.4-Cisco.1/8.6.5 ready at Tue, 6
Apr 1999 13:35:39 -0700 (PDT)
S:EHLO mmoip-c.cisco.com
R:250-quisp.cisco.com Hello [172.22.95.16], pleased to meet you
R:250-EXPN
R:250-VERB
R:250-8BITMIME
R:250-SIZE
R:250-DSN
R:250-ETRN
R:250-XUSR
R:250 HELP
S:MAIL FROM:<testing@> RET=HDRS
R:250 <testing@>... Sender ok
S:RCPT TO:<ilyau@cisco.com> NOTIFY=SUCCESS ORCPT=rfc822;testing@
R:250 <ilyau@cisco.com>... Recipient ok
R:354 Enter mail, end with "." on a line by itself
S:Received:(Cisco Powered Fax System) by mmoip-c.cisco.com for
<ilyau@cisco.com> (with Cisco NetWorks); Fri, 17 Oct 1997 14:54:27 +0800
S:To: <ilyau@cisco.com>
S:Message-ID:<000F1997145427146@mmoip-c.cisco.com>
S:Date:Fri, 17 Oct 1997 14:54:27 +0800
S:Subject:mmoip-c subject here
S:X-Mailer:IOS (tm) 5300 Software (C5300-IS-M)
S:MIME-Version:1.0
S:Content-Type:multipart/mixed;
S: boundary="yradnuoB=_000E1997145426826.mmoip-ccisco.com"
```

```
S:From:"Test User" <testing@>
S:--yradnuoB=_000E1997145426826.mmoip-ccisco.com
S:Content-ID:<00101997145427150@mmoip-c.cisco.com>
S:--yradnuoB=_000E1997145426826.mmoip-ccisco.com--
Sending terminating dot ...(socket=0)
S:.
R:250 NAA09092 Message accepted for delivery
S:QUIT
R:221 quisp.cisco.com closing connection
Freeing SMTP ctx at 0x6121D454
returned from work_routine, context freed
```

Related Commands

Command	Description
debug mta send all	Displays output for all of the on-ramp client connections.

debug ncia circuit

To display circuit-related information between the native client interface architecture (NCIA) server and client, use the **debug ncia circuit** privileged EXEC command. The **no** form of this command disables debugging output.

debug ncia circuit [**error** | **event** | **flow-control** | **state**]

no debug ncia circuit [**error** | **event** | **flow-control** | **state**]

Syntax Description

error	(Optional) Displays the error situation for each circuit.
event	(Optional) Displays the packets received and sent for each circuit.
flow-control	(Optional) Displays the flow control information for each circuit.
state	(Optional) Displays the state changes for each circuit.

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing SNA applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

You cannot enable debugging output for a particular client or particular circuit.

Caution Do not enable the **debug ncia circuit** command during normal operation because this command generates a substantial amount of output messages and could slow down the router.

Examples

The following is sample output from the **debug ncia circuit error** command. In this example, the possible errors are displayed. The first error message indicates that the router is out of memory. The second message indicates that the router has an invalid circuit control block. The third message indicates that the router is out of memory. The remaining messages identify errors related to the finite state machine.

```
Router# debug ncia circuit error

NCIA: ncia_circuit_create memory allocation fail
NCIA: ncia_send_ndlc: invalid circuit control block
NCIA: send_ndlc: fail to get buffer for ndlc primitive xxx
NCIA: ncia circuit fsm: Invalid input
NCIA: ncia circuit fsm: Illegal state
NCIA: ncia circuit fsm: Illegal input
NCIA: ncia circuit fsm: Unexpected input
NCIA: ncia circuit fsm: Unknown error rtn code
```

The following is sample output from the **debug ncia circuit event** command. In this example, a session start-up sequence is displayed.

```
Router# debug ncia circuit event

NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_START_DL, Len: 24, tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 0, lfs 16, ws 1
NCIA: create circuit: saddr 4000.1060.1000, ssap 4, daddr 4000.3000.0003, dsap 8 sid:
          8B09A8
NCIA: send NDLC_DL_STARTED to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_DL_STARTED, Len: 2,4 tmac: 4000.1060.1000,
```

```
          tsap: 4, csap 8, oid: 8A91E8, tid 8B09A8, lfs 16, ws 1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8B09A8, FC 0x81
NCIA: send NDLC_XID_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 18, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_CONTACT_STN to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_CONTACT_STN, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_STN_CONTACTED, Len: 12, sid: 8B09A8, FC 0xC1
NCIA: send NDLC_INFO_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_INFO_FRAME, Len: 30, sid: 8A91E8, FC 0xC1
```

Table 134 describes the significant fields in the output.

Table 134 debug ncia circuit event Field Descriptions

Field	Description
IN	Incoming message from client.
OUT	Outgoing message to client.
Ver_Id	NDLC version ID.
MsgType	NDLC message type.
Len	NDLC message length.
tmac	Target MAC.
tsap	Target SAP.
csap	Client SAP.
oid	Origin ID.
tid	Target ID.
lfs	Largest frame size flag.
ws	Window size.
saddr	Source MAC address.
ssap	Source SAP.
daddr	Destination MAC address.
dsap	Destination SAP.
sid	Session ID.
FC	Flow control flag.

In the following messages, an NDLC_START_DL messages is received from a client. to start a data-link session:

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_START_DL, Len: 24, tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 0, lfs 16, ws 1
NCIA: create circuit: saddr 4000.1060.1000, ssap 4, daddr 4000.3000.0003, dsap 8 sid:
          8B09A8
```

The next two messages indicate that an NDLC_DL_STARTED message is sent to a client. The server informs the client that a data-the link session is started.

```
NCIA: send NDLC_DL_STARTED to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_DL_STARTED, Len: 2,4 tmac: 4000.1060.1000,
          tsap: 4, csap 8, oid: 8A91E8, tid 8B09A8, lfs 16, ws 1
```

In the following two messages, an NDLC_XID_FRAME message is received from a client, and the client starts an XID exchange:

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8B09A8, FC 0x81
NCIA: send NDLC_XID_FRAME to client 10.2.20.3 for ckt: 8B09A8
```

In the following two messages, an NDLC_XID_FRAME message is sent from a client, and an DLC_XID_FRAME message is received from a client:

```
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 12, sid: 8A91E8, FC 0xC1
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_XID_FRAME, Len: 18, sid: 8B09A8, FC 0xC1
```

The next two messages show that an NDLC_CONTACT_STN message is sent to a client:

```
NCIA: send NDLC_CONTACT_STN to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_CONTACT_STN, Len: 12, sid: 8A91E8, FC 0xC1
```

In the following message, an NDLC_STN_CONTACTED message is received from a client. The client informs the server that the station has been contacted.

```
NCIA(IN): Ver_Id: 0x81, MsgType: NDLC_STN_CONTACTED, Len: 12, sid: 8B09A8, FC 0xC1
```

In the last two messages, an NDLC_INFO_FRAME is sent to a client, and the server sends data to the client:

```
NCIA: send NDLC_INFO_FRAME to client 10.2.20.3 for ckt: 8B09A8
NCIA(OUT): Ver_Id: 0x81, MsgType: NDLC_INFO_FRAME, Len: 30, sid: 8A91E8, FC 0xC1
```

The following is sample output from the **debug ncia circuit flow-control** command. In this example, the flow control in a session startup sequence is displayed:

```
Router# debug ncia circuit flow-control

NCIA: no flow control in NDLC_DL_STARTED frame
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0x81, IW 1 GP 2 CW 2, Client IW 1 GP 0 CW 1
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 2 CW 2, Client IW 1 GP 2 CW 2
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0xC1, IW 1 GP 5 CW 3, Client IW 1 GP 2 CW 2
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 5 CW 3, Client IW 1 GP 5 CW 3
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
NCIA: ncia_flow_control_in FC 0xC1, IW 1 GP 9 CW 4, Client IW 1 GP 5 CW 3
NCIA: grant client more packet by sending Repeat Window Op
NCIA: ncia_flow_control_out FC: 0xC1, IW 1 GP 8 CW 4, Client IW 1 GP 9 CW 4
NCIA: reduce ClientGrantPacket by 1 (Granted: 8)
NCIA: receive FCA for circuit 8ADE00
NCIA: receive Increment Window Op for circuit 8ADE00
```

Table 135 describes the significant fields shown in the display.

Table 135 debug ncia circuit flow-control Field Descriptions

Field	Description
IW	Initial window size.
GP	Granted packet number.
CW	Current window size.

The following is sample output from the **debug ncia circuit state** command. In this example, a session startup sequence is displayed:

```
Router# debug ncia circuit state

NCIA: pre-server fsm: event CONN_OPENED
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - STDL state: CLSOED
NCIA: ncia server fsm action 32
NCIA: circuit state: CLOSED -> START_DL_RCVD
NCIA: server event: DLU - TestStn.Rsp state: START_DL_RCVD
NCIA: ncia server fsm action 17
NCIA: circuit state: START_DL_RCVD -> DL_STARTED_SND
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - XID state: DL_STARTED_SND
NCIA: ncia server fsm action 33
NCIA: circuit state: DL_STARTED_SND -> DL_STARTED_SND
NCIA: server event: DLU - ReqOpnStn.Req state: DL_STARTED_SND
NCIA: ncia server fsm action 33
NCIA: circuit state: DL_STARTED_SND -> OPENED
NCIA: server event: DLU - Id.Rsp state: OPENED
NCIA: ncia server fsm action 11
NCIA: circuit state: OPENED -> OPENED
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - XID state: OPENED
NCIA: ncia server fsm action 33
NCIA: circuit state: OPENED -> OPENED
NCIA: server event: DLU - Connect.Req state: OPENED
NCIA: ncia server fsm action 6
NCIA: circuit state: OPENED -> CONNECT_PENDING
NCIA: pre-server fsm: event NDLC_PRIMITIVES
NCIA: server event: WAN - CONR state: CONNECT_PENDING
NCIA: ncia server fsm action 33 --> CLS_CONNECT_CNF sets NciaClsBusy
NCIA: circuit state: CONNECT_PENDING -> CONNECTED
NCIA: server event: DLU - Flow.Req (START) state: CONNECTED
NCIA: ncia server fsm action 25 --> unset NciaClsBusy
NCIA: circuit state: CONNECTED -> CONNECTED
NCIA: server event: DLU - Data.Rsp state: CONNECTED
NCIA: ncia server fsm action 8
NCIA: circuit state: CONNECTED -> CONNECTED
```

Table 136 describes the significant fields shown in the display.

Table 136 debug ncia circuit state Field Descriptions

Field	Description
WAN	Event from WAN (client).
DLU	Event from upstream module—dependent logical unit (DLU).
ADMIN	Administrative event.
TIMER	Timer event.

Related Commands

Command	Description
debug dmsp fax-to-doc	Enables debugging of DLSw+.
debug ncia client	Displays debug information for all NCIA client processing that occurs in the router.
debug ncia server	Displays debug information for the NCIA server and its upstream software modules.

debug ncia client

To display debug information for all native client interface architecture (NCIA) client processing that occurs in the router, use the **debug ncia client** privileged EXEC command. The **no** form of this command disables debugging output.

debug ncia client [*ip-address* | **error** [*ip-address*] | **event** [*ip-address*] | **message** [*ip-address*]]

no debug ncia client [*ip-address* | **error** [*ip-address*] | **event** [*ip-address*] | **message** [*ip-address*]]

Syntax Description

ip-address	(Optional) The remote client IP address.
error	(Optional) Triggers the recording of messages only when errors occur. The current state and event of an NCIA client are normally included in the message. If you do not specify an IP address, the error messages are logged for all active clients.
event	(Optional) Triggers the recording of messages that describe the current state and event—and sometimes the action that just completed—for the NCIA client. If you do not specify an IP address, the messages are logged for all active clients.
message	(Optional) Triggers the recording of messages that contain up to the first 32 bytes of data in a TCP packet sent to or received from an NCIA client. If you do not specify an IP address, the messages are logged for all active clients.

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing SNA applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

Use the **debug ncia client error** command to see only certain error conditions that occur.

Use the **debug ncia client event** command to determine the sequences of activities that occur while a NCIA client is in different processing states.

Use the **debug ncia client message** command to see only the first 32 bytes of data in a TCP packet sent to or received from an NCIA client.

The **debug ncia client** command can be used in conjunction with the **debug ncia server** and **debug ncia circuit** commands to get a complete picture of NCIA activity.

Examples

The following is sample output from the **debug ncia circuit** command. Following the example is a description of each sample output message.

```
Router# debug ncia client

NCIA: Passive open 10.2.20.123(1088) -> 1973
NCIA: index for client hash queue is 27
NCIA: number of element in client hash queue 27 is 1
NCIA: event PASSIVE_OPEN, state NCIA_CLOSED for client 10.2.20.123
NCIA: Rcvd msg type NDLC_CAP_XCHG in tcp packet for client 10.2.20.123
NCIA: First 17 byte of data rcvd: 8112001100000000000000400050104080C
NCIA: Sent msg type NDLC_CAP_XCHG in tcp packet to client 10.2.20.123
NCIA: First 17 byte of data sent: 8112001110000000100004000501040800C
NCIA: event CAP_CMD_RCVD, state NCIA_CAP_WAIT, for client 10.2.20.123, cap xchg cmd sent
```

```
NCIA: Rcvd msg type NDLC_CAP_XCHG in tcp packet for client 10.2.20.123
NCIA: First 17 byte of data rcvd: 81120011100000001000000050104080C
NCIA: event CAP_RSP_RCVD, state NCIA_CAP_NEG for client 10.2.20.123

NCIA: Rcvd msg type NDLC_PEER_TEST_REQ in tcp packet for client 10.2.20.123
NCIA: First 4 byte of data rcvd: 811D0004
NCIA: event KEEPALIVE_RCVD, state NCIA_OPENED for client 10.2.20.123
NCIA: Sent msg type NDLC_PEER_TEST_RSP in tcp packet to client 10.2.20.123
NCIA: First 4 byte of data sent: 811E0004IA

NCIA: event TIME_OUT, state NCIA_OPENED, for client 10.2.20.123, keepalive_count = 0
NCIA: Sent msg type NDLC_PEER_TEST_REQ, in tcp packet to client 10.2.20.123
NCIA: First 4 byte of data sent: 811D0004
NCIA: Rcvd msg type NDLC_PEER_TEST_RSP in tcp packet for client 10.2.20.123
NCIA: First 4 byte of data rcvd: 811E0004
NCIA: event KEEPALIVE_RSP_RCVD, state NCIA_OPENED for client 10.2.20.123

NCIA: Error, event PASIVE_OPEN, state NCIA_OPENED, for client 10.2.20.123, should not have
occurred.
NCIA: Error, active_open for pre_client_fsm while client 10.2.20.123 is active or not
configured, registered.
```

Messages in lines 1 through 12 show the events that occur when a client connects to the router (the NCIA server). These messages show a passive_open process.

Messages in lines 13 to 17 show the events that occur when a TIME_OUT event is detected by a client PC workstation. The workstation sends an NDLC_PEER_TEST_REQ message to the NCIA server, and the router responds with an NDLC_PEER_TEST_RSP message.

Messages in lines 18 to 23 show the events that occur when a TIME_OUT event is detected by the router (the NCIA server). The router sends an NDLC_PEER_TEST_REQ message to the client PC workstation, and the PC responds with an NDLC_PEER_TEST_RSP message.

When you use the **debug ncia client message** command, the messages shown on lines 6, 8, 11, 14, 17, 20, and 22 are output in addition to other messages not shown in this example.

When you use the **debug ncia client error** command, the messages shown on lines 24 and 25 are output in addition to other messages not shown in this example.

Related Commands

Command	Description
debug ncia circuit	Displays debug information for all NCIA client processing that occurs in the router.
debug ncia server	Displays debug information for the NCIA server and its upstream software modules.

debug ncia server

To display debug information for the native client interface architecture (NCIA) server and its upstream software modules, use the **debug ncia server** privileged EXEC command. The **no** form of this command disables debugging output.

debug ncia server

no debug ncia server

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

NCIA is an architecture developed by Cisco for accessing SNA applications. This architecture allows native SNA interfaces on hosts and clients to access TCP/IP backbones.

The **debug ncia server** command displays all Cisco Link Services (CLS) messages between the NCIA server and its upstream modules, such as data-link switching (DLSw) and downstream physical units (DSPUs). Use this command when a problem exists between the NCIA server and other software modules within the router.

You cannot enable debugging output for a particular client or particular circuit.

Examples

The following is sample output from the **debug ncia server** command. In this example, a session startup sequence is displayed. Following the example is a description of each group of sample output messages.

```
Router# debug ncia server

NCIA: send CLS_TEST_STN_IND to DLU
NCIA: Receive TestStn.Rsp
NCIA: send CLS_ID_STN_IND to DLU
NCIA: Receive ReqOpnStn.Req
NCIA: send CLS_REQ_OPNSTN_CNF to DLU
NCIA: Receive Id.Rsp
NCIA: send CLS_ID_IND to DLU
NCIA: Receive Connect.Req
NCIA: send CLS_CONNECT_CNF to DLU
NCIA: Receive Flow.Req
NCIA: Receive Data.Req
NCIA: send CLS_DATA_IND to DLU
NCIA: send CLS_DISC_IND to DLU
NCIA: Receive Disconnect.Rsp
```

In the following messages, the client is sending a test message to the host and the test message is received by the host:

```
NCIA: send CLS_TEST_STN_IND to DLU
NCIA: Receive TestStn.Rsp
```

In the next message, the server is sending an XID message to the host:

```
NCIA: send CLS_ID_STN_IND to DLU
```

In the next two messages, the host opens the station and the server responds:

```
NCIA: Receive ReqOpnStn.Req
NCIA: send CLS_REQ_OPNSTN_CNF to DLU
```

In the following two messages, the client is performing an XID exchange with the host:

```
NCIA: Receive Id.Rsp
NCIA: send CLS_ID_IND to DLU
```

In the next group of messages, the host attempts to establish a session with the client:

```
NCIA: Receive Connect.Req
NCIA: send CLS_CONNECT_CNF to DLU
NCIA: Receive Flow.Req
```

In the next two messages, the host sends data to the client:

```
NCIA: Receive Data.Req
NCIA: send CLS_DATA_IND to DLU
```

In the last two messages, the client closes the session:

```
NCIA: send CLS_DISC_IND to DLU
NCIA: Receive Disconnect.Rsp
```

Related Commands

Command	Description
debug dmsp fax-to-doc	Enables debugging of DLSw+.
debug ncia circuit	Displays circuit-related information between the NCIA server and client.
debug ncia client	Displays debug information for all NCIA client processing that occurs in the router.

debug netbios error

To display information about Network Basic Input/Output System (NetBIOS) protocol errors, use the **debug netbios error** privileged EXEC command. The **no** form of this command disables debugging output.

debug netbios error

no debug netbios error

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For complete information on the NetBIOS process, use the **debug netbios packet** command along with the **debug netbios error** command.

Examples

The following is sample output from the **debug netbios error** command. This example shows that an illegal packet has been received on the asynchronous interface.

```
Router# debug netbios error

Async1 nbf Bad packet
```

Related Commands

Command	Description
debug netbios-name-cache	Displays name caching activities on a router.
debug netbios packet	Displays general information about NetBIOS packets.

debug netbios-name-cache

To display name caching activities on a router, use the **debug netbios-name-cache** privileged EXEC command. The **no** form of this command disables debugging output.

debug netbios-name-cache

no debug netbios-name-cache

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Examine the display to diagnose problems in NetBIOS name caching.

Examples

The following is sample output from the **debug netbios-name-cache** command:

```
Router# debug netbios-name-cache

NETBIOS: L checking name ORINDA, vrn=0
NetBIOS name cache table corrupted at offset 13
NetBIOS name cache table corrupted at later offset, at location 13
NETBIOS: U chk name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
NETBIOS: U upd name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
NETBIOS: U add name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
NETBIOS: U no memory to add cache entry. name=ORINDA,addr=1000.4444.5555
NETBIOS: Invalid structure detected in netbios_name_cache_ager
NETBIOS: flushed name=ORINDA, addr=1000.4444.5555
NETBIOS: expired name=ORINDA, addr=1000.4444.5555
NETBIOS: removing entry. name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0
NETBIOS: Tossing ADD_NAME/STATUS/NAME/ADD_GROUP frame
NETBIOS: Lookup Failed -- not in cache
NETBIOS: Lookup Worked, but split horizon failed
NETBIOS: Could not find RIF entry
NETBIOS: Cannot duplicate packet in netbios_name_cache_proxy
```

Note The sample display is a composite output. Debugging output that you actually see would not necessarily occur in this sequence.

Table 137 describes the significant fields shown in the display.

Table 137 debug netbios-name-cache Field Descriptions

Field	Description
NETBIOS	NetBIOS name caching debugging output.
L, U	L means lookup; U means update.
addr=1000.4444.5555	MAC address of machine being looked up in NetBIOS name cache.
idb=TR1	Indicates that the name of machine was learned from Token Ring interface number 1; idb is into interface data block.

Table 137 debug netbios-name-cache Field Descriptions (continued)

Field	Description
vrn=0	Packet comes from virtual ring number 0. This packet actually comes from a real Token Ring interface, because virtual ring number 0 is not valid.
type=1	Indicates the way that the router learned about the specified machine. The possible values are as follows: • 1 - Learned from traffic • 2 - Learned from a remote peer • 4, 8 - Statically entered via the configuration of the router

With the first line of output, the router declares that it has examined the NetBIOS name cache table for the machine name ORINDA and that the packet that prompted the lookup came from virtual ring 0. In this case, this packet comes from a real interface—virtual ring number 0 is not valid.

```
NETBIOS: L checking name ORINDA, vrn=0
```

The following two lines indicate that an invalid NetBIOS entry exists and that the corrupted memory was detected. The invalid memory will be removed from the table; no action is needed.

```
NetBIOS name cache table corrupted at offset 13
NetBIOS name cache table corrupted at later offset, at location 13
```

The following line indicates that the router attempted to check the NetBIOS cache table for the name ORINDA with MAC address 1000.4444.5555. This name was obtained from Token Ring interface 1. The type field indicates that the name was learned from traffic.

```
NETBIOS: U chk name=ORINDA, addr=1000.4444.5555, idb=TR1, vrn=0, type=1
```

The following line indicates that the NetBIOS name ORINDA is in the name cache table and was updated to the current value:

```
NETBIOS: U upd name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
```

The following line indicates that the NetBIOS name ORINDA is not in the table and must be added to the table:

```
NETBIOS: U add name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0,type=1
```

The following line indicates that there was insufficient cache buffer space when the router tried to add this name:

```
NETBIOS: U no memory to add cache entry. name=ORINDA,addr=1000.4444.5555
```

The following line indicates that the NetBIOS ager detects an invalid memory in the cache. The router clears the entry; no action is needed.

```
NETBIOS: Invalid structure detected in netbios_name_cache_ager
```

The following line indicates that the entry for ORINDA was flushed from the cache table:

```
NETBIOS: flushed name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the entry for ORINDA timed out and was flushed from the cache table:

```
NETBIOS: expired name=ORINDA, addr=1000.4444.5555
```

The following line indicates that the router removed the ORINDA entry from its cache table:

```
NETBIOS: removing entry. name=ORINDA,addr=1000.4444.5555,idb=TR1,vrn=0
```

The following line indicates that the router discarded a NetBIOS packet of type ADD_NAME, STATUS, NAME_QUERY, or ADD_GROUP. These packets are discarded when multiple copies of one of these packet types are detected during a certain period of time.

```
NETBIOS: Tossing ADD_NAME/STATUS/NAME/ADD_GROUP frame
```

The following line indicates that the system could not find a NetBIOS name in the cache:

```
NETBIOS: Lookup Failed -- not in cache
```

The following line indicates that the system found the destination NetBIOS name in the cache, but located on the same ring from which the packet came. The router will drop this packet because the packet should not leave this ring.

```
NETBIOS: Lookup Worked, but split horizon failed
```

The following line indicates that the system found the NetBIOS name in the cache, but the router could not find the corresponding RIF. The packet will be sent as a broadcast frame.

```
NETBIOS: Could not find RIF entry
```

The following line indicates that no buffer was available to create a NetBIOS name cache proxy. A proxy will not be created for the packet, which will be forwarded as a broadcast frame.

```
NETBIOS: Cannot duplicate packet in netbios_name_cache_proxy
```

Related Commands

Command	Description
debug netbios error	Displays information about NetBIOS protocol errors.
debug netbios packet	Displays general information about NetBIOS packets.

debug netbios packet

To display general information about NetBIOS packets, use the **debug netbios packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug netbios packet

no debug netbios packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For complete information on the NetBIOS process, use the **debug netbios error** command along with the **debug netbios packet** command.

Examples

The following is sample output from the **debug netbios packet** and **debug netbios error** commands. This example shows the LLC header for an asynchronous interface followed by the NetBIOS information. For additional information on the NetBIOS fields, refer to *IBM LAN Technical Reference IEEE 802.2*.

```
Router# debug netbios packet

Async1 (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_NAME_QUERY
 Resp_correlator= 0x6F 0x0
 Src name=CS-NT-1

Async1 (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_GROUP_QUERY
 Resp_correlator= 0x6F 0x0
 Src name=COMMSERVER-WG

Async1 (i) U-format UI C_R=0x0
(i) NETBIOS_ADD_NAME_QUERY
 Resp_correlator= 0x6F 0x0
 Src name=CS-NT-1

Ethernet0 (i) U-format UI C_R=0x0
(i) NETBIOS_DATAGRAM
 Length= 0x2C 0x0
 Dest name=COMMSERVER-WG
 Src name=CS-NT-3
```

Related Commands

Command	Description
debug netbios error	Displays information about NetBIOS protocol errors.
debug netbios-name-cache	Displays name caching activities on a router.

debug nhrp

To display information about Next Hop Resolution Protocol (NHRP) activity, use the **debug nhrp** privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp

no debug nhrp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command when some nodes on a TCP/IP or IPX network are not responding. Output from the command shows whether the router is sending or receiving NHRP packets.

Examples

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

```
Router# debug nhrp

NHRP: Cache update 172.19.145.57 None
NHRP: Sent request src 172.19.145.56 dst 255.255.255.255
NHRP M: id 0 src 172.19.145.56 dst 172.19.145.57
NHRP: Encapsulation succeeded. MAC addr ffff.ffff.ffff.
NHRP: O 86 bytes out Ethernet1 dest 255.255.255.255
NHRP: Recv reply Size 64
NHRP M: id 0 src 172.19.145.56 dst 172.19.145.57
NHRP: Cache update 172.19.145.57 0000.0c14.59d3.
```

Table 138 describes the significant fields shown in the display.

Table 138 debug nhrp Field Descriptions

Field	Descriptions
NHRP and NHRP M	NHRP debugging output and mandatory header debugging output.
Cache update	NHRP cache is being revised.
Sent request src dst	NHRP request packet was sent from the specified source address. NHRP packet was sent to the specified destination address.
id src dst	Sequence number of the packet. Sequence number of the source address. Sequence number of the destination address.
Encapsulation succeeded. MAC addr	NHRP packet was encapsulated. Link-layer address used as the destination address for the NHRP packet.

Table 138 debug nhrp Field Descriptions (continued)

Field	Descriptions
O 86 bytes out Ethernet1 dest	Size of the NHRP packet (in this case, the output was 86 bytes). Interface that the packet was sent out on, and the network-layer destination address.
Recv reply Size	Indicates receipt of an NHRP reply packet and the size of the packet excluding the link-layer header.

Related Commands

Command	Description
debug nhrp options	Displays information about NHRP option processing.
debug nhrp packet	Displays a dump of NHRP packets.

debug nhrp extension

To display the extensions portion of a NHRP packet, use the **debug nhrp extension** privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp extension

no debug nhrp extension

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug nhrp extension** command:

```
Router# debug nhrp extension

NHRP extension processing debugging is on
Router#
Forward Transit NHS Record Extension(4):
 (C-1) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 7200
       addr_len: 20(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client NBMA: 47.0091810000000002ba08e101.525354555354.01
       client protocol: 135.206.58.54
Reverse Transit NHS Record Extension(5):
Responder Address Extension(3):
 (C) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 7200
       addr_len: 20(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client NBMA: 47.0091810000000002ba08e101.525354555355.01
       client protocol: 135.206.58.55
Forward Transit NHS Record Extension(4):
 (C-1) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 7200
       addr_len: 20(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client NBMA: 47.0091810000000002ba08e101.525354555354.01
       client protocol: 135.206.58.54
Reverse Transit NHS Record Extension(5):
Responder Address Extension(3):
Forward Transit NHS Record Extension(4):
Reverse Transit NHS Record Extension(5):
```

debug nhrp options

To display information about NHRP option processing, use the **debug nhrp options** privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp options

no debug nhrp options

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to show you whether there are problems or error situations with NHRP option processing (for example, unknown options).

Examples

The following is sample output from the **debug nhrp options** command:

```
Router# debug nhrp options

NHRP-OPT: MASK 4
NHRP-OPT-MASK: FFFFFFFF
NHRP-OPT: NETID 4
NHRP-OPT: RESPONDER 4
NHRP-OPT: RECORD 0
NHRP-OPT: RRECORD 0
```

Table 139 describes the significant fields shown in the display.

Table 139 debug nhrp options Field Descriptions

Field	Descriptions
NHRP-OPT	NHRP options debugging output.
MASK 4	Number of bytes of information in the destination prefix option.
NHRP-OPT-MASK	Contents of the destination prefix option.
NETID	Number of bytes of information in the subnetwork identifier option.
RESPONDER	Number of bytes of information in the responder address option.
RECORD	Forward record option.
RRECORD	Reverse record option.

Related Commands

Command	Description
debug nhrp	Displays information about NHRP activity.
debug nhrp packet	Displays a dump of NHRP packets.

debug nhrp packet

To display a dump of NHRP packets, use the **debug nhrp packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp packet

no debug nhrp packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug nhrp packet

NHRP activity debugging is on
Router#
NHRP: Send Purge Request via ATM3/0.1, packet size: 72
 src: 135.206.58.55, dst: 135.206.58.56
 (F) afn: NSAP(3), type: IP(800), hop: 255, ver: 1
     shtl: 20(NSAP), sstl: 0(NSAP)
 (M) flags: "reply required", reqid: 2
     src NBMA: 47.0091810000000002ba08e101.525354555355.01
     src protocol: 135.206.58.55, dst protocol: 135.206.58.56
 (C-1) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 0
       addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client protocol: 135.206.58.130
NHRP: Receive Purge Reply via ATM3/0.1, packet size: 72
 (F) afn: NSAP(3), type: IP(800), hop: 254, ver: 1
     shtl: 20(NSAP), sstl: 0(NSAP)
 (M) flags: "reply required", reqid: 2
     src NBMA: 47.0091810000000002ba08e101.525354555355.01
     src protocol: 135.206.58.55, dst protocol: 135.206.58.56
 (C-1) code: no error(0)
       prefix: 0, mtu: 9180, hd_time: 0
       addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 0
       client protocol: 135.206.58.130
```

debug nhrp rate

To display information about NHRP traffic rate limits, use the **debug nhrp rate** privileged EXEC command. The **no** form of this command disables debugging output.

debug nhrp rate

no debug nhrp rate

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to verify that the traffic is consistent with the setting of the NHRP commands (such as **ip nhrp use** and **ip max-send** commands).

Examples

The following is sample output from the **debug nhrp rate** command:

```
Router# debug nhrp rate

NHRP-RATE: Sending initial request
NHRP-RATE: Retransmitting request (retrans ivl 2)
NHRP-RATE: Retransmitting request (retrans ivl 4)
NHRP-RATE: Ethernet1: Used 3
```

Table 140 describes the significant fields shown in the display.

Table 140 debug nhrp rate Field Descriptions

Field	Descriptions
NHRP-RATE	NHRP rate debugging output.
Sending initial request	First time an attempt was made to send an NHRP packet to a particular destination.
Retransmitting request	Indicates that the NHRP packet was re-sent, and shows the time interval (in seconds) to wait before the NHRP packet is re-sent again.
Ethernet1:	Interface over which the NHRP packet was sent.
Used 3	Number of packets sent out of the default maximum five (in this case, three were sent).

Related Commands

Command	Description
debug nhrp	Displays information about NHRP activity.
debug nhrp options	Displays information about NHRP option processing

debug ntp

To display debug messages for Network Time Protocol (NTP) features, use the **debug ntp** command. To stop the output of ntp debugging messages, use the **no** form of this command.

debug ntp {**adjust** | **authentication** | **events** | **loopfilter** | **packets** | **params** | **refclock** | **select** | **sync** | **validity**}

no debug ntp {**adjust** | **authentication** | **events** | **loopfilter** | **packets** | **params** | **refclock** | **select** | **sync** | **validity**}

Syntax Description

adjust	Displays debugging information on NTP clock adjustments.
authentication	Displays debugging information on NTP authentication.
events	Displays debugging information on NTP events.
loopfilter	Displays debugging information on NTP loop filters.
packets	Displays debugging information on NTP packets.
params	Displays debugging information on NTP clock parameters.
refclock	Displays debugging information on NTP reference clocks.
select	Displays debugging information on NTP clock selection.
sync	Displays debugging information on NTP clock synchronization.
validity	Displays debugging information on NTP peer clock validity.

Defaults

Debug commands are disabled by default.

Command History

Release	Modification
12.0 T	This command was introduced in a release prior to Cisco IOS Release 12.1.

Related Commands

Command	Description
ntp refclock	Configures an external clock source for use with NTP services.

debug oam

To display operation and maintenance (OAM) events, use the **debug oam** privileged EXEC command. The **no** form of this command disables debugging output.

debug oam

no debug oam

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug oam

4/0(O): VCD:0x0 DM:0x300 *OAM Cell* Length:0x39
0000 0300 0070 007A 0018 0100 0000 05FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
FFFF FFFF FFFF FFFF FF6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A00 0000
```

Table 141 describes the significant fields in the display.

Table 141 debug oam Field Descriptions

Field	Description
0000	VCD Special OAM indicator.
0300	Descriptor MODE bits for the AIP.
0	GFC (4 bits).
07	VPI (8 bits).
0007	VCI (16 bits).
A	Payload type field (PTI) (4 bits).
00	Header Error Correction (8 bits).
1	OAM Fault mangement cell (4 bits).
8	OAM LOOPBACK indicator (4 bits).
01	Loopback indicator value, always 1 (8 bits).
00000005	Loopback unique ID, sequence number (32 bits).
FF6A	Fs and 6A required in the remaining cell, per UNI3.0.

debug packet

To display per-packet debugging output, use the **debug packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug packet [**interface** *number* [**vcd** *vcd-number*] | **vc** *vpi/vci* | *vc-name*]

no debug packet [**interface** *number* [**vcd** *vcd-number*] | **vc** *vpi/vci* | *vc-name*]

Syntax Description

interface *number*	(Optional) interface or subinterface number.
vcd *vcd-number*	(Optional) Number of the virtual circuit designator (VCD).
vc *vpi/vci*	(Optional) VPI and VCI numbers of the VC.
vc-name	(Optional) Name of the PVC or SVC.

Usage Guidelines

The **debug packet** command displays all process-level packets for both outbound and inbound packets. This command is useful for determining whether packets are being received and sent correctly. The output reports information online when a packet is received or a transmission is attempted.

For sent packets, the information is displayed only after the protocol data unit (PDU) is entirely encapsulated and a next hop VC is found. If information is not displayed, the address translation probably failed during encapsulation. When a next hop VC is found, the packet is displayed exactly as it will be presented on the wire. Having a display indicates that the packets are properly encapsulated for transmission.

For received packets, information is displayed for all incoming frames. The display can show whether the sending station properly encapsulates the frames. Because all incoming frames are displayed, this information is useful when performing back-to-back testing and corrupted frames cannot be dropped by an intermediary switch.

The **debug packet** command also displays the initial bytes of the actual PDU in hexadecimal. This information can be decoded only by qualified support or engineering personnel.

Caution Because the **debug packet** command generates a substantial amount of output for every packet processed, use it only when traffic on the network is low, so other activity on the system is not adversely affected.

Examples

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

```
Router# debug packet

2/0.5(I): VCD:0x9 VCI:0x23 Type:0x0 SAP:AAAA CTL:03 OUI:000000 TYPE:0800 Length0x70
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFF 0105
```

Table 142 describes the significant fields in the display.

Table 142 debug packet Field Descriptions

Field	Description
2/0.5	Indicates the subinterface that generated this packet.
(I)	Indicates a receive packet. (O) indicates an output packet.
VCD: 0x*n*	Indicates the virtual circuit associated with this packet, where *n* is some value.
DM: 0x*nnnn*	Indicates the descriptor mode bits on output only, where *nnnn* is a hexadecimal value.
TYPE:*n*	Displays the encapsulation type for this packet.
Length:*n*	Displays the total length of the packet including the headers.

The following two lines of output are the binary data, which are the contents of the protocol PDU before encapsulation:

```
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFF 0105
```

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

```
Router# debug packet

Ethernet0: Unknown ARPA, src 0000.0c00.6fa4, dst ffff.ffff.ffff, type 0x0a0
data 00000c00f23a00000c00ab45, len 60
Serial3: Unknown HDLC, size 64, type 0xaaaa, flags 0x0F00
Serial2: Unknown PPP, size 128
Serial7: Unknown FRAME-RELAY, size 174, type 0x5865, DLCI 7a
Serial0: compressed TCP/IP packet dropped
```

Table 143 describes the significant fields shown in the display.

Table 143 debug packet Field Descriptions

Field	Description
Ethernet0	Name of the Ethernet interface that received the packet.
Unknown	Network could not classify this packet. Examples include packets with unknown link types.

Table 143 debug packet Field Descriptions (continued)

Field	Description
ARPA	Packet uses ARPA-style encapsulation. Possible encapsulation styles vary depending on the media command mode (MCM) and encapsulation style. Ethernet (MCM)—Encapsulation Style: • APOLLO • ARP • ETHERTALK • ISO1 • ISO3 • LLC2 • NOVELL-ETHER • SNAP
	FDDI (MCM)—Encapsulation Style: • APOLLO • ISO1 • ISO3 • LLC2 • SNAP Frame Relay—Encapsulation Style: • BRIDGE • FRAME-RELAY

Table 143 debug packet Field Descriptions (continued)

Field	Description
	Serial (MCM)—Encapsulation Style: • BFEX25 • BRIDGE • DDN-X25 • DDNX25-DCE • ETHERTALK • FRAME-RELAY • HDLC • HDH • LAPB • LAPBDCE • MULTI-LAPB • PPP • SDLC-PRIMARY • SDLC-SECONDARY • SLIP • SMDS • STUN • X25 • X25-DCE
	Token Ring (MCM)—Encapsulation Style: • 3COM-TR • ISO1 • ISO3 • MAC • LLC2 • NOVELL-TR • SNAP • VINES-TR
src 0000.0c00.6fa4	MAC address of the node generating the packet.
dst.ffff.ffff.ffff	MAC address of the destination node for the packet.
type 0x0a0	Packet type.
data...	First 12 bytes of the datagram following the MAC header.
len 60	Length of the message (in bytes) that the interface received from the wire.
size 64	Length of the message (in bytes) that the interface received from the wire. Equivalent to the len field.

Table 143 debug packet Field Descriptions (continued)

Field	Description
flags 0x0F00	HDLC or PP flags field.
DLCI 7a	The DLCI number on Frame Relay.
compressed TCP/IP packet dropped	TCP header compression is enabled on an interface and the packet is not HDLC or X25.

debug pots

To display information on the telephone interfaces, use the **debug pots** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug pots {**driver** | **csm**} [**1** | **2**]

no debug pots {**driver** | **csm**} [**1** | **2**]

Syntax Description

driver	Displays driver debug information.
csm	Displays CSM debug information.
1	(Optional) Displays information for telephone port 1 only.
2	(Optional) Displays information for telephone port 2 only.

Usage Guidelines

The **debug pots** command displays driver and CSM debug information for telephone ports 1 and 2.

Examples

The following is a sample display from the **debug pots driver 1** command. This sample display indicates that the telephone port driver is not receiving caller ID information from the ISDN line. Therefore, the analog caller ID device attached to the telephone port does not display caller ID information.

```
Router# debug pots driver 1

00:01:51:POTS DRIVER port=1 activate ringer: cadence=0 callerId=Unknown
00:01:51:POTS DRIVER port=1 state=Idle drv_event=RING_EVENT
00:01:51:POTS DRIVER port=1 enter_ringing
00:01:51:POTS DRIVER port=1 cmd=19
00:01:51:POTS DRIVER port=1 activate disconnect
00:01:51:POTS DRIVER port=1 state=Ringing drv_event=DISCONNECT_EVENT
00:01:51:POTS DRIVER port=1 cmd=1A
00:01:51:POTS DRIVER port=1 enter_idle
00:01:51:POTS DRIVER port=1 ts connect: 0 0
00:01:51:POTS DRIVER port=1 cmd=D
00:01:51:POTS DRIVER port=1 report onhook
00:01:51:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:01:51:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:01:51:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:01:51:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:01:53:POTS DRIVER port=1 activate ringer: cadence=0 callerId=Unknown
00:01:53:POTS DRIVER port=1 state=Idle drv_event=RING_EVENT
00:01:53:POTS DRIVER port=1 enter_ringing
00:01:53:POTS DRIVER port=1 cmd=19
00:01:55:POTS DRIVER port=1 cmd=1A
00:02:49:POTS DRIVER port=1 state=Ringing drv_event=OFFHOOK_EVENT
00:02:49:POTS DRIVER port=1 cmd=1A
00:02:49:POTS DRIVER port=1 enter_suspend
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 report offhook
00:02:49:POTS DRIVER port=1 activate connect: endpt=1 calltype=TWO_PARTY_CALL
00:02:49:POTS DRIVER port=1 state=Suspend drv_event=CONNECT_EVENT
00:02:49:POTS DRIVER port=1 enter_connect: endpt=1 calltype=0
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 ts connect: 1 0
00:02:49:POTS DRIVER port=1 activate connect: endpt=1 calltype=TWO_PARTY_CALL
```

```
00:02:49:POTS DRIVER port=1 state=Connect drv_event=CONNECT_EVENT
00:02:49:POTS DRIVER port=1 enter_connect: endpt=1 calltype=0
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 ts connect: 1 0
00:02:55:POTS DRIVER port=1 state=Connect drv_event=ONHOOK_EVENT
00:02:55:POTS DRIVER port=1 enter_idle
00:02:55:POTS DRIVER port=1 ts connect: 0 0
00:02:55:POTS DRIVER port=1 cmd=D
00:02:55:POTS DRIVER port=1 report onhook
00:02:55:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:02:55:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:02:55:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:02:55:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
```

The following is sample display from the **debug pots csm 1** command. This sample display indicates that a dial peer contains an invalid destination pattern (555-1111).

```
Router# debug pots csm 1

01:57:28:EVENT_FROM_ISDN:dchanidb=0x66CB38, call_id=0x11, ces=0x2 bchan=0x0, event=0x1,
cause=0x0
01:57:28:Dial peer not found, route call to port 1
01:57:28:CSM_PROC_IDLE:CSM_EVENT_ISDN_CALL, call_id=0x11, port=1
01:57:28:Calling number ‘5551111’
01:57:40:CSM_PROC_RINGING:CSM_EVENT_VDEV_OFFHOOK, call_id=0x11, port=1
01:57:40:EVENT_FROM_ISDN:dchan_idb=0x66CB38, call_id=0x11, ces=0x2 bchan=0x0, event=0x4,
cause=0x0
01:57:40:CSM_PROC_CONNECTING:CSM_EVENT_ISDN_CONNECTED, call_id=0x11, port=1
01:57:47:CSM_PROC_CONNECTING:CSM_EVENT_VDEV_ONHOOK, call_id=0x11, port=1
01:57:201863503872: %ISDN-6-DISCONNECT:Interface BRI0:1 disconnected from unknown, call
lasted 5485 seconds
01:57:47: %ISDN-6-DISCONNECT:Interface BRI0:1 disconnected from unknown, call lasted 5485
seconds
01:57:47:EVENT_FROM_ISDN:dchan _idb=0x66CB38, call_id=0x11, ces=0x2 bchan=0xFFFFFFFF,
event=0x0, cause=0x1
01:57:47:CSM_PROC_NEAR_END_DISCONNECT:CSM_
```

debug pots csm

To activate events from which an application can determine and display the status and progress of calls to and from POTS ports, use the **debug pots csm** EXEC command.

debug pots csm

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1.(2)XF	This command was introduced on the Cisco 800 series routers.

Usage Guidelines

To see debug messages, enter the **logging console** global configuration mode command as follows:

```
router(config)# logging console

router(config)# exit
```

Debug messages are displayed in one of two formats that are relevant to the POTS dial feature:

```
hh:mm:ss: CSM_STATE: CSM_EVENT, call id = ??, port = ?
```

or

```
hh:mm:ss: EVENT_FROM_ISDN:dchan_idb=0x???????, call_id=0x????, ces=? bchan=0x????????,
event=0x?, cause=0x??
```

Table 144 describes the significant fields shown in the display.

Table 144 debug pots csm Field Descriptions:

Command Elements	Description
hh:mm:ss	Timestamp (in hours, minutes, and seconds).
CSM_STATE	One of the call CSM states listed in Table 145.
CSM_EVENT	One of the CSM events listed in Table 146.
call id	Hexadecimal value from 0x00 to 0xFF.
port	Telephone port 1 or 2.
EVENT_FROM_ISDN	A CSM event. Table 146 shows a list of CSM events.
dchan_idb	Internal data structure address.
ces	Connection end point suffix used by ISDN.
bchan	Channel used by the call. A value of 0xFFFFFFFF indicates that a channel is not assigned.

Command Elements	Description
event	A hexadecimal value that is translated into a CSM event. Table 147 shows a list of events and the corresponding CSM events.
cause	A hexadecimal value that is given to call-progressing events. Table 148 shows a list of cause values and definitions.

Table 145 shows the values for CSM states.

Table 145 CSM States

CSM State	Description
CSM_IDLE_STATE	Telephone on the hook.
CSM_RINGING	Telephone ringing.
CSM_SETUP	Setup for outgoing call in progress.
CSM_DIALING	Dialing number of outgoing call.
CSM_IVR_DIALING	Interactive voice response (IVR) for Japanese telephone dialing.
CSM_CONNECTING	Waiting for carrier to connect the call.
CSM_CONNECTED	Call connected.
CSM_DISCONNECTING	Waiting for carrier to disconnect the call.
CSM_NEAR_END_DISCONNECTING	Waiting for carrier to disconnect the call.
CSM_HARD_HOLD	Call on hard hold.
CSM_CONSULTATION_HOLD	Call on consultation hold.
CSM_WAIT_FOR_HOLD	Waiting for carrier to put call on hard hold.
CSM_WAIT_FOR_CONSULTATION_HOLD	Waiting for carrier to put call on consultation hold.
CSM_CONFERENCE	Waiting for carrier to complete call conference.
CSM_TRANSFER	Waiting for carrier to transfer call.
CSM_APPLIC_DIALING	Call initiated from Cisco IOS CLI.

Table 146 shows the values for CSM events.

Table 146 CSM Events

CSM Events	Description
CSM_EVENT_INTER_DIGIT_TIMEOUT	Time waiting for dial digits has expired.
CSM_EVENT_TIMEOUT	Near- or far-end disconnect timeout.
CSM_EVENT_ISDN_CALL	Incoming call.
CSM_EVENT_ISDN_CONNECTED	Call connected.
CSM_EVENT_ISDN_DISCONNECT	Far end disconnected.
CSM_EVENT_ISDN_DISCONNECTED	Call disconnected.
CSM_EVENT_ISDN_SETUP	Outgoing call requested.

Table 146 CSM Events (continued)

CSM Events	Description
CSM_EVENT_ISDN_SETUP_ACK	Outgoing call accepted.
CSM_EVENT_ISDN_PROC	Call proceeding and dialing completed.
CSM_EVENT_ISDN_CALL_PROGRESSING	Call being received in band tone.
CSM_EVENT_ISDN_HARD_HOLD	Call on hard hold.
CSM_EVENT_ISDN_HARD_HOLD_REJ	Hold attempt rejected.
CSM_EVENT_ISDN_CHOLD	Call on consultation hold.
CSM_EVENT_ISDN_CHOLD_REJ	Consultation hold attempt rejected.
CSM_EVENT_ISDN_RETRIEVED	Call retrieved.
CSM_EVENT_ISDN_RETRIEVE_REJ	Call retrieval attempt rejected.
CSM_EVENT_ISDN_TRANSFERRED	Call transferred.
CSM_EVENT_ISDN_TRANSFER_REJ	Call transfer attempt rejected.
CSM_EVENT_ISDN_CONFERENCE	Call conference started.
CSM_EVENT_ISDN_CONFERENCE_REJ	Call conference attempt rejected.
CSM_EVENT_ISDN_IF_DOWN	ISDN interface down.
CSM_EVENT_ISDN_INFORMATION	ISDN information element received (used by NTT IVR application).
CSM_EVENT_VDEV_OFFHOOK	Telephone off the hook.
CSM_EVENT_VDEV_ONHOOK	Telephone on the hook.
CSM_EVENT_VDEV_FLASHHOOK	Telephone hook switch has flashed.
CSM_EVENT_VDEV_DIGIT	DTMF digit has been detected.
CSM_EVENT_VDEV_APPLICATION_CALL	Call initiated from Cisco IOS CLI.

Table 147 shows the values for events that are translated into CSM events.

Table 147 Event Values

Hexadecimal Value	Event	CSM Event
0x0	DEV_IDLE	CSM_EVENT_ISDN_DISCONNECTED
0x1	DEV_INCALL	CSM_EVENT_ISDN_CALL
0x2	DEV_SETUP_ACK	CSM_EVENT_ISDN_SETUP_ACK
0x3	DEV_CALL_PROC	CSM_EVENT_ISDN_PROC
0x4	DEV_CONNECTED	CSM_EVENT_ISDN_CONNECTED
0x5	DEV_CALL_PROGRESSING	CSM_EVENT_ISDN_CALL_PROGRESSING
0x6	DEV_HOLD_ACK	CSM_EVENT_ISDN_HARD_HOLD
0x7	DEV_HOLD_REJECT	CSM_EVENT_ISDN_HARD_HOLD_REJ
0x8	DEV_CHOLD_ACK	CSM_EVENT_ISDN_CHOLD
0x9	DEV_CHOLD_REJECT	CSM_EVENT_ISDN_CHOLD_REJ

Table 147 Event Values (continued)

Hexadecimal Value	Event	CSM Event
0xa	DEV_RETRIEVE_ACK	CSM_EVENT_ISDN_RETRIEVED
0xb	DEV_RETRIEVE_REJECT	CSM_EVENT_ISDN_RETRIEVE_REJ
0xc	DEV_CONFR_ACK	CSM_EVENT_ISDN_CONFERENCE
0xd	DEV_CONFR_REJECT	CSM_EVENT_ISDN_CONFERENCE_REJ
0xe	DEV_TRANS_ACK	CSM_EVENT_ISDN_TRANSFERRED
0xf	DEV_TRANS_REJECT	CSM_EVENT_ISDN_TRANSFER_REJ

Table 148 shows cause values that are assigned only to call-progressing events.

Table 148 Cause Values

Hexadecimal Value	Cause Definitions
0x01	UNASSIGNED_NUMBER
0x02	NO_ROUTE
0x03	NO_ROUTE_DEST
0x04	NO_PREFIX
0x06	CHANNEL_UNACCEPTABLE
0x07	CALL_AWARDED
0x08	CALL_PROC_OR_ERROR
0x09	PREFIX_DIALED_ERROR
0x0a	PREFIX_NOT_DIALED
0x0b	EXCESSIVE_DIGITS
0x0d	SERVICE_DENIED
0x10	NORMAL_CLEARING
0x11	USER_BUSY
0x12	NO_USER_RESPONDING
0x13	NO_USER_ANSWER
0x15	CALL_REJECTED
0x16	NUMBER_CHANGED
0x1a	NON_SELECTED_CLEARING
0x1b	DEST_OUT_OF_ORDER
0x1c	INVALID_NUMBER_FORMAT
0x1d	FACILITY_REJECTED
0x1e	RESP_TO_STAT_ENQ
0x1f	UNSPECIFIED_CAUSE
0x22	NO_CIRCUIT_AVAILABLE
0x26	NETWORK_OUT_OF_ORDER

Table 148 Cause Values (continued)

Hexadecimal Value	Cause Definitions
0x29	TEMPORARY_FAILURE
0x2a	NETWORK_CONGESTION
0x2b	ACCESS_INFO_DISCARDED
0x2c	REQ_CHANNEL_NOT_AVAIL
0x2d	PRE_EMPTED
0x2f	RESOURCES_UNAVAILABLE
0x32	FACILITY_NOT_SUBSCRIBED
0x33	BEARER_CAP_INCOMPAT
0x34	OUTGOING_CALL_BARRED
0x36	INCOMING_CALL_BARRED
0x39	BEARER_CAP_NOT_AUTH
0x3a	BEAR_CAP_NOT_AVAIL
0x3b	CALL_RESTRICTION
0x3c	REJECTED_TERMINAL
0x3e	SERVICE_NOT_ALLOWED
0x3f	SERVICE_NOT_AVAIL
0x41	CAP_NOT_IMPLEMENTED
0x42	CHAN_NOT_IMPLEMENTED
0x45	FACILITY_NOT_IMPLEMENT
0x46	BEARER_CAP_RESTRICTED
0x4f	SERV_OPT_NOT_IMPLEMENT
0x51	INVALID_CALL_REF
0x52	CHAN_DOES_NOT_EXIST
0x53	SUSPENDED_CALL_EXISTS
0x54	NO_CALL_SUSPENDED
0x55	CALL_ID_IN_USE
0x56	CALL_ID_CLEARED
0x58	INCOMPATIBLE_DEST
0x5a	SEGMENTATION_ERROR
0x5b	INVALID_TRANSIT_NETWORK
0x5c	CS_PARAMETER_NOT_VALID
0x5f	INVALID_MSG_UNSPEC
0x60	MANDATORY_IE_MISSING
0x61	NONEXISTENT_MSG
0x62	WRONG_MESSAGE
0x63	BAD_INFO_ELEM

Table 148 Cause Values (continued)

Hexadecimal Value	Cause Definitions
0x64	INVALID_ELEM_CONTENTS
0x65	WRONG_MSG_FOR_STATE
0x66	TIMER_EXPIRY
0x67	MANDATORY_IE_LEN_ERR
0x6f	PROTOCOL_ERROR
0x7f	INTERWORKING_UNSPEC

Examples

This section provides debug output examples for three call scenarios, displaying the sequence of events that occur during a POTS dial call or POTS disconnect call.

Call Scenario 1

In this example call scenario, port 1 is on the hook, the application dial is set to call 4085552221, and the far-end successfully connects.

```
Router# debug pots csm

Router# test pots 1 dial 4085552221#

Router#
```

The following screen output shows an event indicating that port 1 is being used by the dial application:

```
01:58:27: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
```

The following screen output shows events indicating that the CSM is receiving the application digits of the number to dial:

```
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
```

The following screen output shows that the telephone connected to port 1 is off the hook:

```
    01:58:39: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
```

The following screen output shows a call-proceeding event pair indicating that the router ISDN software has sent the dialed digits to the ISDN switch:

```
01:58:40: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0x0,
event=0x3, cause=0x0
01:58:40: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id =
0x8004, port = 1
```

The following screen output shows the call-progressing event pair indicating that the telephone at the far end is ringing:

```
01:58:40: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x0
01:58:40: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8004, port
= 1
```

The following screen output shows a call-connecting event pair indicating that the telephone at the far end has answered:

```
01:58:48: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x4, cause=0x0
01:58:48: CSM_PROC_CONNECTING: CSM_EVENT_ISDN_CONNECTED, call id = 0x8004, port = 1
```

The following screen output shows a call-progressing event pair indicating that the telephone at the far end has hung up and that the calling telephone is receiving an in-band tone from the ISDN switch:

```
01:58:55: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x10
01:58:55: CSM_PROC_CONNECTED: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8004, port = 1
```

The following screen output shows that the telephone connected to port 1 has hung up:

```
01:58:57: CSM_PROC_CONNECTED: CSM_EVENT_VDEV_ONHOOK, call id = 0x8004, port = 1
```

The following screen output shows an event pair indicating that the call has been terminated:

```
01:58:57: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x0, cause=0x0
01:58:57: CSM_PROC_NEAR_END_DISCONNECT: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8004,
port = 1
813_local#
```

Call Scenario 2

In this example scenario, port 1 is on the hook, the application dial is set to call 4085552221, and the destination number is busy.

```
Router# debug pots csm

Router# test pots 1 dial 4085552221#

Router#
```

The following screen output shows that port 1 is used by the dial application:

```
01:59:42: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
```

The following screen output shows the events indicating that the CSM is receiving the application digits of the number to call:

```
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
```

The following screen output shows an event indicating that the telephone connected to port 1 is off the hook:

```
01:59:52: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
```

The following screen output shows a call-proceeding event pair indicating that the telephone at the far end is busy:

```
01:59:52: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0x0,
event=0x3, cause=0x11
01:59:52: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id = 0x8005, port = 1
```

The following screen output shows a call-progressing event pair indicating that the calling telephone is receiving an in-band busy tone from the ISDN switch:

```
01:59:58: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x0
01:59:58: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8005, port
= 1
```

The following screen output shows an event indicating that the calling telephone has hung up:

```
02:00:05: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_VDEV_ONHOOK, call id = 0x8005, port = 1
```

The following screen output shows an event pair indicating that the call has been terminated:

```
02:00:05: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0xFFFFFFFF,
event=0x0, cause=0x0
02:00:05: CSM_PROC_NEAR_END_DISCONNECT: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8005,
port = 1
```

Call Scenario 3

In this example call scenario, port 1 is on the hook, the application dial is set to call 408-666-1112, the far end successfully connects, and the command **test pots disconnect** terminates the call:

```
Router# debug pots csm

Router# test pots 1 dial 4086661112

Router#
```

The following screen output follows the same sequence of events as shown in Call Scenario 1:

```
1d03h: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1

1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1

1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
   bchan=0x0, event=0x3, cause=0x0
1d03h: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id = 0x8039, port = 1

1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
   bchan=0xFFFFFFFF, event=0x5, cause=0x0

1d03h: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8039,
    port = 1
```

```
Router# test pots 1 disconnect
```

The **test pots disconnect** command disconnects the call before you physically need to put the telephone back on the hook:

```
1d03h: CSM_PROC_CONNECTING: CSM_EVENT_VDEV_APPLICATION_HANGUP_CALL, call id = 0x8039,
     port = 1
1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
   bchan=0xFFFFFFFF, event=0x0, cause=0x0

1d03h: CSM_PROC_DISCONNECTING: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8039,
    port = 1
1d03h: CSM_PROC_DISCONNECTING: CSM_EVENT_TIMEOUT, call id = 0x8039, port = 1
```

debug ppp

To display information on traffic and exchanges in an internetwork implementing the PPP, use the **debug ppp** privileged EXEC command. The **no** form of this command disables debugging output.

debug ppp {**packet** | **negotiation** | **error** | **authentication** | **compression** | **cbcp**}

no debug ppp {**packet** | **negotiation** | **error** | **authentication** | **compression** | **cbcp**}

Syntax Description

packet	Displays PPP packets being sent and received. (This command displays low-level packet dumps.)
negotiation	Displays PPP packets sent during PPP startup, where PPP options are negotiated.
error	Displays protocol errors and error statistics associated with PPP connection negotiation and operation.
authentication	Displays authentication protocol messages, including Challenge Authentication Protocol (CHAP) packet exchanges and Password Authentication Protocol (PAP) exchanges.
compression	Displays information specific to the exchange of PPP connections using MPPC. This command is useful for obtaining incorrect packet sequence number information where MPPC compression is enabled.
cbcp	Displays protocol errors and statistics associated with PPP connection negotiations using MSCB.

Usage Guidelines

Use the **debug ppp** command when trying to find the following:

- The Network Control Protocols (NCPs) that are supported on either end of a PPP connection
- Any loops that might exist in a PPP internetwork
- Nodes that are (or are not) properly negotiating PPP connections
- Errors that have occurred over the PPP connection
- Causes for CHAP session failures
- Causes for PAP session failures
- Information specific to the exchange of PPP connections using the Callback Control Protocol (CBCP), used by Microsoft clients
- Incorrect packet sequence number information where MPPC compression is enabled

Refer to Internet RFCs 1331, 1332, and 1333 for details concerning PPP-related nomenclature and protocol information.

Caution The **debug ppp compression** command is CPU-intensive and should be used with caution. This command should be disabled immediately after debugging.

Examples

The following is sample output from the **debug ppp packet** command as seen from the Link Quality Monitor (LQM) side of the connection. This display example depicts packet exchanges under normal PPP operation.

```
Router# debug ppp packet

PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 3 len = 12
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 4 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 4 len = 12
PPP Serial4: O LCP ECHOREP(A) id 4 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 5 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 5 len = 12
PPP Serial4: O LCP ECHOREP(A) id 5 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 6 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 6 len = 12
PPP Serial4: O LCP ECHOREP(A) id 6 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 7 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 7 len = 12
PPP Serial4: O LCP ECHOREP(A) id 7 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

Table 149 describes the significant fields shown in the display.

Table 149 debug ppp packet Field Descriptions

Field	Description
PPP	PPP debugging output.
Serial4	Interface number associated with this debugging information.
(o), O	Packet was detected as an output packet.
(i), I	Packet was detected as an input packet.
lcp_slqr()	Procedure name; running LQM, send a Link Quality Report (LQR).
lcp_rlqr()	Procedure name; running LQM, received an LQR.
input (C021)	Router received a packet of the specified packet type (in hexadecimal notation). A value of C025 indicates packet of type LQM.
state = OPEN	PPP state; normal state is OPEN.

Table 149 debug ppp packet Field Descriptions (continued)

Field	Description
magic = D21B4	Magic Number for indicated node; when output is indicated, this is the Magic Number of the node on which debugging is enabled. The actual Magic Number depends on whether the packet detected is indicated as I or O.
datagramsize 52	Packet length including header.
code = ECHOREQ(9)	Identifies the type of packet received. Both forms of the packet, string and hexadecimal, are presented.
len = 48	Packet length without header.
id = 3	ID number per Link Control Protocol (LCP) packet format.
pkt type 0xC025	Packet type in hexadecimal notation; typical packet types are C025 for LQM and C021 for LCP.
LCP ECHOREQ(9)	Echo Request; value in parentheses is the hexadecimal representation of the LCP type.
LCP ECHOREP(A)	Echo Reply; value in parentheses is the hexadecimal representation of the LCP type.

To elaborate on the displayed output, consider the partial exchange. This sequence shows that one side is using ECHO for its keepalives and the other side is using LQRs.

```
Router# debug ppp packet

PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 3 len = 12
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The first line states that the router with debugging enabled has sent an LQR to the other side of the PPP connection:

```
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The next two lines indicate that the router has received a packet of type C025 (LQM) and provides details about the packet:

```
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
```

The next two lines indicate that the router received an ECHOREQ of type C021 (LCP). The other side is sending ECHOs. The router on which debugging is configured for LQM but also responds to ECHOs.

```
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
```

Next, the router is detected to have responded to the ECHOREQ with an ECHOREP and is preparing to send out an LQR:

```
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The following is sample output from the **debug ppp negotiation** command. This is a normal negotiation, where both sides agree on Network Control Program (NCP) parameters. In this case, protocol type IP is proposed and acknowledged.

```
Router# debug ppp negotiation

ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = 3D567F8 acked (ok)
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
ppp: ipcp_reqci: returning CONFACK.
 (ok)
PPP Serial4: state = ACKSENT fsm_rconfack(8021): rcvd id 4
```

Table 150 describes significant fields shown in the display.

Table 150 debug ppp Command Negotiation Field Descriptions

Field	Description
ppp	PPP debugging output.
sending CONFREQ	Router sent a configuration request.
type = 4 (CI_QUALITYTYPE)	Type of LCP configuration option that is being negotiated and a descriptor. A type value of 4 indicates Quality Protocol negotiation; a type value of 5 indicates Magic Number negotiation.
value = C025/3E8	For Quality Protocol negotiation, indicates NCP type and reporting period. In the example, C025 indicates LQM; 3E8 is a hexadecimal value translating to about 10 seconds (in hundredths of a second).
value = 3D56CAC	For Magic Number negotiation, indicates the Magic Number being negotiated.
received config	Receiving node has received the proposed option negotiation for the indicated option type.
acked	Acknowledgment and acceptance of options.
state = ACKSENT	Specific PPP state in the negotiation process.
ipcp_reqci	IPCP notification message; sending CONFACK.
fsm_rconfack (8021)	Procedure fsm_rconfack processes received CONFACKs, and the protocol (8021) is IP.

The first two lines indicate that the router is trying to bring up LCP and will use the indicated negotiation options (Quality Protocol and Magic Number). The value fields are the values of the options themselves. C025/3E8 translates to Quality Protocol LQM. 3E8 is the reporting period (in hundredths of a second). 3D56CAC is the value of the Magic Number for the router.

```
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
```

The next two lines indicate that the other side negotiated for options 4 and 5 as requested and acknowledged both. If the responding end does not support the options, a CONFREJ is sent by the responding node. If the responding end does not accept the value of the option, a CONFNAK is sent with the value field modified.

```
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = 3D567F8 acked (ok)
```

The next three lines indicate that the router received a CONFACK from the responding side and displays accepted option values. Use the rcvd id field to verify that the CONFREQ and CONFACK have the same ID field.

```
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
```

The next line indicates that the router has IP routing enabled on this interface and that the IPCP NCP negotiated successfully:

```
ppp: ipcp_reqci: returning CONFACK.
```

In the last line, the state of the router is listed as ACKSENT.

```
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5\
```

The following is sample output from when the **debug ppp packet** and **debug ppp negotiation** commands are enabled at the same time.

```
router# debug ppp negotiation
router# debug ppp packet

ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = D4C64
PPP Serial4: O LCP CONFREQ(1) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 76 100
PPP Serial4(i): pkt type 0xC021, datagramsize 22
PPP Serial4: I LCP CONFREQ(1) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 84 240
PPP Serial4: input(C021) state = REQSENT code = CONFREQ(1) id = 4 len = 18
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = D54F0 acked
PPP Serial4: O LCP CONFACK(2) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 84 240 (ok)
PPP Serial4(i): pkt type 0xC021, datagramsize 22
PPP Serial4: I LCP CONFACK(2) id 4 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 0 13 76 100
PPP Serial4: input(C021) state = ACKSENT code = CONFACK(2) id = 4 len = 18
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 4
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = D4C64
ipcp: sending CONFREQ, type = 3 (CI_ADDRESS), Address = 2.1.1.2
PPP Serial4: O IPCP CONFREQ(1) id 3 (10) Type3 (6) 2 1 1 2
PPP Serial4: I IPCP CONFREQ(1) id 3 (10) Type3 (6) 2 1 1 1
PPP Serial4(i): pkt type 0x8021, datagramsize 14
PPP Serial4: input(8021) state = REQSENT code = CONFREQ(1) id = 3 len = 10
ppp Serial4: Negotiate IP address: her address 2.1.1.1 (ACK)
ppp: ipcp_reqci: returning CONFACK.
PPP Serial4: O IPCP CONFACK(2) id 3 (10) Type3 (6) 2 1 1 1 (ok)
PPP Serial4: I IPCP CONFACK(2) id 3 (10) Type3 (6) 2 1 1 2
PPP Serial4: input(8021) state = ACKSENT code = CONFACK(2) id = 3 len = 10
PPP Serial4: state = ACKSENT fsm_rconfack(8021): rcvd id 3
ipcp: config ACK received, type = 3 (CI_ADDRESS), Address = 2.1.1.2
PPP Serial4(o): lcp_slqr() state = OPEN magic = D4C64, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D54F0, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D54F0, len = 48
PPP Serial4(o): lcp_slqr() state = OPEN magic = D4C64, len = 48
```

This field shows a decimal representation of the Magic Number.

This field shows a decimal representation of the NCP value.

This field shows a decimal representation of the reporting period.

This exchange represents a successful PPP negotiation for support of NCP type IPCP.

S2877

The following is sample output from the **debug ppp negotiation** command when the remote side of the connection is unable to respond to LQM requests:

```
Router# debug ppp negotiation

ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44C1488
```

The following is sample output when no response is detected for configuration requests (with both the **debug ppp negotiation** and **debug ppp packet** command enabled):

```
Router# debug ppp negotiation

Router# debug ppp packet

ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 14 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E0980 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 15 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E1828 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 16 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E27C8 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: O LCP CONFREQ(1) id 17 (12) QUALITYTYPE (8) 192 37 0 0 3 232
   MAGICNUMBER (6) 4 77 253 200
ppp: TIMEout: Time= 44E3768 State= 3
```

The following is sample output from the **debug ppp error** command. These messages might appear when the Quality Protocol option is enabled on an interface that is already running PPP.

```
Router# debug ppp error

PPP Serial3(i): rlqr receive failure. successes = 15
PPP: myrcvdiffp = 159 peerxmitdiffp = 41091
PPP: myrcvdiffo = 2183 peerxmitdiffo = 1714439
PPP: threshold = 25
```

```
PPP Serial4(i): rlqr transmit failure. successes = 15
PPP: myxmitdiffp = 41091 peerrcvdiffp = 159
PPP: myxmitdiffo = 1714439 peerrcvdiffo = 2183
PPP: l->OutLQRs = 1 LastOutLQRs = 1
PPP: threshold = 25
PPP Serial3(i): lqr_protrej() Stop sending LQRs.
PPP Serial3(i): The link appears to be looped back.
```

Table 151 describes the significant fields shown in the display.

Table 151 debug ppp Error Field Descriptions

Field	Description
PPP	PPP debugging output.
Serial3(i)	Interface number associated with this debugging information; indicates that this is an input packet.
rlqr receive failure	Request to negotiate the Quality Protocol option is not accepted.
myrcvdiffp = 159	Number of packets received over the time period.
peerxmitdiffp = 41091	Number of packets sent by the remote node over this period.
myrcvdiffo = 2183	Number of octets received over this period.
peerxmitdiffo = 1714439	Number of octets sent by the remote node over this period.
threshold = 25	Maximum error percentage acceptable on this interface. This percentage is calculated by the threshold value entered in the **ppp quality** *number* interface configuration command. A value of 100 – *number* (100 minus *number*) is the maximum error percentage. In this case, a *number* of 75 was entered. This means that the local router must maintain a minimum 75 percent non-error percentage, or the PPP link will be considered down.
OutLQRs = 1	Local router's current send LQR sequence number.
LastOutLQRs = 1	The last sequence number that the remote node side has seen from the local node.

The following is sample output from the **debug ppp authentication** command. Use this **debug** command to determine why an authentication fails.

```
Router# debug ppp authentication

Serial0: Unable to authenticate. No name received from peer
Serial0: Unable to validate CHAP response. USERNAME pioneer not found.
Serial0: Unable to validate CHAP response. No password defined for USERNAME pioneer
Serial0: Failed CHAP authentication with remote.
Remote message is Unknown name
Serial0: remote passed CHAP authentication.
Serial0: Passed CHAP authentication with remote.
Serial0: CHAP input code = 4 id = 3 len = 48
```

In general, these messages are self-explanatory. Fields that can show optional output are outlined in Table 152.

Table 152 debug ppp authentication Field Descriptions

Field	Description
Serial0	Interface number associated with this debugging information and CHAP access session in question.
USERNAME pioneer not found.	The name *pioneer* in this example is the name received in the CHAP response. The router looks up this name in the list of usernames that are configured for the router.
Remote message is Unknown name	The following messages can appear: • No name received to authenticate • Unknown name • No secret for given name • Short MD5 response received • MD compare failed
code = 4	Specific CHAP type packet detected. Possible values are as follows: • 1—Challenge • 2—Response • 3—Success • 4—Failure
id = 3	ID number per LCP packet format.
len = 48	Packet length without header.

The following shows sample output from the **debug ppp** command using the **cbcp** keyword. This output depicts packet exchanges under normal PPP operation where the Cisco access server is waiting for the remote PC to respond to the MSCB request. The router also has **debug ppp negotiation** and **service timestamps msec** commands enabled.

```
Router# debug ppp cbcp

Dec 17 00:48:11.302: As8 MCB: User mscb Callback Number - Client ANY
Dec 17 00:48:11.306: Async8 PPP: O MCB Request(1) id 1 len 9
Dec 17 00:48:11.310: Async8 MCB: O  1  1  0  9  2  5  0  1  0
Dec 17 00:48:11.314: As8 MCB: O Request Id 1 Callback Type Client-Num delay 0
Dec 17 00:48:13.342: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:13.346: Async8 PPP: O MCB Request(1) id 2 len 9
Dec 17 00:48:13.346: Async8 MCB: O  1  2  0  9  2  5  0  1  0
Dec 17 00:48:13.350: As8 MCB: O Request Id 2 Callback Type Client-Num delay 0
Dec 17 00:48:15.370: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:15.374: Async8 PPP: O MCB Request(1) id 3 len 9
Dec 17 00:48:15.374: Async8 MCB: O  1  3  0  9  2  5  0  1  0
Dec 17 00:48:15.378: As8 MCB: O Request Id 3 Callback Type Client-Num delay 0
Dec 17 00:48:17.398: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:17.402: Async8 PPP: O MCB Request(1) id 4 len 9
Dec 17 00:48:17.406: Async8 MCB: O  1  4  0  9  2  5  0  1  0
Dec 17 00:48:17.406: As8 MCB: O Request Id 4 Callback Type Client-Num delay 0
Dec 17 00:48:19.426: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:19.430: Async8 PPP: O MCB Request(1) id 5 len 9
Dec 17 00:48:19.430: Async8 MCB: O  1  5  0  9  2  5  0  1  0
Dec 17 00:48:19.434: As8 MCB: O Request Id 5 Callback Type Client-Num delay 0
Dec 17 00:48:21.454: As8 MCB: Timeout in state WAIT_RESPONSE
```

```
Dec 17 00:48:21.458: Async8 PPP: O MCB Request(1) id 6 len 9
Dec 17 00:48:21.462: Async8 MCB: O  1  6  0  9  2  5  0  1  0
Dec 17 00:48:21.462: As8 MCB: O Request Id 6 Callback Type Client-Num delay 0
Dec 17 00:48:23.482: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:23.486: Async8 PPP: O MCB Request(1) id 7 len 9
Dec 17 00:48:23.490: Async8 MCB: O  1  7  0  9  2  5  0  1  0
Dec 17 00:48:23.490: As8 MCB: O Request Id 7 Callback Type Client-Num delay 0
Dec 17 00:48:25.510: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:25.514: Async8 PPP: O MCB Request(1) id 8 len 9
Dec 17 00:48:25.514: Async8 MCB: O  1  8  0  9  2  5  0  1  0
Dec 17 00:48:25.518: As8 MCB: O Request Id 8 Callback Type Client-Num delay 0
Dec 17 00:48:26.242: As8 PPP: I pkt type 0xC029, datagramsize 18
Dec 17 00:48:26.246: Async8 PPP: I MCB Response(2) id 8 len 16
Dec 17 00:48:26.250: Async8 MCB: I  2  8  0  10  2  C  C  1  32  34  39  32  36  31  33  0
Dec 17 00:48:26.254: As8 MCB: Received response
Dec 17 00:48:26.258: As8 MCB: Response CBK-Client-Num 2 12 12,  addr 1-2492613
Dec 17 00:48:26.262: Async8 PPP: O MCB Ack(3) id 9 len 16
Dec 17 00:48:26.266: Async8 MCB: O  3  9  0  10  2  C  C  1  32  34  39  32  36  31  33  0
Dec 17 00:48:26.270: As8 MCB: O Ack Id 9 Callback Type Client-Num delay 12
Dec 17 00:48:26.270: As8 MCB: Negotiated MCB with peer
Dec 17 00:48:26.390: As8 LCP: I TERMREQ [Open] id 4 len 8 (0x00000000)
Dec 17 00:48:26.390: As8 LCP: O TERMACK [Open] id 4 len 4
Dec 17 00:48:26.394: As8 MCB: Peer terminating the link
Dec 17 00:48:26.402: As8 MCB: Initiate Callback for mscb at 2492613 using Async
```

The following is sample output from the **debug ppp compression** command with **service timestamps** enabled and shows a typical PPP packet exchange between the router and Microsoft client where the MPPC header sequence numbers increment correctly:

```
Router# debug ppp compression

00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2003/0x0003
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2004/0x0004
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2005/0x0005
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2006/0x0006
00:04:14: BR0:1 MPPC: Decomp - hdr/exp_cc# 0x2007/0x0007
```

Table 153 describes the fields for the **debug ppp compression** output.

Table 153 debug ppp compression Field Descriptions

Field	Description
interface	Interface enabled with MPPC.
Decomp - hdr/	Decompression header and bit settings.
exp_cc#	Expected coherency count.
0x2003	Received sequence number.
0x0003	Expected sequence number.

The following shows sample output from **debug ppp negotiation** and **debug ppp error** commands, which can be used to troubleshoot initial PPP negotiation and setup errors. This example shows a virtual interface (virtual interface 1) during normal PPP operation and CCP negotiation.

```
Router# debug ppp negotiation error

Vt1 PPP: Unsupported or un-negotiated protocol. Link arp
VPDN: Chap authentication succeeded for p5200
Vi1 PPP: Phase is DOWN, Setup
Vi1 VPDN: Virtual interface created for dinesh@cisco.com
```

```
Vi1 VPDN: Set to Async interface
Vi1 PPP: Phase is DOWN, Setup
Vi1 VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
Vi1 CCP: Re-Syncing history using legacy method
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
Vi1 PPP: Treating connection as a dedicated line
Vi1 PPP: Phase is ESTABLISHING, Active Open
Vi1 LCP: O CONFREQ [Closed] id 1 len 25
Vi1 LCP:    ACCM 0x000A0000 (0x0206000A0000)
Vi1 LCP:    AuthProto CHAP (0x0305C22305)
Vi1 LCP:    MagicNumber 0x000FB69F (0x0506000FB69F)
Vi1 LCP:    PFC (0x0702)
Vi1 LCP:    ACFC (0x0802)
Vi1 VPDN: Bind interface direction=2
Vi1 PPP: Treating connection as a dedicated line
Vi1 LCP: I FORCED CONFREQ len 21
Vi1 LCP:    ACCM 0x000A0000 (0x0206000A0000)
Vi1 LCP:    AuthProto CHAP (0x0305C22305)
Vi1 LCP:    MagicNumber 0x12A5E4B5 (0x050612A5E4B5)
Vi1 LCP:    PFC (0x0702)
Vi1 LCP:    ACFC (0x0802)
Vi1 VPDN: PPP LCP accepted sent & rcv CONFACK
Vi1 PPP: Phase is AUTHENTICATING, by this end
Vi1 CHAP: O CHALLENGE id 1 len 27 from "l_4000"
Vi1 CHAP: I RESPONSE id 20 len 37 from "dinesh@cisco.com"
Vi1 CHAP: O SUCCESS id 20 len 4
Vi1 PPP: Phase is UP
Vi1 IPCP: O CONFREQ [Closed] id 1 len 10
Vi1 IPCP:    Address 15.2.2.3 (0x03060F020203)
Vi1 CCP: O CONFREQ [Not negotiated] id 1 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 IPCP: I CONFREQ [REQsent] id 1 len 34
Vi1 IPCP:    Address 0.0.0.0 (0x030600000000)
Vi1 IPCP:    PrimaryDNS 0.0.0.0 (0x810600000000)
Vi1 IPCP:    PrimaryWINS 0.0.0.0 (0x820600000000)
Vi1 IPCP:    SecondaryDNS 0.0.0.0 (0x830600000000)
Vi1 IPCP:    SecondaryWINS 0.0.0.0 (0x840600000000)
Vi1 IPCP: Using the default pool
Vi1 IPCP: Pool returned 11.2.2.5
Vi1 IPCP: O CONFREJ [REQsent] id 1 len 16
Vi1 IPCP:    PrimaryWINS 0.0.0.0 (0x820600000000)
Vi1 IPCP:    SecondaryWINS 0.0.0.0 (0x840600000000)
Vi1 CCP: I CONFREQ [REQsent] id 1 len 15
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP:    Stacker history 1 check mode EXTENDED (0x1105000104)
Vi1 CCP: Already accepted another CCP option, rejecting this STACKER
Vi1 CCP: O CONFREJ [REQsent] id 1 len 9
Vi1 CCP:    Stacker history 1 check mode EXTENDED (0x1105000104)
Vi1 IPCP: I CONFACK [REQsent] id 1 len 10
Vi1 IPCP:    Address 15.2.2.3 (0x03060F020203)
Vi1 CCP: I CONFACK [REQsent] id 1 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: I CONFREQ [ACKrcvd] id 2 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: O CONFACK [ACKrcvd] id 2 len 10
Vi1 CCP:    MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: State is Open
Vi1 IPCP: I CONFREQ [ACKrcvd] id 2 len 22
Vi1 IPCP:    Address 0.0.0.0 (0x030600000000)
Vi1 IPCP:    PrimaryDNS 0.0.0.0 (0x810600000000)
Vi1 IPCP:    SecondaryDNS 0.0.0.0 (0x830600000000)
Vi1 IPCP: O CONFNAK [ACKrcvd] id 2 len 22
Vi1 IPCP:    Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:    PrimaryDNS 171.69.1.148 (0x8106AB450194)
```

```
Vi1 IPCP:    SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: I CONFREQ [ACKrcvd] id 3 len 22
Vi1 IPCP:    Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:    PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:    SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: O CONFACK [ACKrcvd] id 3 len 22
Vi1 IPCP:    Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP:    PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP:    SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: State is Open
Vi1 IPCP: Install route to 11.2.2.5
```

debug ppp bap

To display general BACP transactions, use the **debug ppp bap** privileged EXEC command. The **no** form of this command disables debugging output.

debug ppp bap [**error** | **event** | **negotiation**]

no debug ppp bap [**error** | **event** | **negotiation**]

Syntax Description

error	(Optional) Displays local errors.
event	(Optional) Displays information about protocol actions and transitions between action states (pending, waiting, idle) on the link.
negotiation	(Optional) Displays successive steps in negotiations between peers.

Usage Guidelines

Do not use this command when memory is scarce or in very high traffic situations.

Examples

The following types of events generate the debug messages displayed in the figures in this section:

- A dial attempt failed.
- A BACP group was created.
- A BACP group was removed.
- The precedence of the group changed.
- Attempting to dial a number.
- Received a BACP message.
- Discarding a BACP message.
- Received an unknown code.
- Cannot find the appropriate BACP group on input.
- Displaying the response type.
- Incomplete mandatory options notification.
- Invalid outgoing message type.
- Unable to build an output message.
- Sending a BACP message.
- Details about the sent message (type of message, its identifier, the virtual access interface that sent it).

The following is sample output from the **debug ppp bap** command:

```
Router# debug ppp bap

BAP Virtual-Access1: group "laudrup" (2) (multilink) without precedence created

BAP laudrup: sending CallReq, id 2, len 38 on BRI3:1 to remote
BAP Virtual-Access1: received CallRsp, id 2, len 13
BAP laudrup: CallRsp, id 2, ACK
BAP laudrup: attempt1 to dial 19995776677 on BRI3
```

```
 ---> reason BAP - Multilink bundle overloaded
BAP laudrup: sending StatusInd, id 2, len 44 on Virtual-Access1 to remote
BAP Virtual-Access1: received StatusRsp, id 2, len 1
BAP laudrup: StatusRsp, id 2, ACK
```

Table 154 describes some basic information about the group, the events, and the sent-message details.

Table 154 debug ppp bap Field Descriptions

Field	Description
BAP Virtual-Access1:	Identifier of the virtual access interface in use.
group "laudrup"	Name of the BACP group.
sending CallReq	Action initiated; in this case, sending a call request.
on BRI3:1 to remote	Physical interface being used.
BAP laudrup: attempt1 to dial 19995776677 on BRI3	Call initiated, number being dialed, and physical interface being used.
---> reason BAP - Multilink bundle overloaded	Reason for initiating the BACP call.
BAP laudrup: sending StatusInd, id 2, len 44 on Virtual-Access1 to remote	Details about the sent message: It was a status indication message, had identifier 2, had a BACP datagram length 44, and was sent on virtual access interface 1. You can display information about the virtual access interface by using the **show interfaces virtual-access** EXEC command. (The length shown at the end of each negotiated option includes the 2-byte type and length header.)

The **debug ppp bap event** command might show state transitions and protocol actions, in addition to the basic **debug ppp bap** command.

The following is sample output from the **debug ppp bap event** command:

```
Router# debug ppp bap event

BAP laudrup: Idle --> AddWait
BAP laudrup: AddWait --> AddPending
BAP laudrup: AddPending --> Idle
```

The following is sample output from the **debug ppp bap event** command:

```
Router# debug ppp bap event

Peer does not support a message type
No response to a particular request
No response to all request retransmissions
Not configured to initiate link addition
Expected action by peer has not occurred
Exceeded number of retries
No links available to call out
Unable to provide phone numbers for callback
Maximum number of links in the group
Minimum number of links in the group
Unable to process link addition at present
Unable to process link removal at present
Not configured/unable to initiate link removal
Link addition completed notification
Link addition failed notification
Determination of location of the group config
Link with specified discriminator not in group
Link removal failed
Call failure with status
Failed to dial specified number
Discarding retransmission
Unable to find received identifier
Received StatusInd when no call pending
Discarding message with no phone delta
Unable to send message in particular state
Received a zero identifier
Request has precedence
```

The error messages displayed might be added to the basic output when the **debug ppp bap error** command is used. Because the errors are very rare, you might never see these messages.

```
Router# debug ppp bap error

Unable to find appropriate request for received response
Invalid message type of queue
Received request is not part of the group
Add link attempt failed to locate group
Remove link attempt failed to locate group
Unable to inform peer of link addition
Changing of precedence cannot locate group
Received short header/illegal length/short packet
Invalid configuration information length
Unable to NAK incomplete options
Unable to determine current number of links
No interface list to dial on
Attempt to send invalid data
Local link discriminator is not in group
Received response type is incorrect for identifier
```

The messages displayed might be added to the basic output when the **debug ppp bap negotiation** command is used:

```
Router# debug ppp bap negotiation

BAP laudrup: adding link speed 64 kbps for type 0x1 len 5
BAP laudrup: adding reason "User initiated addition", len 25
BAP laudrup: CallRsp, id 4, ACK
BAP laudrup: link speed 64 kbps for types 0x1, len 5 (ACK)
BAP laudrup: phone number "1: 0 2: ", len 7 (ACK)
BAP laudrup: adding call status 0, action 0 len 4
BAP laudrup: adding 1 phone numbers "1: 0 2: " len 7
BAP laudrup: adding reason "Successfully added link", len 25
BAP laudrup: StatusRsp, id 4, ACK
```

Additional negotiation messages might also be displayed for the following:

```
Received BAP message
Sending message
Decode individual options for send/receive
Notification of invalid options
```

The following shows additional reasons for a particular BAP action that might be displayed in an "adding reason" line of the **debug ppp bap negotiation** command output:

```
"Outgoing add request has precedence"
"Outgoing remove request has precedence"
"Unable to change request precedence"
"Unable to determine valid phone delta"
"Attempting to add link"
"Link addition is pending"
"Attempting to remove link"
"Link removal is pending"
"Precedence of peer marked CallReq for no action"
"Callback request rejected due to configuration"
"Call request rejected due to configuration"
"No links of specified type(s) available"
"Drop request disallowed due to configuration"
"Discriminator is invalid"
"No response to call requests"
"Successfully added link"
"Attempt to dial destination failed"
"No interfaces present to dial out"
"No dial string present to dial out"
"Mandatory options incomplete"
"Load has not exceeded threshold"
"Load is above threshold"
"Currently attempting to dial destination"
"No response to CallReq from race condition"
```

Table 155 describes the reasons for a BACP Negotiation Action.

Table 155 Explanation of Reasons for BACP Negotiation Action

Reason	Explanation
"Outgoing add request has precedence"	Received a CallRequest or CallbackRequest while we were waiting on a CallResponse or CallbackResponse to a sent request. We are the favored peer from the initial BACP negotiation, so we are issuing a NAK to our peer request.
"Outgoing remove request has precedence"	Received a LinkDropQueryRequest while waiting on a LinkDropQueryResponse to a sent request. We are the favored peer from the initial BACP negotiation, therefore we are issuing a NAK to our peer request.
"Unable to change request precedence"	Received a CallRequest, CallbackRequest, or LinkDropQueryRequest while waiting on a LinkDropQueryResponse to a sent request. Our peer is deemed to be the favored peer from the initial BACP negotiation and we were unable to change the status of our outgoing request in response to the favored request, so we are issuing a NAK. (This is an internal error and should never be seen.)
"Unable to determine valid phone delta"	Received a CallRequest from our peer but are unable to provide the required phone delta for the response, so we are issuing a NAK. (This is an internal error and should never be seen.)
"Attempting to add link"	Received a LinkDropQueryRequest while attempting to add a link; a NAK is issued.
"Link addition is pending"	Received a LinkDropQueryRequest, CallRequest, or CallbackRequest while attempting to add a link as the result of a previous operation; a NAK is issued in the response.
"Attempting to remove link"	Received a CallRequest or CallbackRequest while attempting to remove a link; a NAK is issued.
"Link removal is pending"	Received a CallRequest, CallbackRequest, or LinkDropQueryRequest while attempting to remove a link as the result of a previous operation; a NAK is issued in the response.
"Precedence of peer marked CallReq for no action"	Received an ACK to a previously unfavored CallRequest; we are issuing a CallStatusIndication to inform our peer that there will be no further action on our part as per this response.
"Callback request rejected due to configuration"	Received a CallbackRequest but we are configured not to accept them; a REJect is issued to our peer.
"Call request rejected due to configuration"	Received a CallRequest but we are configured not to accept them; a REJect is issued to our peer.
"No links of specified type(s) available"	We received a CallRequest but no links of the specified type and speed are available; a NAK is issued.
"Drop request disallowed due to configuration"	Received a LinkDropQueryRequest but we are configured not to accept them; a NAK is issued to our peer.

Table 155 Explanation of Reasons for BACP Negotiation Action (continued)

Reason	Explanation
"Discriminator is invalid"	Received a LinkDropQueryRequest but the local link discriminator is not contained within the bundle; a NAK is issued.
"No response to call requests"	After no response to our CallRequest message, a CallStatusIndication is sent to the peer informing that no more action will be taken on behalf of this operation.
"Successfully added link"	Sent as part of the CallStatusIndication informing our peer that we successfully completed the addition of a link to the bundle as the result of the transmission of a CallRequest or the reception of a CallbackRequest.
"Attempt to dial destination failed"	Sent as part of the CallStatusIndication informing our peer that we failed in an attempt to add a link to the bundle as the result of the transmission of a CallRequest or the reception of a CallbackRequest. The retry field with the CallStatusIndication informs the peer of our intentions.
"No interfaces present to dial out"	There are no available interfaces to dial out on to attempt to add a link to the bundle, and we will not retry the dial attempt.
"No dial string present to dial out"	We do not have a dial string to dial out with to attempt to add a link to the bundle, and we are not going to retry the dial attempt. (This is an internal error and should never be seen.)
"Mandatory options incomplete"	Received a CallRequest, CallbackRequest, LinkDropQueryRequest, or CallStatusIndication and the mandatory options are not present, so a NAK is issued in the response. (A CallStatusResponse is an ACK, however).
"Load has not exceeded threshold"	Received a CallRequest or CallbackRequest but we are issuing a NAK in the response. We are monitoring the load of the bundle, and so we determine when links should be added to the bundle.
"Load is above threshold"	Received a LinkDropQueryRequest but we are issuing a NAK in the response. We are monitoring the load of the bundle, and so we determine when links should be removed from the bundle.
"Currently attempting to dial destination"	Received a CallbackRequest which is a retransmission of one that we previously ACK'd and are dialing the number suggested in the request. We are issuing an ACK because we did so previously, even though our peer never saw the previous response.
"No response to CallReq from race condition"	We issued a CallRequest but failed to receive a response, and we are issuing a CallStatusIndication to inform our peer of our intention not to proceed with the operation.

debug ppp multilink fragments

To display information about individual multilink fragments and important multilink events, use the **debug ppp multilink fragments** privileged EXEC command. The **no** form of this command disables debugging output.

debug ppp multilink fragments

no debug ppp multilink fragments

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Caution The **debug ppp multilink fragments** command has some memory overhead and should not be used when memory is scarce or in very high traffic situations.

Examples

The following is sample output from the **debug ppp multilink fragments** command when used with the **ping** EXEC command. The debug output indicates that a multilink PPP packet on interface BRI 0 (on the B channel) is an input (I) or output (O) packet. The output also identifies the sequence number of the packet and the size of the fragment.

```
Router# debug ppp multilink fragments

Router# ping 7.1.1.7
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 7.1.1.7, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/34/36 ms
Router#
2:00:28: MLP BRI0: B-Channel 1: O seq 80000000: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000001: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000001: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000000: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000002: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000003: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000003: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000002: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000004: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000005: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000005: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000004: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000006: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000007: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000007: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000006: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000008: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000009: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000009: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000008: size 58
```

debug ppp multilink events

To display information about events affecting multilink groups established for BACP, use the **debug ppp multilink events** privileged EXEC command. The **no** form of this command disables debugging output.

debug ppp multilink events

no debug ppp multilink events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Caution Do not use this command when memory is scarce or in very high traffic situations.

Examples

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

```
Router# debug ppp multilink events

MLP laudrup: established BAP group 4 on Virtual-Access1, physical BRI3:1
MLP laudrup: removed BAP group 4
```

Other event messages include the following:

```
Unable to find bundle for BAP group identifier
Unable to find physical interface to start BAP
Unable to create BAP group
Attempt to start BACP when inactive or running
Attempt to start BACP on non-MLP interface
Link protocol has gone down, removing BAP group
Link protocol has gone down, BAP not running or present
```

Table 156 describes the significant fields shown in the display.

Table 156 debug ppp multilink events Field Descriptions

Field	Description
MLP laudrup	Name of the multilink group.
established BAP group 4	Internal identifier. The same identifiers are used in the **show ppp bap group** command output.
Virtual-Access1	Dynamic access interface number.
physical BRI3:1	Bundle was established from a call on this interface.
removed BAP group 4	When the bundle is removed, the associated BACP group (with its ID) is also removed.

debug priority

To display priority queueing output, use the **debug priority** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug priority

no debug priority

Syntax Description

This command has no arguments or keywords.

Examples

The following example shows how to enable priority queueing output:

```
Router# debug priority

Priority output queueing debugging is on
```

The following is sample output from the **debug priority** command when the Frame Relay PVC Interface Priority Queueing (FR PIPQ) feature is configured on serial interface 0:

```
Router# debug priority

00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 200 -> medium
00:49:05:PQ:Serial0 output (Pk size/Q 24/1)
00:49:05:PQ:Serial0 dlci 300 -> normal
00:49:05:PQ:Serial0 output (Pk size/Q 24/2)
00:49:05:PQ:Serial0 dlci 400 -> low
00:49:05:PQ:Serial0 output (Pk size/Q 24/3)
```

Related Commands

Command	Description
debug custom-queue	Displays custom queueing output.

debug proxy h323 statistics

To enable proxy RTP statistics, use the **debug proxy h323 statistics** privileged EXEC command. The **no** form of this command disables the proxy RTP statistics.

debug proxy h323 statistics

no debug proxy h323 statistics

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)NA	This command was introduced.

Usage Guidelines

Enter the **show proxy h323 detail-call** EXEC command to see the statistics.

debug pvcd

To display the PVC Discovery events and ILMI MIB traffic used when discovering PVCs, use the **debug pvcd** privileged EXEC command. The **no** form of this command disables debugging output.

debug pvcd

no debug pvcd

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is primarily used by Cisco technical support representatives.

Examples

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

```
Router# debug pvcd

PVCD: PVCD enabled w/ Subif
PVCD(2/0): clearing event queue
PVCD: 2/0 Forgetting discovered PVCs...
PVCD: Removing all dynamic PVCs on 2/0
PVCD: Restoring MIXED PVCs w/ default parms on 2/0
PVCD: Marking static PVCs as UNKNWN on 2/0
PVCD: Marking static PVC 0/50 as UNKNWN on 2/0 ...
PVCD: Trying to discover PVCs on 2/0...
PVCD: pvcd_discoverPVCs
PVCD: pvcd_ping
PVCD: fPortEntry.5.0 = 2
PVCD: pvcd_getPeerVccTableSize
PVCD: fLayerEntry.5.0 = 13
PVCD:end allocating VccTable size 13
PVCD:  pvcd_getPeerVccTable
PVCD:******* 2/0: getNext on fVccEntry = NULL TYPE/VALUE numFileds = 19 numVccs = 13
PVCD: Creating Dynamic PVC 0/33 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/33: DYNAMIC
PVCD: After _create_pvc() VC 0/33: DYNAMIC0/33 on 2/0 :  UBR PCR = -1
PVCD: Creating Dynamic PVC 0/34 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/34: DYNAMIC
PVCD: After _create_pvc() VC 0/34: DYNAMIC0/34 on 2/0 :  UBR PCR -1
PVCD: Creating Dynamic PVC 0/44 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/44: DYNAMIC
PVCD: After _create_pvc() VC 0/44: DYNAMIC0/44 on 2/0 :  UBR PCR = -1
PVCD: PVC 0/50 with INHERITED_QOSTYPE
PVCD: _oi_state_change ( 0/50, 1 = ILMI_VC_UP )
PVCD: Creating Dynamic PVC 0/60 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/60: DYNAMIC
PVCD: After _create_pvc() VC 0/60: DYNAMIC0/60 on 2/0 :  UBR PCR = -1
PVCD: Creating Dynamic PVC 0/80 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/80: DYNAMIC
PVCD: After _create_pvc() VC 0/80: DYNAMIC0/80 on 2/0 :  UBR PCR = -1
PVCD: Creating Dynamic PVC 0/99 on 2/0
```

debug qllc error

To display quality link line control (QLLC) errors, use the **debug qllc error** privileged EXEC command. The **no** form of this command disables debugging output.

debug qllc error

no debug qllc error

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command helps you track down errors in the QLLC interactions with X.25 networks. Use the **debug qllc error** command in conjunction with the **debug x25 all** command to see the connection. The data shown by this command only flows through the router on the X.25 connection. Some forms of this command can generate a substantial amount of output and network traffic.

Examples

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

```
Router# debug qllc error

%QLLC-3-GENERRMSG: qllc_close - bad qllc pointer Caller 00407116 Caller 00400BD2
QLLC 4000.1111.0002: NO X.25 connection. Discarding XID and calling out
```

The following line indicates that the QLLC connection was closed:

```
%QLLC-3-GENERRMSG: qllc_close - bad qllc pointer Caller 00407116 Caller 00400BD2
```

The following line shows the virtual MAC address of the failed connection:

```
QLLC 4000.1111.0002: NO X.25 connection. Discarding XID and calling out
```

debug qllc event

To enable debugging of QLLC events, use the **debug qllc event** privileged EXEC command. The **no** form of this command disables debugging output.

debug qllc event

no debug qllc event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug qllc event** command to display primitives that might affect the state of a QLLC connection. An example of these events is the allocation of a QLLC structure for a logical channel indicator when an X.25 call has been accepted with the QLLC call user data. Other examples are the receipt and transmission of LAN explorer and XID frames.

Examples

The following is sample output from the **debug qllc event** command:

```
Router# debug qllc event

QLLC: allocating new qllc lci 9
QLLC: tx POLLING TEST, da 4001.3745.1088, sa 4000.1111.0001
QLLC: rx explorer response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
QLLC: gen NULL XID, da c001.3745.1088, sa 4000.1111.0001, rif 0830.1A91.1901.A040, dsap 4,
ssap 4
QLLC: rx XID response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
```

The following line indicates that a new QLLC data structure has been allocated:

```
QLLC: allocating new qllc lci 9
```

The following lines show transmission and receipt of LAN explorer or test frames:

```
QLLC: tx POLLING TEST, da 4001.3745.1088, sa 4000.1111.0001
QLLC: rx explorer response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
```

The following lines show XID events:

```
QLLC: gen NULL XID, da c001.3745.1088, sa 4000.1111.0001, rif 0830.1A91.1901.A040, dsap 4,
ssap 4
QLLC: rx XID response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
```

debug qllc packet

To display QLLC events and QLLC data packets, use the **debug qllc packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug qllc packet

no debug qllc packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command helps you to track down errors in the QLLC interactions with X.25 networks. The data shown by this command only flows through the router on the X25 connection. Use the **debug qllc packet** command in conjunction with the **debug x25 all** command to see the connection and the data that flows through the router.

Examples

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

```
Router# debug qllc packet

14:38:05: Serial2/5 QLLC I: Data Packet.-RSP    9 bytes.
14:38:07: Serial2/6 QLLC I: Data Packet.-RSP 112 bytes.
14:38:07: Serial2/6 QLLC O: Data Packet. 128 bytes.
14:38:08: Serial2/6 QLLC I: Data Packet.-RSP    9 bytes.
14:38:08: Serial2/6 QLLC I: Data Packet.-RSP 112 bytes.
14:38:08: Serial2/6 QLLC O: Data Packet. 128 bytes.
14:38:08: Serial2/6 QLLC I: Data Packet.-RSP    9 bytes.
14:38:12: Serial2/5 QLLC I: Data Packet.-RSP 112 bytes.
14:38:12: Serial2/5 QLLC O: Data Packet. 128 bytes.
```

The following lines indicate that a packet was received on the interfaces:

```
14:38:05: Serial2/5 QLLC I: Data Packet.-RSP    9 bytes.
14:38:07: Serial2/6 QLLC I: Data Packet.-RSP 112 bytes.
```

The following lines show that a packet was sent on the interfaces:

```
14:38:07: Serial2/6 QLLC O: Data Packet. 128 bytes.
14:38:12: Serial2/5 QLLC O: Data Packet. 128 bytes.
```

debug qllc state

To enable debugging of QLLC events, use the **debug qllc state** privileged EXEC command. The **no** form of this command disables debugging output.

debug qllc state

no debug qllc state

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug qllc state** command to show when the state of a QLLC connection has changed. The typical QLLC connection goes from states ADM to SETUP to NORMAL. The NORMAL state indicates that a QLLC connection exists and is ready for data transfer.

Examples

The following is sample output from the **debug qllc state** command:

```
Router# debug qllc state

Serial2 QLLC O: QSM-CMD
Serial2: X25 O D1 DATA (5) Q 8 lci 9 PS 4 PR 3
QLLC: state ADM -> SETUP
Serial2: X25 I D1 RR (3) 8 lci 9 PR 5
Serial2: X25 I D1 DATA (5) Q 8 lci 9 PS 3 PR 5
Serial2 QLLC I: QUA-RSPQLLC: addr 00, ctl 73

QLLC: qsetupstate: recvd qua rsp
QLLC: state SETUP -> NORMAL
```

The following line indicates that a QLLC connection attempt is changing state from ADM to SETUP:

```
QLLC: state ADM -> SETUP
```

The following line indicates that a QLLC connection attempt is changing state from SETUP to NORMAL:

```
QLLC: state SETUP -> NORMAL
```

debug qllc timer

To display QLLC timer events, use the **debug qllc timer** privileged EXEC command. The **no** form of this command disables debugging output.

debug qllc timer

no debug qllc timer

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The QLLC process periodically cycles and checks status of itself and its partner. If the partner is not found in the desired state, a LAPB primitive command is re-sent until the partner is in the desired state or the timer expires.

Examples

The following is sample output from the **debug qllc timer** command:

```
Router# debug qllc timer

14:27:24: Qllc timer lci 257, state ADM retry count 0 Caller 00407116 Caller 00400BD2
14:27:34: Qllc timer lci 257, state NORMAL retry count 0
14:27:44: Qllc timer lci 257, state NORMAL retry count 1
14:27:54: Qllc timer lci 257, state NORMAL retry count 1
```

The following line of output shows the state of a QLLC partner on a given X.25 logical channel identifier:

```
14:27:24: Qllc timer lci 257, state ADM retry count 0 Caller 00407116 Caller 00400BD2
```

Other messages are informational and appear every ten seconds.

debug qllc x25

To display X.25 packets that affect a QLLC connection, use the **debug qllc x25** privileged EXEC command. The **no** form of this command disables debugging output.

debug qllc x25

no debug qllc x25

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is helpful to track down errors in the QLLC interactions with X.25 networks. Use the **debug qllc x25** command in conjunction with the **debug x25 events** or **debug x25 all** commands to see the X.25 events between the router and its partner.

Examples

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

```
Router# debug qllc x25

15:07:23: QLLC X25 notify lci 257 event 1
15:07:23: QLLC X25 notify lci 257 event 5
15:07:34: QLLC X25 notify lci 257 event 3 Caller 00407116 Caller 00400BD2
15:07:35: QLLC X25 notify lci 257 event 4
```

Table 157 describes fields of output.

Table 157 debug qllc x.25 Field Descriptions

Field	Description
15:07:23	Displays the time of day.
QLLC X25 notify 257	Indicates that this is a QLLC X25 message.
event <*n*>	Indicates the type of event, *n*. Values for *n* can be as follows: • 1—Circuit is cleared • 2—Circuit has been reset • 3—Circuit is connected • 4—Circuit congestion has cleared • 5—Circuit has been deleted

debug radius

To display information associated with RADIUS, use the **debug radius** privileged EXEC command. The **no** form of this command disables debugging output.

debug radius

no debug radius

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

RADIUS is a distributed security system that secures networks against unauthorized access. Cisco supports RADIUS under the authentication, authorization, and accounting (AAA) security system.

Use the **debug aaa authentication** command to get a high-level view of login activity. When RADIUS is used on the router, you can use the **debug radius** command for more detailed debugging information.

Examples

The following is sample output from the **debug aaa authentication** command for a RADIUS login attempt that failed. The information indicates that RADIUS is the authentication method used.

```
Router# debug aaa authentication

14:02:55: AAA/AUTHEN (164826761): Method=RADIUS
14:02:55: AAA/AUTHEN (164826761): status = GETPASS
14:03:01: AAA/AUTHEN/CONT (164826761): continue_login
14:03:01: AAA/AUTHEN (164826761): status = GETPASS
14:03:01: AAA/AUTHEN (164826761): Method=RADIUS
14:03:04: AAA/AUTHEN (164826761): status = FAIL
```

The following is partial sample output from the **debug radius** command that shows a login attempt that failed because of a key mismatch (that is, a configuration problem):

```
Router# debug radius

13:55:19: Radius: IPC Send 0.0.0.0:1645, Access-Request, id 0x7, len 57
13:55:19:         Attribute 4 6 AC150E5A
13:55:19:         Attribute 5 6 0000000A
13:55:19:         Attribute 1 7 62696C6C
13:55:19:         Attribute 2 18 19D66483
13:55:22: Radius: Received from 171.69.1.152:1645, Access-Reject, id 0x7, len 20
13:55:22: Radius: Reply for 7 fails decrypt
```

The following is partial sample output from the **debug radius** command that shows a successful login attempt as indicated by an Access-Accept message:

```
Router# debug radius

13:59:02: Radius: IPC Send 0.0.0.0:1645, Access-Request, id 0xB, len 56
13:59:02:         Attribute 4 6 AC150E5A
13:59:02:         Attribute 5 6 0000000A
13:59:02:         Attribute 1 6 62696C6C
13:59:02:         Attribute 2 18 0531FEA3
13:59:04: Radius: Received from 171.69.1.152:1645, Access-Accept, id 0xB, len 26
13:59:04:         Attribute 6 6 00000001
```

The following is partial sample output from the **debug radius** command that shows an unsuccessful login attempt as indicated by the Access-Reject message:

```
Router# debug radius

13:57:56: Radius: IPC Send 0.0.0.0:1645, Access-Request, id 0xA, len 57
13:57:56:         Attribute 4 6 AC150E5A
13:57:56:         Attribute 5 6 0000000A
13:57:56:         Attribute 1 7 62696C6C
13:57:56:         Attribute 2 18 49C28F6C
13:57:59: Radius: Received from 171.69.1.152:1645, Access-Reject, id 0xA, len 20
```

Related Commands

Command	Description
debug aaa accounting	Displays information on accountable events as they occur.
debug aaa authentication	Displays information on AAA/TACACS+ authentication.

debug ras

To display RAS events, use the **debug ras** privileged EXEC command. The **no** form of this command disables debugging output.

debug ras

no debug ras

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(2)	This command was introduced.

Examples

The following examples are sample output from the **debug ras** command.

Proxy Details Trace with RAS Trace Enabled

In the following reports, the proxy registers with the gatekeeper and the trace is collected on the proxy with RAS trace enabled. A report is taken from a proxy and a gatekeeper.

```
Router# debug ras

H.323 RAS Messages debugging is on
Router#
      RASlib::ras_sendto: msg length 34 sent to 40.0.0.33
      RASLib::RASSendGRQ: GRQ sent to 40.0.0.33
      RASLib::RASRecvData: successfully rcvd message of length 45 from 40.0.0.33:1719
      RASLib::RASRecvData: GCF rcvd from [40.0.0.33:1719] on sock[0x67E570]
      RASlib::ras_sendto: msg length 76 sent to 40.0.0.33
      RASLib::RASSendRRQ: RRQ sent to 40.0.0.33
      RASLib::RASRecvData: successfully rcvd message of length 81 from 40.0.0.33:1719
      RASLib::RASRecvData: RCF rcvd from [40.0.0.33:1719] on sock [0x67E570]

Router# debug ras

H.323 RAS Messages debugging is on
Router#
      RASLib::RASRecvData: successfully rcvd message of length 34 from 101. 0.0.1:24999
      RASLib::RASRecvData: GRQ rcvd from [101.0.0.1:24999] on sock[5C8D28]
      RASlib::ras_sendto: msg length 45 sent to 40.0.0.31
      RASLib::RASSendGCF: GCF sent to 40.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 76 from 101.0.0.1:24999
      RASLib::RASRecvData: RRQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
      RASlib::ras_sendto: msg length 81 sent to 40.0.0.31
      RASLib::RASSendRCF: RCF sent to 40.0.0.31
```

Gatekeeper Trace with RAS Turned On, Call Being Established

This report shows a proxy call scenario. A trace is collected on a gatekeeper with RAS turned on. The call is being established.

```
Router# debug ras

H.323 RAS Messages debugging is on
```

```
Router#  RASLib::RASRecvData: successfully rcvd message of length 116 from 50.0.0.12:1700
      RASLib::RASRecvData: ARQ rcvd from [50.0.0.12:1700] on sock [0x5C8D28]
      RASLib::RAS_WK_TInit: ipsock [0x68BD30] setup successful
      RASlib::ras_sendto: msg length 80 sent to 102.0.0.1
      RASLib::RASSendLRQ: LRQ sent to 102.0.0.1
      RASLib::RASRecvData: successfully rcvd message of length 111 from 102.0.0.1:1719
      RASLib::RASRecvData: LCF rcvd from [102.0.0.1:1719] on sock [0x68BD30]
     RASLib::parse_lcf_nonstd: LCF Nonstd decode succeeded, remlen = 0
RASlib::ras_sendto: msg length 16 sent to 50.0.0.12
      RASLib::RASSendACF: ACF sent to 50.0.0.12
      RASLib::RASRecvData: successfully rcvd message of length 112 from 101.0.0.1:24999
      RASLib::RASRecvData: ARQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
      RASlib::ras_sendto: msg length 93 sent to 40.0.0.31
      RASLib::RASSendACF: ACF sent to 40.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 123 from 101.0.0.1:24999
      RASLib::RASRecvData: ARQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
      RASlib::ras_sendto: msg length 16 sent to 40.0.0.31
      RASLib::RASSendACF: ACF sent to 40.0.0.31
```

Gatekeeper Trace with RAS Turned On, Call Being Torn Down

This report shows two proxy call scenarios. A trace is collected on the gatekeeper with RAS turned on. The call is being torn down.

```
Router# debug ras

H.323 RAS Messages debugging is on
Router#
      RASlib::ras_sendto: msg length 3 sent to 40.0.0.31
      RASLib::RASSendDCF: DCF sent to 40.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 55 from 101.0.0.1:24999
      RASLib::RASRecvData: DRQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
      RASlib::ras_sendto: msg length 3 sent to 40.0.0.31
      RASLib::RASSendDCF: DCF sent to 40.0.0.31
      RASLib::RASRecvData: successfully rcvd message of length 55 from 50.0.0.12:1700
      RASLib::RASRecvData: DRQ rcvd from [50.0.0.12:1700] on sock [0x5C8D28]
      RASlib::ras_sendto: msg length 3 sent to 50.0.0.12
      RASLib::RASSendDCF: DCF sent to 50.0.0.12
```

Source Proxy Trace with RAS Turned On, Call Being Established

This report shows two proxy call scenarios. A trace is collected on the source proxy with RAS turned on. The call is being established.

```
Router# debug ras

H.323 RAS Messages debugging is on
Router#  RASlib::ras_sendto: msg length 112 sent to 40.0.0.33
      RASLib::RASSendARQ: ARQ sent to 40.0.0.33
      RASLib::RASRecvData: successfully rcvd message of length 93 from 40.0.0.33:1719
      RASLib::RASRecvData: ACF rcvd from [40.0.0.33:1719] on sock [0x67E570]
     RASLib::parse_acf_nonstd: ACF Nonstd decode succeeded, remlen = 0
RASlib::ras_sendto: msg length 123 sent to 40.0.0.33
      RASLib::RASSendARQ: ARQ sent to 40.0.0.33
      RASLib::RASRecvData: successfully rcvd message of length 16 from 40.0.0.33:1719
      RASLib::RASRecvData: ACF rcvd from [40.0.0.33:1719] on sock [0x67E570]
```

Source Proxy Trace with RAS Turned On, Call Being Torn Down

This report shows two proxy call scenarios. A trace is collected on the source proxy with RAS turned on. The call is being torn down.

```
Router# debug ras

H.323 RAS Messages debugging is on
```

```
Router#  RASLib::RASSendDRQ: DRQ sent to 40.0.0.33
      RASlib::ras_sendto: msg length 55 sent to 40.0.0.33
      RASLib::RASSendDRQ: DRQ sent to 40.0.0.33
      RASLib::RASRecvData: successfully rcvd message of length 3 from 40.0.0.33:1719
      RASLib::RASRecvData: DCF rcvd from [40.0.0.33:1719] on sock [0x67E570]
      RASLib::RASRecvData: successfully rcvd message of length 3 from 40.0.0.33:1719
      RASLib::RASRecvData: DCF rcvd from [40.0.0.33:1719] on sock [0x67E570]
```

debug redundancy

To enable the display of events for troubleshooting redundant DSCs, use the **debug redundancy** privileged EXEC command. Use the **no** form of this command to turn off the command.

debug redundancy {**all** | **ui** | **clk** | **hub**}

no debug redundancy {**all** | **ui** | **clk** | **hub**}

Syntax Description

all	Displays all available information on redundant DSCs, including that specified by the following options in this table.
ui	Displays information on the user interface of the redundant DSCs.
clk	Displays information on the clocks of the redundant DSCs.
hub	Displays information on the BIC hub of the redundant DSCs. The hub is the Fast Ethernet link between the router and the DSC.

Defaults

The command is disabled by default.

Command History

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

Usage Guidelines

This command is issued from the router shelf console.

Examples

The output from this command consists of event announcements that can be used by authorized troubleshooting personnel.

debug resource-pool

To see and trace resource pool management activity, use the **debug resource-pool** privileged EXEC command. Use the **no** form of this command to disable this function.

debug resource-pool

no debug resource-pool

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Usage Guidelines

Enter the **debug resource-pool** command to see and trace resource pool management activity.

Table 158 Resource Pooling States

State	Description
RM_IDLE	No call activity.
RM_RES_AUTHOR	Call waiting for authorization, message sent to AAA.
RM_RES_ALLOCATING	Call authorized, resource-grp-mgr allocating.
RM_RES_ALLOCATED	Resource allocated, connection acknowledgment sent to signalling state. Call should get connected and become active.
RM_AUTH_REQ_IDLE	Signalling module disconnected call while in RM_RES_AUTHOR. Waiting for authorization response from AAA.
RM_RES_REQ_IDLE	Signalling module disconnected call while in RM_RES_ALLOCATING. Waiting for resource allocation response from resource-group manager.
RM_DNIS_AUTHOR	An intermediate state before proceeding with RPM authorization.
RM_DNIS_AUTH_SUCCEEDED	DNIS authorization succeeded.
RM_DNIS_RES_ALLOCATED	DNIS resource allocated.
RM_DNIS_AUTH_REQ_IDLE	DNIS authorization request idle.
RM_DNIS_AUTHOR_FAIL	DNIS authorization failed.
RM_DNIS_RES_ALLOC_SUCCESS	DNIS resource allocation succeeded.
RM_DNIS_RES_ALLOC_FAIL	DNIS resource allocation failed.
RM_DNIS_RPM_REQUEST	DNIS resource pool management requested.

You can use the resource pool state to isolate problems. For example, if a call fails authorization in the RM_RES_AUTHOR state, investigate further with AAA authorization debugs to determine whether the problem lies in the resource-pool manager, AAA, or dispatcher.

Examples

The following example shows different instances where you can use the **debug resource-pool** command:

```
Router# debug resource-pool

RM general debugging is on

Router# show debug

General OS:
  AAA Authorization debugging is on
Resource Pool:
  resource-pool general debugging is on
Router #
Router #ping 21.1.1.10
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 21.1.1.10, timeout is 2 seconds:
*Jan  8 00:10:30.358: RM state:RM_IDLE event:DIALER_INCALL DS0:0:0:0:1
*Jan  8 00:10:30.358: RM: event incoming call

/* An incoming call is received by RM */

*Jan  8 00:10:30.358: RM state:RM_DNIS_AUTHOR event:RM_DNIS_RPM_REQUEST
DS0:0:0:0:1

/* Receives an event notifying to proceed with RPM authorization while
in DNIS authorization state */

*Jan  8 00:10:30.358: RM:RPM event incoming call
*Jan  8 00:10:30.358: RPM profile cp1 found

/* A customer profile "cp1" is found  matching for the incoming call, in
the local database */

*Jan  8 00:10:30.358: RM state:RM_RPM_RES_AUTHOR
event:RM_RPM_RES_AUTHOR_SUCCESS DS0:0:0:0:1

/* Resource authorization success event received while in  resource
authorization state*/

*Jan  8 00:10:30.358: Allocated resource from res_group isdn1
*Jan  8 00:10:30.358: RM:RPM profile "cp1", allocated resource "isdn1"
successfully
*Jan  8 00:10:30.358: RM state:RM_RPM_RES_ALLOCATING
event:RM_RPM_RES_ALLOC_SUCCESS DS0:0:0:0:1

/* Resource allocation sucess event received while attempting to
allocate a resource */
*Jan  8 00:10:30.358: Se0:1 AAA/ACCT/RM: doing resource-allocated
(local) (nothing to do)
*Jan  8 00:10:30.366: %LINK-3-UPDOWN: Interface Serial0:1, changed state
to up
*Jan  8 00:10:30.370: %LINK-3-UPDOWN: Interface Serial0:1, changed state
to down
*Jan  8 00:10:30.570: Se0:1 AAA/ACCT/RM: doing resource-update (local)
cp1 (nothing to do)
*Jan  8 00:10:30.578: %LINK-3-UPDOWN: I.nterface Serial0:0, changed
state to up
```

```
*Jan  8 00:10:30.582: %DIALER-6-BIND: Interface Serial0:0 bound to
profile Dialer0...
Success rate is 0 percent (0/5)
Router #
*Jan  8 00:10:36.662: %ISDN-6-CONNECT: Interface Serial0:0 is now
connected to 71017
*Jan  8 00:10:52.990: %DIALER-6-UNBIND: Interface Serial0:0 unbound from
profile Dialer0
*Jan  8 00:10:52.990: %ISDN-6-DISCONNECT: Interface Serial0:0
disconnected from 71017 , call lasted 22 seconds
*Jan  8 00:10:53.206: %LINK-3-UPDOWN: Interface Serial0:0, changed state
to down
*Jan  8 00:10:53.206: %ISDN-6-DISCONNECT: Interface Serial0:1
disconnected from unknown , call lasted 22 seconds
*Jan  8 00:10:53.626: RM state:RM_RPM_RES_ALLOCATED event:DIALER_DISCON
DS0:0:0:0:1

/* Received  Disconnect event from signalling stack  for a call which
has a resource allocated. */

*Jan  8 00:10:53.626: RM:RPM event call drop

/* RM processing the disconnect event */

*Jan  8 00:10:53.626: Deallocated resource from res_group isdn1
*Jan  8 00:10:53.626: RM state:RM_RPM_DISCONNECTING
event:RM_RPM_DISC_ACK DS0:0:0:0:1

/* An intermediate state while the DISCONNECT event is being processed
by external servers, before RM goes back into IDLE state.
*/
```

Table 159 debug resource-pool Field Descriptions

Field	Description
RM state:RM_IDLE	Resource manager state that displays no active calls.
RM state:RM_RES_AUTHOR	Resource authorization state.
RES_AUTHOR_SUCCESS DS0: shelf:slot:port:channel	Actual physical resource that is used
Allocated resource from res_group	Physical resource group that accepts the call.
RM profile <*x*>, allocated resource <*x*>	Specific customer profile and resource group names used to accept the call.
RM state: RM_RES_ALLOCATING	Resource manager state that unifies a call with a physical resource.

debug rif

To display information on entries entering and leaving the routing information field (RIF) cache, use the **debug rif** privileged EXEC command. The **no** form of this command disables debugging output.

debug rif

no debug rif

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

In order to use the **debug rif** command to display traffic source-routed through an interface, fast switching of source route bridging (SRB) frames must first be disabled with the **no source-bridge route-cache** interface configuration command.

Examples

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

```
router# debug rif

RIF: U chk da=9000.5a59.04f9,sa=0110.2222.33c1 [4880.3201.00A1.0050] type 8 on
static/remote/0
RIF: U chk da=0000.3080.4aed,sa=0000.0000.0000 [] type 8 on TokenRing0/0
RIF: U add 1000.5a59.04f9 [4880.3201.00A1.0050] type 8
RIF: L checking da=0000.3080.4aed, sa=0000.0000.0000
RIF: rcvd TEST response from 9000.5a59.04f9
RIF: U upd da=1000.5a59.04f9,sa=0110.2222.33c1 [4880.3201.00A1.0050]
RIF: rcvd XID response from 9000.5a59.04f9
SR1: sent XID response to 9000.5a59.04f9
```

SDLLC or Local-Ack entry

Non-SDLLC or non-Local-Ack entry

S2559

The first line of output is an example of a RIF entry for an interface configured for SDLLC or Local-Ack. Table 160 describes significant fields shown in the display.

Table 160 debug rif Field Descriptions

Field	Description
RIF:	This message describes RIF debugging output.
U chk	Update checking. The entry is being updated; the timer is set to zero (0).
da=9000.5a59.04f9	Destination MAC address.
sa=0110.2222.33c1	Source MAC address. This field contains values of zero (0000.0000.0000) in a non-SDLLC or non-Local-Ack entry.
[4880.3201.00A1.0050]	RIF string. This field is blank (null RIF) in a non-SDLLC or non-Local-Ack entry.

Table 160 debug rif Field Descriptions (continued)

Field	Description
type 8	Possible values follow: • 0—Null entry • 1—This entry was learned from a particular Token Ring port (interface) • 2—Statically configured • 4—Statically configured for a remote interface • 8—This entry is to be aged • 16—This entry (which has been learned from a remote interface) is to be aged • 32—This entry is not to be aged • 64—This interface is to be used by LAN Network Manager (and is not to be aged)
on static/remote/0	This route was learned from a real Token Ring port, in contrast to a virtual ring.

The following line of output is an example of a RIF entry for an interface that is not configured for SDLLC or Local-Ack:

```
RIF: U chk da=0000.3080.4aed,sa=0000.0000.0000 [] type 8 on TokenRing0/0
```

Notice that the source address contains only zero values (0000.0000.0000), and that the RIF string is null ([ ]). The last element in the entry indicates that this route was learned from a virtual ring, rather than a real Token Ring port.

The following line shows that a new entry has been added to the RIF cache:

```
RIF: U add 1000.5a59.04f9 [4880.3201.00A1.0050] type 8
```

The following line shows that a RIF cache lookup operation has taken place:

```
RIF: L checking da=0000.3080.4aed, sa=0000.0000.0000
```

The following line shows that a TEST response from address 9000.5a59.04f9 was inserted into the RIF cache:

```
RIF: rcvd TEST response from 9000.5a59.04f9
```

The following line shows that the RIF entry for this route has been found and updated:

```
RIF: U upd da=1000.5a59.04f9,sa=0110.2222.33c1 [4880.3201.00A1.0050]
```

The following line shows that an XID response from this address was inserted into the RIF cache:

```
RIF: rcvd XID response from 9000.5a59.04f9
```

The following line shows that the router sent an XID response to this address:

```
SR1: sent XID response to 9000.5a59.04f9
```

Table 160, Part 1 explains the other possible lines of **debug rif** Command output.

Table 160, Part 1 debug rif Field Descriptions

Field	Description
RIF: L Sending XID for <*address*>	Router/bridge wanted to send a packet to *address* but did not find it in the RIF cache. It sent an XID explorer packet to determine which RIF it should use. The attempted packet is dropped.
RIF: L No buffer for XID to <*address*>	Similar to the previous description; however, a buffer in which to build the XID packet could not be obtained.
RIF: U remote rif too small <*rif*>	Packet's RIF was too short to be valid.
RIF: U rej <*address*> too big <*rif*>	Packet's RIF exceeded the maximum size allowed and was rejected. The maximum size is 18 bytes.
RIF: U upd interface <*address*>	RIF entry for this router/bridge's interface has been updated.
RIF: U ign <*address*> interface update	RIF entry that would have updated an interface corresponding to one of this router's interfaces.
RIF: U add <*address*> <*rif*>	RIF entry for *address* has been added to the RIF cache.
RIF: U no memory to add rif for <*address*>	No memory to add a RIF entry for *address*.
RIF: removing rif entry for <*address*, *type code*>	RIF entry for *address* has been forcibly removed.
RIF: flushed <*address*>	RIF entry for *address* has been removed because of a RIF cache flush.
RIF: expired <*address*>	RIF entry for *address* has been aged out of the RIF cache.

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.

debug route-map ipc

To display a summary of the one-way IPC messages set from the RP to the VIP about NetFlow policy routing when distributed Cisco Express Forwarding (dCEF) is enabled, use the **debug route-map ipc** privileged EXEC command. The **no** form of this command disables debugging output.

debug route-map ipc

no debug route-map ipc

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

This command is especially helpful for policy routing with dCEF switching.

This command displays a summary of one-way IPC messages from the RP to the VIP about NetFlow policy routing. If you execute this command on the RP, the messages are shown as "Sent." If you execute this command on the VIP console, the IPC messages are shown as "Received."

Examples

The following is sample output of the **debug route-map ipc** command executed at the RP:

```
Router# debug route-map ipc

Routemap related IPC debugging is on

Router# configure terminal

Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip cef distributed

Router(config)#^Z

Router#

RM-IPC: Clean routemap config in slot 0
RM-IPC: Sent clean-all-routemaps; len 12
RM-IPC: Download all policy-routing related routemap config to slot 0
RM-IPC: Sent add routemap test(seq:10); n_len 5; len 17
RM-IPC: Sent add acl 1 of routemap test(seq:10); len 21
RM-IPC: Sent add min 10 max 300 of routemap test(seq:10); len 24
RM-IPC: Sent add preced 1 of routemap test(seq:10); len 17
RM-IPC: Sent add tos 4 of routemap test(seq:10); len 17
RM-IPC: Sent add nexthop 50.0.0.8 of routemap test(seq:10); len 20
RM-IPC: Sent add default nexthop 50.0.0.9 of routemap test(seq:10); len 20
RM-IPC: Sent add interface Ethernet0/0/3(5) of routemap test(seq:10); len 20
RM-IPC: Sent add default interface Ethernet0/0/2(4) of routemap test(seq:10); len 20
```

The following is sample output of the **debug route-map ipc** command executed at the VIP:

```
VIP-Slot0# debug route-map ipc
```

```
Routemap related IPC debugging is on

VIP-Slot0#
RM-IPC: Rcvd clean-all-routemaps; len 12
RM-IPC: Rcvd add routemap test(seq:10); n_len 5; len 17
RM-IPC: Rcvd add acl 1 of routemap test(seq:10); len 21
RM-IPC: Rcvd add min 10 max 300 of routemap test(seq:10); len 24
RM-IPC: Rcvd add preced 1 of routemap test(seq:10); len 17
RM-IPC: Rcvd add tos 4 of routemap test(seq:10); len 17
RP-IPC: Rcvd add nexthop 50.0.0.8 of routemap test(seq:10); len 20
RP-IPC: Rcvd add default nexthop 50.0.0.9 of routemap test(seq:10); len 20
RM-IPC: Rcvd add interface Ethernet0/3 of routemap tes; len 20
RM-IPC: Rcvd add default interface Ethernet0/2 of routemap test(seq:10); len 20
```

debug rtr error

To enable logging of SA Agent run-time errors, use the **debug rtr error** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug rtr error [*probe*]

no debug rtr error [*probe*]

Syntax Description:

probe	(Optional) Number of the probe in the range from 0 to 31.

Defaults

Logging is off.

Command History

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

Usage Guidelines

The **debug rtr error** command displays run-time errors. When a probe number other than 0 is specified, all run-time errors for that probe are displayed when the probe is active. When the probe number is 0 all run-time errors relating to the Response Time Reporter scheduler process are displayed. When no probe number is specified, all run-time errors for all active probes configured on the router and probe control are displayed.

Note Use the **debug rtr error** command before using the **debug rtr trace** command because the **debug rtr error** command generates a lesser amount of debug output.

Examples

The following example shows output from the **debug rtr error** command. The output indicates failure because the target is not there or because the responder is not enabled on the target. All debug output for the Response Time Reporter (including the **debug rtr trace** command) has the format shown in Table 161.

```
Router# debug rtr error

May  5 05:00:35.483: control message failure:1
May  5 05:01:35.003: control message failure:1
May  5 05:02:34.527: control message failure:1
May  5 05:03:34.039: control message failure:1
May  5 05:04:33.563: control message failure:1
May  5 05:05:33.099: control message failure:1
May  5 05:06:32.596: control message failure:1
May  5 05:07:32.119: control message failure:1
May  5 05:08:31.643: control message failure:1
May  5 05:09:31.167: control message failure:1
May  5 05:10:30.683: control message failure:1
```

Table 161 describes the significant fields shown in the display.

Table 161 debug rtr error Field Descriptions

Field	Description
RTR 1	Number of the probe generating the message.
Error Return Code	Message identifier indicating the error type (or error itself).
LU0 RTR Probe 1	Name of the process generating the message.
in echoTarget on call luReceive LuApiReturnCode of InvalidHandle - invalid host name or API handle	Supplemental messages that pertain to the message identifier.

Related Commands

Command	Description
debug rtr trace	Traces the execution of an SA Agent operation.

debug rtr trace

To trace the execution of an SA Agent operation, use the **debug rtr trace** privileged EXEC command. To disable trace debugging output (but not **debug rtr error** output), use the **no** form of this command.

debug rtr trace [*probe*]

no debug rtr trace [*probe*]

Syntax Description:

probe	(Optional) Number of the probe in the range from 0 to 31.

Command History

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

Usage Guidelines

When a probe number other than 0 is specified, execution for that probe is traced. When the probe number is 0, the Response Time Reporter scheduler process is traced. When no probe number is specified, all active probes and every probe control is traced.

The **debug rtr trace** command also enables **debug rtr error** command for the specified probe. However, the **no debug rtr trace** command does not disable the **debug rtr error** command. You must manually disable the command by using the **no debug rtr error** command.

All debug output (including **debug rtr error** command output) has the format shown in the **debug rtr error** command output example.

Note The **debug rtr trace** command can generate a large number of debug messages. First use the **debug rtr error** command, and then use the **debug rtr trace** on a per-probe basis.

Examples

The following output is from the **debug rtr trace** command. In this example, a probe is traced through a single operation attempt: the setup of a connection to the target, and the attempt at an echo to calculate UDP packet response time.

```
Router# debug rtr trace

Router# RTR 1:Starting An Echo Operation - IP RTR Probe 1

May  5 05:25:08.584:rtt hash insert :3.0.0.3 3383
May  5 05:25:08.584:source=3.0.0.3(3383)  dest-ip=5.0.0.1(9)
May  5 05:25:08.588:sending control msg:
May  5 05:25:08.588: Ver:1 ID:51 Len:52
May  5 05:25:08.592:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:25:08.607:receiving reply
May  5 05:25:08.607: Ver:1 ID:51 Len:8
May  5 05:25:08.623:local delta:8
May  5 05:25:08.627:delta from responder:1
May  5 05:25:08.627:received <16> bytes and responseTime = 3 (ms)
May  5 05:25:08.631:rtt hash remove:3.0.0.3 3383RTR 1:Starting An Echo Operation - IP RTR
Probe 1
```

```
May  5 05:26:08.104:rtt hash insert :3.0.0.3 2974
May  5 05:26:08.104:source=3.0.0.3(2974)  dest-ip=5.0.0.1(9)
May  5 05:26:08.108:sending control msg:
May  5 05:26:08.108: Ver:1 ID:52 Len:52
May  5 05:26:08.112:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:26:08.127:receiving reply
May  5 05:26:08.127: Ver:1 ID:52 Len:8
May  5 05:26:08.143:local delta:8
May  5 05:26:08.147:delta from responder:1
May  5 05:26:08.147:received <16> bytes and responseTime = 3 (ms)
May  5 05:26:08.151:rtt hash remove:3.0.0.3 2974RTR 1:Starting An Echo Operation - IP RTR
Probe 1
```

Related Commands

Command	Description
debug rtr error	Enables logging of SA Agent run-time errors.

debug rtsp

To show the status of the Real Time Streaming Protocol (RTSP) client/server, use the **debug rstp** command. To disables the display of output use the **no** form of this command.

debug rstp *type* [**all** | **api** | **pmh** | **session** | **socket**]

[**no**] **debug rstp** *type* [**all** | **api** | **pmh** | **session** | **socket**]

Syntax Description

all	(Optional) Displays debug messages for all RTSP client debug trace.
api	(Optional) Displays debug output for the RTSP client API.
pmh	(Optional) Displays debug output for the RTSP Protocol Message Handler.
session	(Optional) Displays debug output for the RTSP client session information.
socket	(Optional) Displays debug output for the RTSP client socket data.

Defaults

Debug is not enabled.

Command History

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

Related Commands

Command	Description
debug rtsp api	Displays debug output for the RTSP client API.
debug rtsp client session	Displays debug output for the RTSP client data.
debug rtsp socket	Displays debug output for the RTSP client socket data.

debug rtsp api

To display information about the Real Time Streaming Protocol (RTSP) API messages passed down to the RTSP client, use the **debug rtsp api** command. To disable the output, use the **no** form of this command.

debug rtsp api

[**no**] **debug rtsp api**

Syntax Description

This command has no arguments or keywords.

Defaults

Debug is not enabled.

Command History

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

Examples

The following example displays output from the **debug rtsp api** command:

```
router# debug rtsp api

RTSP client API debugging is on
router#
Jan  1 00:23:15.775:rtsp_api_create_session:sess_id=0x61A07C78,
          evh=0x60D6E62C context=0x61A07B28
Jan  1 00:23:15.775:rtsp_api_request:msg=0x61C2B10C
Jan  1 00:23:15.775:rtsp_api_handle_req_set_params:msg=0x61C2B10C
Jan  1 00:23:15.775:rtsp_api_free_msg_buffer:msg=0x61C2B10C
Jan  1 00:23:15.775:rtsp_api_request:msg=0x61C293CC
Jan  1 00:23:15.775:rtsp_api_handle_req_set_params:msg=0x61C293CC
Jan  1 00:23:15.775:rtsp_api_free_msg_buffer:msg=0x61C293CC
Jan  1 00:23:15.775:rtsp_api_request:msg=0x61C2970C
Jan  1 00:23:15.775:rtsp_api_handle_req_set_params:msg=0x61C2970C
Jan  1 00:23:15.775:rtsp_api_free_msg_buffer:msg=0x61C2970C
router#
Jan  1 00:23:15.775:rtsp_api_request:msg=0x61C29A4C
router#
Jan  1 00:23:22.099:rtsp_api_free_msg_buffer:msg=0x61C29A4C
Jan  1 00:23:22.115:rtsp_api_request:msg=0x61C2A40C
Jan  1 00:23:22.115:rtsp_api_free_msg_buffer:msg=0x61C2A40C
Router#
```

Related Commands

Command	Description
debug rtsp client session	Displays debug output for the RTSP client data.
debug rtsp pmh	Displays debug messages for the PMH.
debug rtsp socket	Displays debug output for the RTSP client socket data.

debug rtsp client session

To display debug messages about the Real Time Streaming Protocol (RTSP) client or the current session, use the **debug rtsp** command. To disable the output, use the **no** form of this command.

debug rtsp [**client** | **session**]

no debug rtsp [**client** | **session**]

Syntax Description

client	(Optional) Displays client information and stream information for the stream that is currently active.
session	(Optional) Displays cumulative information about the session, packet statistics, and general call information such as call ID, session ID, individual RTSP stream URLs, packet statistics, and play duration.

Defaults

Debug is not enabled.

Command History

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

Examples

The following example displays the debug messages of the RTSP session:

```
Router# debug rtsp session

RTSP client session debugging is on
router#
Jan  1 00:08:36.099:rtsp_get_new_scb:
Jan  1 00:08:36.099:rtsp_initialize_scb:
Jan  1 00:08:36.099:rtsp_control_process_msg:
Jan  1 00:08:36.099:rtsp_control_process_msg:received MSG request of TYPE 0
Jan  1 00:08:36.099:rtsp_set_event:
Jan  1 00:08:36.099:rtsp_set_event:api_req_msg_type=RTSP_API_REQ_PLAY
Jan  1 00:08:36.103:rtsp_set_event:url:[rtsp://rtsp-cisco.cisco.com:554/en_welcome.au]
Jan  1 00:08:36.103:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:36.103:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_IDLE
                     rtsp_event = RTSP_EV_PLAY_OR_REC
Jan  1 00:08:36.103:act_idle_event_play_or_rec_req:
Jan  1 00:08:36.103:rtsp_resolve_dns:
Jan  1 00:08:36.103:rtsp_resolve_dns:IP Addr = 1.13.79.6:
Jan  1 00:08:36.103:rtsp_connect_to_svr:
Jan  1 00:08:36.103:rtsp_connect_to_svr:socket=0, connection_state = 2
Jan  1 00:08:36.103:rtsp_start_timer:timer (0x62128FD0)starts - delay (10000)
Jan  1 00:08:36.107:rtsp_control_main:SOCK= 0 Event=0x1
Jan  1 00:08:36.107:rtsp_stop_timer:timer(0x62128FD0) stops
Jan  1 00:08:36.107:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:36.107:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_IDLE
                     rtsp_event = RTSP_EV_SVR_CONNECTED
Jan  1 00:08:36.107:act_idle_event_svr_connected:
Jan  1 00:08:36.107:rtsp_control_main:SOCK= 0 Event=0x1
Jan  1 00:08:36.783:rtsp_control_main:SOCK= 0 Event=0x1
```

```
Jan  1 00:08:36.783:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:36.783:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
                     rtsp_event = RTSP_EV_SVR_DESC_OR_ANNOUNCE_RESP
Jan  1 00:08:36.783:act_ready_event_desc_or_announce_resp:
Jan  1
00:08:36.783:act_ready_event_desc_or_announce_resp:RTSP_STATUS_DESC_OR_ANNOUNCE_RESP_OK
Jan  1 00:08:37.287:rtsp_control_main:SOCK= 0 Event=0x1
Jan  1 00:08:37.287:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:37.287:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
                     rtsp_event = RTSP_EV_SVR_SETUP_RESP
Jan  1 00:08:37.287:act_ready_event_setup_resp:
Jan  1 00:08:37.287:act_ready_event_setup_resp:Remote RTP Port=13344
Jan  1 00:08:37.287:rtsp_rtp_stream_setup:scb=0x62128F08, callID=0x7 record=0
Jan  1 00:08:37.287:rtsp_rtp_stream_setup:Starting RTCP session.
        Local IP addr = 1.13.79.45, Remote IP addr = 1.13.79.6,
        Local RTP port = 18748, Remote RTP port = 13344 CallID=8
Jan  1 00:08:37.291:xmit_func = 0x0 vdbptr = 0x61A0FC98
Jan  1 00:08:37.291:rtsp_control_main:CCAPI Queue Event
Jan  1 00:08:37.291:rtsp_rtp_associate_done:ev=0x62070E08, callID=0x7
Jan  1 00:08:37.291:rtsp_rtp_associate_done:scb=0x62128F08
Jan  1 00:08:37.291:rtsp_rtp_associate_done:callID=0x7, pVdb=0x61F4FBC8,
Jan  1 00:08:37.291:                  spi_context=0x6214145C
Jan  1 00:08:37.291:                  disposition=0, playFunc=0x60CA2238,
Jan  1 00:08:37.291:                  codec=0x5, vad=0, mediaType=6,
Jan  1 00:08:37.291:                  stream_assoc_id=1
Jan  1 00:08:37.291:rtsp_rtp_modify_session:scb=0x62128F08, callID=0x7
Jan  1 00:08:37.291:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:37.291:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
                     rtsp_event = RTSP_EV_ASSOCIATE_DONE
Jan  1 00:08:37.291:act_ready_event_associate_done:
Jan  1 00:08:37.291:rtsp_get_stream:
Jan  1 00:08:37.783:rtsp_control_main:SOCK= 0 Event=0x1
Jan  1 00:08:37.783:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:37.783:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
                     rtsp_event = RTSP_EV_SVR_PLAY_OR_REC_RESP
Jan  1 00:08:37.783:act_ready_event_play_or_rec_resp:
Jan  1 00:08:37.783:rtsp_start_timer:timer (0x62128FB0)starts - delay (4249)
rtsp-5#
Jan  1 00:08:42.035:rtsp_process_timer_events:
Jan  1 00:08:42.035:rtsp_process_timer_events:PLAY OR RECORD completed
Jan  1 00:08:42.035:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:42.035:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_PLAY_OR_REC
                     rtsp_event = RTSP_EV_PLAY_OR_REC_TIMER_EXPIRED
Jan  1 00:08:42.035:act_play_event_play_done:
Jan  1 00:08:42.035:act_play_event_play_done:elapsed play time = 4249 total play time =
4249
Jan  1 00:08:42.035:rtsp_send_teardown_to_svr:
Jan  1 00:08:42.487:rtsp_control_main:SOCK= 0 Event=0x1
Jan  1 00:08:42.487:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:42.487:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_PLAY_OR_REC
                     rtsp_event = RTSP_EV_SVR_TEARDOWN_RESP
Jan  1 00:08:42.487:act_play_event_teardown_resp:
Jan  1 00:08:42.487:rtsp_server_closed:
Jan  1 00:08:42.487:rtsp_send_resp_to_api:
Jan  1 00:08:42.487:rtsp_send_resp_to_api:sending RESP=RTSP_STATUS_PLAY_COMPLETE
Jan  1 00:08:42.491:rtsp_rtp_teardown_stream:scb=0x62128F08, callID=0x7
Jan  1 00:08:42.491:rtsp_rtp_stream_cleanup:scb=0x62128F08, callID=0x7
Jan  1 00:08:42.491:rtsp_update_stream_stats:scb=0x62128F08, stream=0x61A43350,
Jan  1 00:08:42.491:call_info=0x6214C67C, callID=0x7
Jan  1 00:08:42.491:rtsp_update_stream_stats:rx_bytes = 25992
Jan  1 00:08:42.491:rtsp_update_stream_stats:rx_packetes = 82
Jan  1 00:08:42.491:rtsp_reinitialize_scb:
Jan  1 00:08:42.503:rtsp_control_process_msg:
Jan  1 00:08:42.503:rtsp_control_process_msg:received MSG request of TYPE 0
```

```
Jan  1 00:08:42.503:rtsp_set_event:
Jan  1 00:08:42.503:rtsp_set_event:api_req_msg_type=RTSP_API_REQ_DESTROY
Jan  1 00:08:42.503:rtsp_session_cleanup:
Jan  1 00:08:42.503:rtsp_create_session_history:scb=0x62128F08, callID=0x7
Jan  1 00:08:42.503:rtsp_insert_session_history_record:current=0x6214BDC8, callID=0x7
Jan  1 00:08:42.503:rtsp_insert_session_history_record:count = 3
Jan  1 00:08:42.503:rtsp_insert_session_history_record:starting history record
deletion_timer of10 minutes
Jan  1 00:08:42.503:rtsp_session_cleanup:deleting session:scb=0x62128F08
Router#
```

Related Commands

Command	Description
debug rtsp api	Displays debug output for the RTSP client API.
debug rtsp client session	Displays debug output for the RTSP client data.
debug rtsp pmh	Displays debug messages for the PMH.
debug rtsp socket	Displays debug output for the RTSP client socket data.

debug rtsp pmh

To display debug information about the Protocol Message Handler (PMH), use the **debug rtsp pmh** command. To disable the output, use the **no** form of this command.

debug rtsp pmh

no debug rtsp pmh

Syntax Description

This command has no arguments or keywords.

Defaults

Debug is not enabled.

Command History

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

Usage Guidelines

Use the **debug rtsp pmh** debug command for the following instances:

- To display packets sent by the gateway (Real Time Streaming Protocol [RTSP] client) to the RTSP server. For example:

```
Mar  1 02:25:11.447:SendBuf:DESCRIBE rtsp://rtsp-cisco.cisco.com/en_welcome.au
RTSP/1.0
CSeq:0
```

- To view packets sent by the RTSP server to the gateway. For example:

```
Mar  1 02:25:11.947:########################################
Mar  1 02:25:11.947:Mesg_line            :RTSP/1.0 200 OK
Mar  1 02:25:11.951:Content_length       :459
Mar  1 02:25:11.951:Header list
Mar  1 02:25:11.951:Content-length:459
Mar  1 02:25:11.951:Content-type:application/sdp
Mar  1 02:25:11.951:Content-base:rtsp://rtsp-cisco.cisco.com/en_welcome.au/
Mar  1 02:25:11.951:X-TSPort:7802
Mar  1 02:25:11.951:Last-Modified:Thu, 07 Oct 1999 13:51:28 GMT
Mar  1 02:25:11.951:Date:Mon, 10 Jan 2000 16:40:59 GMT
Mar  1 02:25:11.951:CSeq:0
```

Examples

The following example output displays the result from entering the **debug rtsp pmh** command:

```
Router# debug rtsp pmh

RTSP client Protocol Message Handler debugging is on
Router#
Jan  1 00:22:34.087:rtsp_pmh_update_play_req_url:
Jan  1 00:22:34.087:rtsp_pmh_parse_url:
Jan  1 00:22:34.087:Input-Url:rtsp://rtsp-cisco.cisco.com:554/en_welcome.au
Jan  1 00:22:34.087:Hostname:rtsp-cisco.cisco.com
Jan  1 00:22:34.087:Port    :554
Jan  1 00:22:34.087:Path    :en_welcome.au
```

```
Jan  1 00:22:34.091:rtsp_pmh_build_desc_req:
Jan  1 00:22:34.091:rtsp_pmh_add_req_line:
Jan  1 00:22:34.091:RequestLine:(DESCRIBE rtsp://rtsp-cisco.cisco.com:554/en_welcome.au
RTSP/1.0
)
Jan  1 00:22:34.091:SendBuf:DESCRIBE rtsp://rtsp-cisco.cisco.com:554/en_welcome.au
RTSP/1.0
CSeq:0


Jan  1 00:22:34.091:last_req  = 0
Jan  1 00:22:34.739:rtsp_pmh_parse_svr_response:
Jan  1 00:22:34.739:rtsp_pmh_create_mesg:
Jan  1 00:22:34.739:########################################
Jan  1 00:22:34.739:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:34.739:Content_length        :482
Jan  1 00:22:34.739:Header list
Jan  1 00:22:34.739:Content-length:482
Jan  1 00:22:34.739:Content-type:application/sdp
Jan  1 00:22:34.739:Content-base:rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/
Jan  1 00:22:34.739:Last-Modified:Thu, 07 Oct 1999 13:51:28 GMT
Jan  1 00:22:34.739:X-TSPort:7802
Jan  1
00:22:34.739:vsrc:http://rtsp-cisco.cisco.com:8080/viewsource/template.html?nuyhtgywkgz6mc
9AbhC4gn5gBsqp4eA1v1yeC3d4ngEt5o5gwuw4t6x05jbhcv66ngE8xg8f
Jan  1
00:22:34.739:Set-Cookie:cbid=ekeghhiljgekgihheoqohpptrrjrktlufkegkioihgjfdlplrnqogpoqlrpsk
qnuffgjcmcl;path=/;expires=Thu,31-Dec-2037 23:59:59 GMT
Jan  1 00:22:34.739:Date:Mon, 10 Apr 2000 15:39:17 GMT
Jan  1 00:22:34.739:CSeq:0
Jan  1 00:22:34.739:Message Body
Jan  1 00:22:34.739:v=0
o=- 939300688 939300688 IN IP4 1.13.79.6
s=<No title>
i=<No author> <No copyright>
a=StreamCount:integer;1
t=0 0
m=audio 0 RTP/AVP 0
a=control:streamid=0
a=rtpmap:0 L8/8000/1
a=length:npt=3.249000
a=range:npt=0-3.249000
a=mimetype:string;"audio/x-pn-au"
a=StartTime:integer;0
a=AvgBitRate:integer;64000
a=AvgPacketSize:integer;320
a=Preroll:integer;0
a=MaxPacketSize:integer;320
a=MaxBitRate:integer;64000
a=OpaqueData:buffer;"AQABAEAfAAA="
a=StreamName:string;"audio/x-pn-au"

Jan  1 00:22:34.739:########################################
Jan  1 00:22:34.739:rtsp_pmh_process_resp_headers:
Jan  1 00:22:34.739:rtsp_pmh_get_header_value:
Jan  1 00:22:34.739:rtsp_pmh_process_resp_headers:Cseq=1
Jan  1 00:22:34.739:rtsp_pmh_get_resp_line:
Jan  1 00:22:34.739:rtsp_pmh_process_resp_headers:Response Status
Jan  1 00:22:34.739:rtsp_pmh_process_resp_headers:Status Code:200
Jan  1 00:22:34.739:rtsp_pmh_process_resp_headers:Reason Phrase:OK
Jan  1 00:22:34.743:rtsp_pmh_parse_mesg_body:
Jan  1 00:22:34.743:rtsp_pmh_process_resp_headers:Response
URL:rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0
Jan  1 00:22:34.743:rtsp_pmh_process_resp_headers:RealServer Duration
```

```
Jan  1 00:22:34.743:rtsp_pmh_process_resp_headers:IP/TV Duration
Jan  1 00:22:34.743:rtsp_pmh_get_range_from_npt:
Jan  1 00:22:34.743:rtsp_pmh_get_range_from_npt:Duration:3249 msecs
Jan  1 00:22:34.743:rtsp_pmh_update_resp_status:
Jan  1 00:22:34.743:rtsp_pmh_update_resp_status:Control Not active
Jan  1 00:22:34.743:########################################
Jan  1 00:22:34.743:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:34.743:Content_length        :482
Jan  1 00:22:34.743:Header list
Jan  1 00:22:34.743:Content-length:482
Jan  1 00:22:34.743:Content-type:application/sdp
Jan  1 00:22:34.743:Content-base:rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/
Jan  1 00:22:34.743:Last-Modified:Thu, 07 Oct 1999 13:51:28 GMT
Jan  1 00:22:34.743:X-TSPort:7802
Jan  1
00:22:34.743:vsrc:http://rtsp-cisco.cisco.com:8080/viewsource/template.html?nuyhtgywkgz6mc
9AbhC4gn5gBsqp4eA1v1yeC3d4ngEt5o5gwuw4t6x05jbhcv66ngE8xg8f
Jan  1
00:22:34.743:Set-Cookie:cbid=ekeghhiljgekgihheoqohpptrrjrktlufkegkioihgjfdlplrnqogpoqlrpsk
qnuffgjcmcl;path=/;expires=Thu,31-Dec-2037 23:59:59 GMT
Jan  1 00:22:34.743:Date:Mon, 10 Apr 2000 15:39:17 GMT
Jan  1 00:22:34.743:CSeq:0
Jan  1 00:22:34.743:Message Body
Jan  1 00:22:34.743:v=0
o=- 939300688 939300688 IN IP4 1.13.79.6
s=<No title>
i=<No author> <No copyright>
a=StreamCount:integer;1
t=0 0
m=audio 0 RTP/AVP 0
a=control:streamid=0
a=rtpmap:0 L8/8000/1
a=length:npt=3.249000
a=range:npt=0-3.249000
a=mimetype:string;"audio/x-pn-au"
a=StartTime:integer;0
a=AvgBitRate:integer;64000
a=AvgPacketSize:integer;320
a=Preroll:integer;0
a=MaxPacketSize:integer;320
a=MaxBitRate:integer;64000
a=OpaqueData:buffer;"AQABAEAfAAA="
a=StreamName:string;"audio/x-pn-au"

Jan  1 00:22:34.743:########################################
Jan  1 00:22:34.743:rtsp_pmh_free_mesg:
Jan  1 00:22:34.743:rtsp_pmh_build_setup_req:
Jan  1 00:22:34.743:rtsp_pmh_add_req_line:
Jan  1 00:22:34.743:RequestLine:(SETUP
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
)
Jan  1 00:22:34.747:rtsp_pmh_build_setup_req:SendBuf:SETUP
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
CSeq:1
Transport:rtp/avp;unicast;client_port=18084


Jan  1 00:22:35.243:rtsp_pmh_parse_svr_response:
Jan  1 00:22:35.243:rtsp_pmh_create_mesg:
Jan  1 00:22:35.243:########################################
Jan  1 00:22:35.243:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:35.243:Content_length        :0
Jan  1 00:22:35.243:Header list
```

```
Jan  1
00:22:35.243:Transport:rtp/avp;unicast;client_port=18084-18085;server_port=23192-23193
Jan  1 00:22:35.243:Session:24457-1
Jan  1 00:22:35.243:Date:Mon, 10 Apr 2000 15:39:17 GMT
Jan  1 00:22:35.243:CSeq:1
Jan  1 00:22:35.243:Message Body
Jan  1 00:22:35.243:########################################
Jan  1 00:22:35.243:rtsp_pmh_process_resp_headers:
Jan  1 00:22:35.243:rtsp_pmh_get_header_value:
Jan  1 00:22:35.243:rtsp_pmh_process_resp_headers:Cseq=2
Jan  1 00:22:35.243:rtsp_pmh_get_resp_line:
Jan  1 00:22:35.243:rtsp_pmh_process_resp_headers:Response Status
Jan  1 00:22:35.243:rtsp_pmh_process_resp_headers:Status Code:200
Jan  1 00:22:35.243:rtsp_pmh_process_resp_headers:Reason Phrase:OK
Jan  1 00:22:35.243:rtsp_pmh_get_header_value:
Jan  1 00:22:35.247:rtsp_pmh_get_header_value:
Jan  1 00:22:35.247:rtsp_pmh_process_resp_headers:RTP PORT= 23192
Jan  1 00:22:35.247:rtsp_pmh_process_resp_headers:RTP PORT= 23192
Jan  1 00:22:35.247:rtsp_pmh_update_resp_status:
Jan  1 00:22:35.247:rtsp_pmh_update_resp_status:Control Not active
Jan  1 00:22:35.247:########################################
Jan  1 00:22:35.247:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:35.247:Content_length        :0
Jan  1 00:22:35.247:Header list
Jan  1
00:22:35.247:Transport:rtp/avp;unicast;client_port=18084-18085;server_port=23192-23193
Jan  1 00:22:35.247:Session:24457-1
Jan  1 00:22:35.247:Date:Mon, 10 Apr 2000 15:39:17 GMT
Jan  1 00:22:35.247:CSeq:1
Jan  1 00:22:35.247:Message Body
Jan  1 00:22:35.247:########################################
Jan  1 00:22:35.247:rtsp_pmh_free_mesg:
Jan  1 00:22:35.247:rtsp_pmh_build_play_req:
Jan  1 00:22:35.247:rtsp_pmh_add_req_line:
Jan  1 00:22:35.247:RequestLine:(PLAY
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
)
Jan  1 00:22:35.247:rtsp_pmh_build_play_req:SendBuf:PLAY
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
Session:24457-1
CSeq:2


Jan  1 00:22:35.735:rtsp_pmh_parse_svr_response:
Jan  1 00:22:35.735:rtsp_pmh_create_mesg:
Jan  1 00:22:35.739:########################################
Jan  1 00:22:35.739:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:35.739:Content_length        :0
Jan  1 00:22:35.739:Header list
Jan  1 00:22:35.739:Date:Mon, 10 Apr 2000 15:39:18 GMT
Jan  1 00:22:35.739:CSeq:2
Jan  1 00:22:35.739:Message Body
Jan  1 00:22:35.739:########################################
Jan  1 00:22:35.739:rtsp_pmh_process_resp_headers:
Jan  1 00:22:35.739:rtsp_pmh_get_header_value:
Jan  1 00:22:35.739:rtsp_pmh_process_resp_headers:Cseq=3
Jan  1 00:22:35.739:rtsp_pmh_get_resp_line:
Jan  1 00:22:35.739:rtsp_pmh_process_resp_headers:Response Status
Jan  1 00:22:35.739:rtsp_pmh_process_resp_headers:Status Code:200
Jan  1 00:22:35.739:rtsp_pmh_process_resp_headers:Reason Phrase:OK
Jan  1 00:22:35.739:rtsp_pmh_update_resp_status:
Jan  1 00:22:35.739:rtsp_pmh_update_resp_status:Control Not active
Jan  1 00:22:35.739:########################################
Jan  1 00:22:35.739:Mesg_line             :RTSP/1.0 200 OK
```

```
Jan  1 00:22:35.739:Content_length        :0
Jan  1 00:22:35.739:Header list
Jan  1 00:22:35.739:Date:Mon, 10 Apr 2000 15:39:18 GMT
Jan  1 00:22:35.739:CSeq:2
Jan  1 00:22:35.739:Message Body
Jan  1 00:22:35.739:########################################
Jan  1 00:22:35.739:rtsp_pmh_free_mesg:
Jan  1 00:22:40.011:rtsp_pmh_build_teardown_req:
Jan  1 00:22:40.011:rtsp_pmh_add_req_line:
Jan  1 00:22:40.011:RequestLine:(TEARDOWN
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
)
Jan  1 00:22:40.011:SendBuf:TEARDOWN
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
Session:24457-1
CSeq:3


Jan  1 00:22:40.443:rtsp_pmh_parse_svr_response:
Jan  1 00:22:40.443:rtsp_pmh_create_mesg:
Jan  1 00:22:40.443:########################################
Jan  1 00:22:40.443:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:40.443:Content_length        :0
Jan  1 00:22:40.443:Header list
Jan  1 00:22:40.443:Date:Mon, 10 Apr 2000 15:39:23 GMT
Jan  1 00:22:40.443:CSeq:3
Jan  1 00:22:40.443:Message Body
Jan  1 00:22:40.443:########################################
Jan  1 00:22:40.443:rtsp_pmh_process_resp_headers:
Jan  1 00:22:40.443:rtsp_pmh_get_header_value:
Jan  1 00:22:40.443:rtsp_pmh_process_resp_headers:Cseq=4
Jan  1 00:22:40.443:rtsp_pmh_get_resp_line:
Jan  1 00:22:40.443:rtsp_pmh_process_resp_headers:Response Status
Jan  1 00:22:40.443:rtsp_pmh_process_resp_headers:Status Code:200
Jan  1 00:22:40.443:rtsp_pmh_process_resp_headers:Reason Phrase:OK
Jan  1 00:22:40.443:rtsp_pmh_update_resp_status:
Jan  1 00:22:40.443:rtsp_pmh_update_resp_status:Control Not active
Jan  1 00:22:40.443:########################################
Jan  1 00:22:40.447:Mesg_line             :RTSP/1.0 200 OK
Jan  1 00:22:40.447:Content_length        :0
Jan  1 00:22:40.447:Header list
Jan  1 00:22:40.447:Date:Mon, 10 Apr 2000 15:39:23 GMT
Jan  1 00:22:40.447:CSeq:3
Jan  1 00:22:40.447:Message Body
Jan  1 00:22:40.447:########################################
Jan  1 00:22:40.447:rtsp_pmh_free_mesg:
Router#


Jan  1 00:14:20.483:rtsp_tcp_socket_connect:
Jan  1 00:14:20.483:rtsp_tcp_socket_connect:Socket = 0
Jan  1 00:14:20.483:          Dest_addr = 1.13.79.6  Dest_Port=554
Jan  1 00:14:20.487:rtsp_send_req_to_svr:Socket = 0 send_buf = DESCRIBE
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au RTSP/1.0
CSeq:0

len = 76
Jan  1 00:14:20.491:rtsp_send_req_to_svr:bytes_sent = 76

Jan  1 00:14:20.491:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:20.491:rtsp_read_svr_resp:NBYTES = -1
Jan  1 00:14:21.155:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:21.159:rtsp_read_svr_resp:NBYTES = 996
Jan  1 00:14:21.223:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
```

```
Jan  1 00:14:21.227:rtsp_read_svr_resp:RESP received OK
Jan  1 00:14:21.227:rtsp_send_req_to_svr:Socket = 0 send_buf = SETUP
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
CSeq:1
Transport:rtp/avp;unicast;client_port=18074

len = 130
Jan  1 00:14:21.227:rtsp_send_req_to_svr:bytes_sent = 130

Jan  1 00:14:21.663:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:21.663:rtsp_read_svr_resp:NBYTES = 159
Jan  1 00:14:21.663:rtsp_read_svr_resp:rcv_buf = RTSP/1.0 200 OK
CSeq:1
Date:Mon, 10 Apr 2000 15:31:04 GMT
Session:24455-1
Transport:rtp/avp;unicast;client_port=18074-18075;server_port=15562-15563


Jan  1 00:14:21.663:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:21.663:rtsp_read_svr_resp:RESP received OK
Jan  1 00:14:21.663:rtsp_send_req_to_svr:Socket = 0 send_buf = PLAY
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
Session:24455-1
CSeq:2

len = 101
Jan  1 00:14:21.667:rtsp_send_req_to_svr:bytes_sent = 101

Jan  1 00:14:22.155:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:22.155:rtsp_read_svr_resp:NBYTES = 65
Jan  1 00:14:22.155:rtsp_read_svr_resp:rcv_buf = RTSP/1.0 200 OK
CSeq:2
Date:Mon, 10 Apr 2000 15:31:04 GMT


Jan  1 00:14:22.155:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:22.155:rtsp_read_svr_resp:RESP received OK
rtsp-5#
Jan  1 00:14:26.411:rtsp_send_req_to_svr:Socket = 0 send_buf = TEARDOWN
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
Session:24455-1
CSeq:3

len = 105
Jan  1 00:14:26.411:rtsp_send_req_to_svr:bytes_sent = 105

Jan  1 00:14:26.863:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:26.863:rtsp_read_svr_resp:NBYTES = 65
Jan  1 00:14:26.863:rtsp_read_svr_resp:rcv_buf = RTSP/1.0 200 OK
CSeq:3
Date:Mon, 10 Apr 2000 15:31:09 GMT


Jan  1 00:14:26.863:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:26.863:rtsp_read_svr_resp:RESP received OK
Jan  1 00:14:26.863:rtsp_close_svr_connection:closing socket 0
Router#
```

Related Commands

Command	Description
debug rtsp api	Displays debug output for the RTSP client API.

Command	Description
debug rtsp client session	Displays debug output for the RTSP client data.
debug rtsp socket	Displays debug output for the RTSP client socket data.

debug rtsp socket

To display debug messages about the packets received or sent on the TCP or User Datagram Protocol (UDP) sockets, use the **debug rtsp socket** command. To disable the output, use the **no** form of this command.

debug rtsp socket

no debug rtsp socket

Syntax Description

This command has no arguments or keywords.

Defaults

Debug is not enabled.

Command History

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

Usage Guidelines

Each RTSP session has a TCP port for control and a UDP (RTP) port for delivery of data. The control connection (TCP socket) is used to exchange a set of messages (request from the RTSP client and the response from the server) for displaying a prompt. The **debug rtsp socket** command enables the user to debug the message exchanges being done on the TCP control connection.

Examples

The following example displays output from the **debug rtsp socket** command:

```
Router# show debug rtsp socket

Jan  1 00:14:20.483:rtsp_tcp_socket_connect:
Jan  1 00:14:20.483:rtsp_tcp_socket_connect:Socket = 0
Jan  1 00:14:20.483:          Dest_addr = 1.13.79.6  Dest_Port=554
Jan  1 00:14:20.487:rtsp_send_req_to_svr:Socket = 0 send_buf = DESCRIBE
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au RTSP/1.0
CSeq:0

len = 76
Jan  1 00:14:20.491:rtsp_send_req_to_svr:bytes_sent = 76

Jan  1 00:14:20.491:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:20.491:rtsp_read_svr_resp:NBYTES = -1
Jan  1 00:14:21.155:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:21.159:rtsp_read_svr_resp:NBYTES = 996
Jan  1 00:14:21.223:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:21.227:rtsp_read_svr_resp:RESP received OK
Jan  1 00:14:21.227:rtsp_send_req_to_svr:Socket = 0 send_buf = SETUP
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
CSeq:1
Transport:rtp/avp;unicast;client_port=18074

len = 130
Jan  1 00:14:21.227:rtsp_send_req_to_svr:bytes_sent = 130
```

```
Jan  1 00:14:21.663:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:21.663:rtsp_read_svr_resp:NBYTES = 159
Jan  1 00:14:21.663:rtsp_read_svr_resp:rcv_buf = RTSP/1.0 200 OK
CSeq:1
Date:Mon, 10 Apr 2000 15:31:04 GMT
Session:24455-1
Transport:rtp/avp;unicast;client_port=18074-18075;server_port=15562-15563

Jan  1 00:14:21.663:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:21.663:rtsp_read_svr_resp:RESP received OK
Jan  1 00:14:21.663:rtsp_send_req_to_svr:Socket = 0 send_buf = PLAY
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
Session:24455-1
CSeq:2

len = 101
Jan  1 00:14:21.667:rtsp_send_req_to_svr:bytes_sent = 101

Jan  1 00:14:22.155:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:22.155:rtsp_read_svr_resp:NBYTES = 65
Jan  1 00:14:22.155:rtsp_read_svr_resp:rcv_buf = RTSP/1.0 200 OK
CSeq:2
Date:Mon, 10 Apr 2000 15:31:04 GMT


Jan  1 00:14:22.155:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:22.155:rtsp_read_svr_resp:RESP received OK
rtsp-5#
Jan  1 00:14:26.411:rtsp_send_req_to_svr:Socket = 0 send_buf = TEARDOWN
rtsp://rtsp-cisco.cisco.com:554/en_welcome.au/streamid=0 RTSP/1.0
Session:24455-1
CSeq:3

len = 105
Jan  1 00:14:26.411:rtsp_send_req_to_svr:bytes_sent = 105

Jan  1 00:14:26.863:rtsp_read_svr_resp:Socket = 0
Jan  1 00:14:26.863:rtsp_read_svr_resp:NBYTES = 65
Jan  1 00:14:26.863:rtsp_read_svr_resp:rcv_buf = RTSP/1.0 200 OK
CSeq:3
Date:Mon, 10 Apr 2000 15:31:09 GMT

Jan  1 00:14:26.863:rtsp_read_svr_resp:rtsp_pmh_parse_svr_response complete
Jan  1 00:14:26.863:rtsp_read_svr_resp:RESP received OK
Jan  1 00:14:26.863:rtsp_close_svr_connection:closing socket 0
Router#
```

Related Commands

Command	Description
debug rtsp api	Displays debug output for the RTSP client API.
debug rtsp client session	Displays debug output for the RTSP client data.
debug rtsp pmh	Displays debug messages for the PMH.

debug rtpspi all

To debug all RTP SPI errors, sessions, and in/out functions, use the **debug rtpspi all** EXEC command. Use the **no debug rtpspi all** command to turn off debugging.

debug rtpspi all

no debug rtpspi all

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco MC3810 and Cisco 3600 series routers (except the Cisco 3620) in a private release that was not generally available.

Usage Guidelines

Caution Be careful when you use this command because it can result in console flooding and reduced voice quality.

Examples

The following example shows a debug trace for RTP SPI errors, sessions, and in/out functions on a gateway:

```
router# debug rtpspi all

RTP SPI Error, Session and function in/out tracings are enabled.

*Mar  1 00:38:59.381:rtpspi_allocate_rtp_port:Entered.
*Mar  1 00:38:59.381:rtpspi_allocate_rtp_port:allocated RTP port 16544
*Mar  1 00:38:59.381:rtpspi_allocate_rtp_port:Success. port = 16544. Leaving.
*Mar  1 00:38:59.381:rtpspi_call_setup_request:entered.
        Call Id = 5, dest = 0.0.0.0;   callInfo:
        final dest flag = 0,
        rtp_session_mode = 0x2,
        local_ip_addrs = 0x5000001,remote_ip_addrs = 0x0,
        local rtp port = 16544, remote rtp port = 0
*Mar  1 00:38:59.381:rtpspi_call_setup_request:spi_info copied for rtpspi_app_data_t.
*Mar  1 00:38:59.385:rtpspi_call_setup_request:leaving
*Mar  1 00:38:59.385:rtpspi_call_setup() entered
*Mar  1 00:38:59.385:rtpspi_initialize_ccb:Entered
```

```
*Mar  1 00:38:59.385:rtpspi_initialize_ccb:leaving
*Mar  1 00:38:59.385:rtpspi_call_setup:rtp_session_mode = 0x2
*Mar  1 00:38:59.385:rtpspi_call_setup:mode = CC_CALL_NORMAL.
        destianation number = 0.0.0.0
*Mar  1 00:38:59.385:rtpspi_call_setup:Passed local_ip_addrs=0x5000001
*Mar  1 00:38:59.385:rtpspi_call_setup:Passed local_rtp_port = 16544
*Mar  1 00:38:59.385:rtpspi_call_setup:Saved RTCP Session = 0x1AF57E0
*Mar  1 00:38:59.385:rtpspi_call_setup:Passed remote rtp port = 0.
*Mar  1 00:38:59.389:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem rtp=0,
rem ip=0x0
*Mar  1 00:38:59.389:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar  1 00:38:59.389:rtpspi_start_rtcp_session:Starting RTCP session.
        Local IP addr = 0x5000001, Remote IP addr = 0x0,
        Local RTP port = 16544, Remote RTP port = 0, mode = 0x2
*Mar  1 00:38:59.389:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar  1 00:38:59.389:rtpspi_call_setup:RTP Session creation Success.
*Mar  1 00:38:59.389:rtpspi_call_setup:calling cc_api_call_connected()
*Mar  1 00:38:59.389:rtpspi_call_setup:Leaving.
*Mar  1 00:38:59.393:rtpspi_bridge:entered. conf id = 1, src i/f = 0x1859E88,
        dest i/f = 0x1964EEC, src call id = 5, dest call id = 4
        call info = 0x1919140, xmit fn = 0xDA7494, tag = 0
*Mar  1 00:38:59.393:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar  1 00:38:59.393:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=4, voip_mode=0x2, rtp_mode=0x2, ssrc_status=0
*Mar  1 00:38:59.393:rtpspi_bridge:Calling cc_api_bridge_done() for 5(0x1AF5400) and
4(0x0).
*Mar  1 00:38:59.393:rtpspi_bridge:leaving.
*Mar  1 00:38:59.397:rtpspi_caps_ind:Entered. vdb = 0x1859E88 call id = 5, srcCallId = 4
*Mar  1 00:38:59.397:rtpspi_caps_ind:caps from VTSP:codec=0x83FB, codec_bytes=0x50,
                    fax rate=0x7F, vad=0x3 modem=0x0
*Mar  1 00:38:59.397:rtpspi_get_rtcp_session_parameters():CURRENT VALUES:
dstIF=0x1964EEC, dstCallID=4, current_seq_num=0x0
*Mar  1 00:38:59.397:rtpspi_get_rtcp_session_parameters():NEW VALUES:
dstIF=0x1964EEC, dstCallID=4, current_seq_num=0x261C
*Mar  1 00:38:59.397:rtpspi_caps_ind:Caps Used:codec=0x1, codec bytes=80,
        fax rate=0x1, vad=0x1, modem=0x1, dtmf_relay=0x1, seq_num_start=0x261D
*Mar  1 00:38:59.397:rtpspi_caps_ind:calling cc_api_caps_ind().
*Mar  1 00:38:59.397:rtpspi_caps_ind:Returning success
*Mar  1 00:38:59.397:rtpspi_caps_ack:Entered. call id = 5, srcCallId = 4
*Mar  1 00:38:59.397:rtpspi_caps_ack:leaving.
*Mar  1 00:38:59.618:rtpspi_call_modify:entered. call-id=5, nominator=0x7,
params=0x18DD440
*Mar  1 00:38:59.618:rtpspi_call_modify:leaving
*Mar  1 00:38:59.618:rtpspi_do_call_modify:Entered. call-id = 5
*Mar  1 00:38:59.622:rtpspi_do_call_modify:Remote RTP port changed. New port=16432
*Mar  1 00:38:59.622:rtpspi_do_call_modify:Remote IP addrs changed. New IP addrs=0x6000001
*Mar  1 00:38:59.622:rtpspi_do_call_modify:new mode 2 is the same as the current mode
*Mar  1 00:38:59.622:rtpspi_do_call_modify:Starting new RTCP session.
*Mar  1 00:38:59.622:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem
rtp=16432, rem ip=0x6000001
*Mar  1 00:38:59.622:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar  1 00:38:59.622:rtpspi_start_rtcp_session:Removing old RTCP session.
*Mar  1 00:38:59.622:rtpspi_start_rtcp_session:Starting RTCP session.
        Local IP addr = 0x5000001, Remote IP addr = 0x6000001,
        Local RTP port = 16544, Remote RTP port = 16432, mode = 0x2
*Mar  1 00:38:59.622:rtpspi_start_rtcp_session:RTCP Timer creation Success. (5)*(5000)
*Mar  1 00:38:59.622:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar  1 00:38:59.622:rtpspi_do_call_modify:RTP Session creation Success.
*Mar  1 00:38:59.622:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar  1 00:38:59.626:rtpspi_do_call_modify:success. leaving
*Mar  1 00:39:05.019:rtpspi_call_modify:entered. call-id=5, nominator=0x7,
params=0x18DD440
*Mar  1 00:39:05.019:rtpspi_call_modify:leaving
*Mar  1 00:39:05.019:rtpspi_do_call_modify:Entered. call-id = 5
```

```
*Mar  1 00:39:05.019:rtpspi_do_call_modify:New remote RTP port = old rtp port = 16432
*Mar  1 00:39:05.019:rtpspi_do_call_modify:New remote IP addrs = old IP addrs = 0x6000001
*Mar  1 00:39:05.019:rtpspi_do_call_modify:Mode changed. new = 3, old = 2
*Mar  1 00:39:05.019:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar  1 00:39:05.023:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=4, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar  1 00:39:05.023:rtpspi_do_call_modify:RTCP Timer start.
*Mar  1 00:39:05.023:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar  1 00:39:05.023:rtpspi_do_call_modify:success. leaving
*Mar  1 00:40:13.786:rtpspi_bridge_drop:entered. src call-id=5, dest call-id=4, tag=0
*Mar  1 00:40:13.786:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar  1 00:40:13.786:rtpspi_modify_rtcp_session_parameters():xmit fn=0x0,
dstIF=0x0, dstCallID=0, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar  1 00:40:13.786:rtpspi_bridge_drop:leaving
*Mar  1 00:40:13.790:rtpspi_call_disconnect:entered. call-id=5, cause=16, tag=0
*Mar  1 00:40:13.790:rtpspi_call_disconnect:leaving.
*Mar  1 00:40:13.790:rtpspi_do_call_disconnect:Entered. call-id = 5
*Mar  1 00:40:13.790:rtpspi_do_call_disconnect:calling rtpspi_call_cleanup(). call-id=5
*Mar  1 00:40:13.794:rtpspi_call_cleanup:entered. ccb = 0x1AF5400, call-id=5, rtp port =
16544
*Mar  1 00:40:13.794:rtpspi_call_cleanup:releasing ccb cache. RTP port=16544
*Mar  1 00:40:13.794:rtpspi_store_call_history_entry():Entered.
*Mar  1 00:40:13.794:rtpspi_store_call_history_entry():Leaving.
*Mar  1 00:40:13.794:rtpspi_call_cleanup:RTCP Timer Stop.
*Mar  1 00:40:13.794:rtpspi_call_cleanup:deallocating RTP port 16544.
*Mar  1 00:40:13.794:rtpspi_free_rtcp_session:Entered.
*Mar  1 00:40:13.794:rtpspi_free_rtcp_session:Success. Leaving
*Mar  1 00:40:13.794::rtpspi_call_cleanup freeing ccb (0x1AF5400)
*Mar  1 00:40:13.794:rtpspi_call_cleanup:leaving
*Mar  1 00:40:13.794:rtpspi_do_call_disconnect:leaving
```

Related Commands

Command	Description
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug rtpspi errors

To debug RTP SPI errors, use the **debug rtpspi errors** EXEC command. Use the **no debug rtpspi errors** command to turn off debugging.

debug rtpspi errors

no debug rtpspi errors

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620) in a private release that was not generally available.

Usage Guidelines

Caution Be careful when you use this command because it can result in console flooding and reduced voice quality.

Examples

This example shows a debug trace for RTP SPI errors on two gateways. The following example shows the debug trace on the first gateway:

```
router# debug rtpspi errors

00:54:13.272:rtpspi_do_call_modify:new mode 2 is the same as the current mode
00:54:18.738:rtpspi_do_call_modify:New remote RTP port = old rtp port = 16452
00:54:18.738:rtpspi_do_call_modify:New remote IP addrs = old IP addrs = 0x6000001
```

The following example shows the debug trace on the second gateway:

```
router# debug rtpspi errors

00:54:08:rtpspi_process_timers:
00:54:08:rtpspi_process_timers:Timer 0x1A5AF9C expired.
00:54:08:rtpspi_process_timers:Timer expired for callID 0x3
00:54:08:rtpspi_process_timers:
00:54:08:rtpspi_process_timers:Timer 0x1A5AF9C expired.
00:54:08:rtpspi_process_timers:Timer expired for callID 0x3
00:54:08:rtpspi_process_timers:
00:54:08:rtpspi_process_timers:Timer 0x1A5AF9C expired.
```

```
00:54:08:rtpspi_process_timers:Timer expired for callID 0x3
00:54:09:rtpspi_process_timers:
00:54:09:rtpspi_process_timers:Timer 0x1A5AFBC expired.
00:54:09:rtpspi_process_timers:Timer expired for callID 0x3
00:54:09:rtpspi_process_timers:
00:54:09:rtpspi_process_timers:Timer 0x1A5B364 expired.
00:54:09:rtpspi_process_timers:Timer expired for callID 0x3
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug rtpspi inout

To debug RTP SPI in/out functions, use the **debug rtpspi inout** EXEC command. Use the **no debug rtpspi inout** command to turn off debugging.

debug rtpspi inout

no debug rtpspi inout

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620 device) in a private release that was not generally available.

Usage Guidelines

Caution Be careful when you use this command because it can result in console flooding and reduced voice quality.

Examples

The following example shows a debug trace for RTP SPI in/out functions on a gateway:

```
router# debug rtpspi inout

*Mar  1 00:57:24.565:rtpspi_allocate_rtp_port:Entered.
*Mar  1 00:57:24.565:rtpspi_allocate_rtp_port:Success. port = 16520. Leaving.
*Mar  1 00:57:24.565:rtpspi_call_setup_request:entered.
        Call Id = 9, dest = 0.0.0.0;   callInfo:
        final dest flag = 0,
        rtp_session_mode = 0x2,
        local_ip_addrs = 0x5000001,remote_ip_addrs = 0x0,
        local rtp port = 16520, remote rtp port = 0
*Mar  1 00:57:24.565:rtpspi_call_setup_request:spi_info copied for rtpspi_app_data_t.
*Mar  1 00:57:24.565:rtpspi_call_setup_request:leaving
*Mar  1 00:57:24.569:rtpspi_call_setup() entered
*Mar  1 00:57:24.569:rtpspi_initialize_ccb:Entered
*Mar  1 00:57:24.569:rtpspi_initialize_ccb:leaving
*Mar  1 00:57:24.569:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem rtp=0,
rem ip=0x0
*Mar  1 00:57:24.569:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar  1 00:57:24.569:rtpspi_call_setup:Leaving.
```

```
*Mar  1 00:57:24.573:rtpspi_bridge:entered. conf id = 3, src i/f = 0x1859E88,
        dest i/f = 0x1964EEC, src call id = 9, dest call id = 8
        call info = 0x1919140, xmit fn = 0xDA7494, tag = 0
*Mar  1 00:57:24.573:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar  1 00:57:24.573:rtpspi_bridge:leaving.
*Mar  1 00:57:24.573:rtpspi_caps_ind:Entered. vdb = 0x1859E88 call id = 9, srcCallId = 8
*Mar  1 00:57:24.577:rtpspi_caps_ind:Returning success
*Mar  1 00:57:24.577:rtpspi_caps_ack:Entered. call id = 9, srcCallId = 8
*Mar  1 00:57:24.577:rtpspi_caps_ack:leaving.
*Mar  1 00:57:24.818:rtpspi_call_modify:entered. call-id=9, nominator=0x7,
params=0x18DD440
*Mar  1 00:57:24.818:rtpspi_call_modify:leaving
*Mar  1 00:57:24.818:rtpspi_do_call_modify:Entered. call-id = 9
*Mar  1 00:57:24.818:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem
rtp=16396, rem ip=0x6000001
*Mar  1 00:57:24.822:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar  1 00:57:24.822:rtpspi_do_call_modify:success. leaving
*Mar  1 00:57:30.296:rtpspi_call_modify:entered. call-id=9, nominator=0x7,
params=0x18DD440
*Mar  1 00:57:30.296:rtpspi_call_modify:leaving
*Mar  1 00:57:30.300:rtpspi_do_call_modify:Entered. call-id = 9
*Mar  1 00:57:30.300:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar  1 00:57:30.300:rtpspi_do_call_modify:success. leaving
*Mar  1 00:58:39.055:rtpspi_bridge_drop:entered. src call-id=9, dest call-id=8, tag=0
*Mar  1 00:58:39.055:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar  1 00:58:39.055:rtpspi_bridge_drop:leaving
*Mar  1 00:58:39.059:rtpspi_call_disconnect:entered. call-id=9, cause=16, tag=0
*Mar  1 00:58:39.059:rtpspi_call_disconnect:leaving.
*Mar  1 00:58:39.059:rtpspi_do_call_disconnect:Entered. call-id = 9
*Mar  1 00:58:39.059:rtpspi_call_cleanup:entered. ccb = 0x1AF5400, call-id=9, rtp port =
16520
*Mar  1 00:58:39.059:rtpspi_store_call_history_entry():Entered.
*Mar  1 00:58:39.059:rtpspi_store_call_history_entry():Leaving.
*Mar  1 00:58:39.059:rtpspi_free_rtcp_session:Entered.
*Mar  1 00:58:39.059:rtpspi_free_rtcp_session:Success. Leaving
*Mar  1 00:58:39.063:rtpspi_call_cleanup:leaving
*Mar  1 00:58:39.063:rtpspi_do_call_disconnect:leaving
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug rtpspi send-nse

To trigger the RTP SPI software module to send a triple redundant NSE, use the **debug rtpspi send-nse** EXEC command. Use the **no debug rtpspi send-nse** to disable this action.

debug rtpspi send-nse *call-ID NSE-event-ID*

no debug rtpspi send-nse *call-ID NSE-event-ID*

Syntax Description

call-ID	Specifies the call ID of the active call. The valid range is from 0 to 65535.
NSE-event-ID	Specifies the NSE Event ID. The valid range is from 0 to 255.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620 router) in a private release that was not generally available.

Examples

The following example shows the RTP SPI software module set to send an NSE:

```
router# debug rtpspi send-nse
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug rtpspi session

To debug all RTP SPI sessions, use the **debug rtpspi session** EXEC command. Use the **no debug rtpspi session** command to turn off debugging.

debug rtpspi session

no debug rtpspi session

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620 router) in a private release that was not generally available.

Examples

The following example shows a debug trace for RTP SPI sessions on a gateway:

```
router# debug rtpspi session

*Mar  1 01:01:51.593:rtpspi_allocate_rtp_port:allocated RTP port 16406
*Mar  1 01:01:51.593:rtpspi_call_setup:rtp_session_mode = 0x2
*Mar  1 01:01:51.593:rtpspi_call_setup:mode = CC_CALL_NORMAL.
        destianation number = 0.0.0.0
*Mar  1 01:01:51.593:rtpspi_call_setup:Passed local_ip_addrs=0x5000001
*Mar  1 01:01:51.593:rtpspi_call_setup:Passed local_rtp_port = 16406
*Mar  1 01:01:51.593:rtpspi_call_setup:Saved RTCP Session = 0x1AFDFBC
*Mar  1 01:01:51.593:rtpspi_call_setup:Passed remote rtp port = 0.
*Mar  1 01:01:51.598:rtpspi_start_rtcp_session:Starting RTCP session.
        Local IP addr = 0x5000001, Remote IP addr = 0x0,
        Local RTP port = 16406, Remote RTP port = 0, mode = 0x2
*Mar  1 01:01:51.598:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar  1 01:01:51.598:rtpspi_call_setup:RTP Session creation Success.
*Mar  1 01:01:51.598:rtpspi_call_setup:calling cc_api_call_connected()
*Mar  1 01:01:51.598:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=10, voip_mode=0x2, rtp_mode=0x2, ssrc_status=0
*Mar  1 01:01:51.598:rtpspi_bridge:Calling cc_api_bridge_done() for 11(0x1AF5400) and
10(0x0).
*Mar  1 01:01:51.602:rtpspi_caps_ind:caps from VTSP:codec=0x83FB, codec_bytes=0x50,
                    fax rate=0x7F, vad=0x3 modem=0x0
*Mar  1 01:01:51.602:rtpspi_get_rtcp_session_parameters():CURRENT VALUES:
dstIF=0x1964EEC, dstCallID=10, current_seq_num=0x0
*Mar  1 01:01:51.602:rtpspi_get_rtcp_session_parameters():NEW VALUES:
dstIF=0x1964EEC, dstCallID=10, current_seq_num=0xF1E
*Mar  1 01:01:51.602:rtpspi_caps_ind:Caps Used:codec=0x1, codec bytes=80,
        fax rate=0x1, vad=0x1, modem=0x1, dtmf_relay=0x1, seq_num_start=0xF1F
```

```
*Mar  1 01:01:51.602:rtpspi_caps_ind:calling cc_api_caps_ind().
*Mar  1 01:01:51.822:rtpspi_do_call_modify:Remote RTP port changed. New port=16498
*Mar  1 01:01:51.822:rtpspi_do_call_modify:Remote IP addrs changed. New IP addrs=0x6000001
*Mar  1 01:01:51.822:rtpspi_do_call_modify:Starting new RTCP session.
*Mar  1 01:01:51.822:rtpspi_start_rtcp_session:Removing old RTCP session.
*Mar  1 01:01:51.822:rtpspi_start_rtcp_session:Starting RTCP session.
        Local IP addr = 0x5000001, Remote IP addr = 0x6000001,
        Local RTP port = 16406, Remote RTP port = 16498, mode = 0x2
*Mar  1 01:01:51.822:rtpspi_start_rtcp_session:RTCP Timer creation Success. (5)*(5000)
*Mar  1 01:01:51.826:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar  1 01:01:51.826:rtpspi_do_call_modify:RTP Session creation Success.
*Mar  1 01:01:51.826:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar  1 01:01:57.296:rtpspi_do_call_modify:Mode changed. new = 3, old = 2
*Mar  1 01:01:57.296:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=10, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar  1 01:01:57.296:rtpspi_do_call_modify:RTCP Timer start.
*Mar  1 01:01:57.296:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar  1 01:03:06.108:rtpspi_modify_rtcp_session_parameters():xmit fn=0x0,
dstIF=0x0, dstCallID=0, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar  1 01:03:06.112:rtpspi_do_call_disconnect:calling rtpspi_call_cleanup(). call-id=11
*Mar  1 01:03:06.112:rtpspi_call_cleanup:releasing ccb cache. RTP port=16406
*Mar  1 01:03:06.112:rtpspi_call_cleanup:RTCP Timer Stop.
*Mar  1 01:03:06.112:rtpspi_call_cleanup:deallocating RTP port 16406.
*Mar  1 01:03:06.112::rtpspi_call_cleanup freeing ccb (0x1AF5400)
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.
sgcp	Starts and allocates resources for the SCGP daemon.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug sdlc

To display information on Synchronous Data Link Control (SDLC) frames received and sent by any router serial interface involved in supporting SDLC end station functions, use the **debug sdlc** privileged EXEC command. The **no** form of this command disables debugging output.

debug sdlc

no debug sdlc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Note Because the **debug sdlc** command can generate many messages and alter timing in the network node, use it only when instructed by authorized support personnel.

Examples

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

```
Router# debug sdlc

SDLC: Sending RR at location 4
Serial3: SDLC O (12495952) C2 CONNECT (2) RR P/F 6
Serial3: SDLC I (12495964) [C2] CONNECT (2) RR P/F 0 (R) [VR: 6 VS: 0]
Serial3: SDLC T [C2] 12496064 CONNECT 12496064 0
SDLC: Sending RR at location 4
Serial3: SDLC O (12496064) C2 CONNECT (2) RR P/F 6
Serial3: SDLC I (12496076) [C2] CONNECT (2) RR P/F 0 (R) [VR: 6 VS: 0]
Serial3: SDLC T [C2] 12496176 CONNECT 12496176 0
```

The following line of output indicates that the router is sending a Receiver Ready packet at location 4 in the code:

```
SDLC: Sending RR at location 4
```

The following line of output describes a frame output event:

```
Serial1/0: SDLC O 04 CONNECT (285) IFRAME P/F 6
```

Table 162 describes the significant fields shown in the display.

Table 162 debug sdlc Field Descriptions for a Frame Output Event

Field	Description
Serial1/0	Interface type and unit number reporting the frame event.
SDLC	Protocol providing the information.

Table 162 debug sdlc Field Descriptions for a Frame Output Event (continued)

Field	Description
O	Command mode of frame event. Possible values are as follows: • I—Frame input • O—Frame output • T—T1 timer expired
04	SDLC address of the SDLC connection.
CONNECT	State of the protocol when the frame event occurred. Possible values are as follows: • CONNECT • DISCONNECT • DISCSENT (disconnect sent) • ERROR (FRMR frame sent) • REJSENT (reject frame sent) • SNRMSENT (SNRM frame sent) • USBUSY • THEMBUSY • BOTHBUSY
(285)	Size of the frame (in bytes).
IFRAME	Frame type name. Possible values are as follows: • DISC—Disconnect • DM—Disconnect mode • FRMR—Frame reject • IFRAME—Information frame • REJ—Reject • RNR—Receiver not ready • RR—Receiver ready • SIM—Set Initialization mode command • SNRM—Set Normal Response Mode • TEST—Test frame • UA—Unnumbered acknowledgment • XID—EXchange ID
P/F	Poll/Final bit indicator. Possible values are as follows: • F—Final (printed for Response frames) • P—Poll (printed for Command frames) • P/F—Poll/Final (printed for RR, RNR, and REJ frames, which can be either Command or Response frames)
6	Receive count; range: 0 to 7.

The following line of output describes a frame input event:

```
Serial1/0: SDLC I 02 CONNECT (16) IFRAME P 7 0,[VR: 7 VS: 0]
```

Table 163 describes the significant fields shown in the display.

Table 163 debug sdlc Field Descriptions for a Frame Input Event

Field	Description
02	SDLC address.
IFRAME	Traffic engineering type.
P	Poll bit P is on.
VR: 7	Receive count; range: 0 to 7.
VS: 0	Send count; range: 0 to 7.

The following line of output describes a frame timer event:

```
Serial1/0: SDLC T 02 CONNECT 0x9CB69E8 P 0
```

Table 164 describes the significant fields shown in the display.

Table 164 debug sdlc Field Descriptions for a Timer Event

Field	Description
Serial1/0	Interface type and unit number reporting the frame event.
SDLC	Protocol providing the information.
T	Timer has expired.
02	SDLC address of this SDLC connection.
CONNECT	State of the protocol when the frame event occurred. Possible values are as follows: • BOTHBUSY • CONNECT • DISCONNECT • DISCSENT (disconnect sent) • ERROR (FRMR frame sent) • REJSENT (reject frame sent) • SNRMSENT (SNRM frame sent) • THEMBUSY • BOTHBUSY
0x9CB69E8	System clock.
0	Retry count; default: 0.

Related Commands

Command	Description
debug list	Filters debugging information on a per-interface or per-access list basis.

debug sdlc local-ack

To display information on the local acknowledgment feature, use the **debug sdlc local-ack** privileged EXEC command. The **no** form of this command disables debugging output.

debug sdlc local-ack [*number*]

no debug sdlc local-ack [*number*]

Syntax Description

number	(Optional) Frame-type that you want to monitor. See the "Usage Guidelines" section.

Usage Guidelines

You can select the frame types you want to monitor; the frame types correspond to bit flags. You can select 1, 2, 4, or 7, which is the decimal value of the bit flag settings. If you select 1, the octet is set to 00000001. If you select 2, the octet is set to 0000010. If you select 4, the octet is set to 00000100. If you want to select all frame types, select 7; the octet is 00000111. The default is 7 for all events. Table 165 defines these bit flags.

Table 165 debug sdlc local-ack Debugging Levels

Debug Command	Meaning
debug sdlc local-ack 1	Only U-Frame events
debug sdlc local-ack 2	Only I-Frame events
debug sdlc local-ack 4	Only S-Frame events
debug sdlc local-ack 7	All SDLC Local-Ack events (default setting)

Caution Because using this command is processor intensive, it is best to use it after hours, rather than in a production environment. It is also best to use this command by itself, rather than in conjunction with other debugging commands.

Examples

The following is sample output from the **debug sdlc local-ack** command:

```
router# debug sdlc local-ack 1
```

Group of associated operations

```
SLACK (Serial3): Input     = Network, LinkupRequest
SLACK (Serial3): Old State = AwaitSdlcOpen          New State = AwaitSdlcOpen

SLACK (Serial3): Output    = SDLC, SNRM

SLACK (Serial3): Input     = SDLC, UA
SLACK (Serial3): Old State = AwaitSdlcOpen          New State = Active

SLACK (Serial3): Output    = Network, LinkResponse
```

S2560

The first line shows the input to the SDLC local acknowledgment state machine:

```
SLACK (Serial3): Input     = Network, LinkupRequest
```

Table 166 describes the significant fields shown in the display.

Table 166 debug sdlc local-ack Field Descriptions

Field	Description
SLACK	SDLC local acknowledgment feature is providing the information.
(Serial3):	Interface type and unit number reporting the event.
Input = Network	Source of the input.
LinkupRequest	Op code. A LinkupRequest is an example of possible values.

The second line shows the change in the SDLC local acknowledgment state machine. In this case the AwaitSdlcOpen state is an internal state that has not changed while this display was captured.

```
SLACK (Serial3): Old State = AwaitSdlcOpen           New State = AwaitSdlcOpen
```

The third line shows the output from the SDLC local acknowledgment state machine:

```
SLACK (Serial3): Output     = SDLC, SNRM
```

debug sdlc packet

To display packet information on Synchronous Data Link Control (SDLC) frames received and sent by any router serial interface involved in supporting SDLC end station functions, use the **debug sdlc packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug sdlc packet [*max-bytes*]

no debug sdlc packet [*max-bytes*]

Syntax Description

max-bytes	(Optional) Limits the number of bytes of data that are printed to the display.

Usage Guidelines

This command requires intensive CPU processing; therefore, we recommend not using it when the router is expected to handle normal network loads, such as in a production environment. Instead, use this command when network response is noncritical. We also recommend that you use this command by itself, rather than in conjunction with other **debug** commands.

Examples

The following is sample output from the **debug sdlc packet** command with the packet display limited to 20 bytes of data:

```
Router# debug sdlc packet 20

 Serial3 SDLC Output
00000 C3842C00 02010010 019000C5 C5C5C5C5 Cd.........EEEEE
00010 C5C5C5C5                            EEEE
 Serial3 SDLC Output
00000 C3962C00 02010011 039020F2          Co.........2
 Serial3 SDLC Output
00000 C4962C00 0201000C 039020F2          Do.........2
 Serial3 SDLC Input
00000     C491                              Dj
```

debug sdllc

To display information about data link-layer frames transferred between a device on a Token Ring and a device on a serial line via a router configured with the SDLLC feature, use the **debug sdllc** privileged EXEC command. The **no** form of this command disables debugging output.

debug sdllc

no debug sdllc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The SDLLC feature translates between the SDLC link-layer protocol used to communicate with devices on a serial line and the LLC2 link-layer protocol used to communicate with devices on a Token Ring.

The router configured with the SDLLC feature must be attached to the serial line. The router sends and receives frames on behalf of the serial device on the attached serial line but acts as an SDLC station.

The topology between the router configured with the SDLLC feature and the Token Ring is network dependent and is not limited by the SDLLC feature.

Examples

The following is sample output from the **debug sdllc** command between link-layer peers from the perspective of the SDLLC-configured router:

```
Router# debug sdllc

SDLLC: rx explorer rsp, da 4000.2000.1001, sa C000.1020.1000, rif
 8840.0011.00A1.0050
SDLLC: tx short xid, sa 4000.2000.1001, da C000.1020.1000, rif
 88C0.0011.00A1.0050, dsap 4 ssap 4
SDLLC: tx long xid, sa 4000.2000.1001, da C000.1020.1000, rif
 88C0.0011.00A1.0050, dsap 4 ssap 4
Rcvd SABME/LINKUP_REQ pak from TR host
SDLLCERR: not from our partner, pak dropped, da 4000.2000.1001,
sa C000.1020.1000, rif 8840.0011.00A1.0050, partner = 5000.1040.1003
```

Table 167 describes the significant fields shown in the display.

Table 167 debug sdllc Field Descriptions

Field	Description
rx	Router receives message from the FEP.
explorer rsp	Response to an explorer (TEST) frame previously sent by the router to the FEP.
da	Destination address. This is the address of the router receiving the response.
sa	Source address. This is the address of the FEP sending the response to the router.
rif	Routing information field (RIF).
tx	Router sent message to the FEP.

Table 167 debug sdllc Field Descriptions (continued)

Field	Description
short xid	Router sent the null XID to the FEP.
dsap	Destination service access point
ssap	Source service access point.
tx long xid	Router sent the XID type 2 to the FEP.
Rcvd	Router received Layer 2 message from the FEP.
SABME/LINKUP_REQ	A set asynchronous Balanced Mode Extended command.
partner =	Partner address.

The following line indicates that an explorer frame response was received by the router at address 4000.2000.1001 from the FEP at address C000.1020.1000 with the specified RIF. The original explorer sent to the FEP from the router is not monitored as part of the **debug sdllc** command.

```
SDLLC: rx explorer rsp, da 4000.2000.1001, sa C000.1020.1000, rif
 8840.0011.00A1.0050
```

The following line indicates that the router sent the null XID (Type 0) to the FEP. The debugging information does not include the response to the XID message sent by the FEP to the router.

```
SDLLC: tx short xid, sa 4000.2000.1001, da C000.1020.1000, rif
 88C0.0011.00A1.0050, dsap 4 ssap 4
```

The following line indicates that the router sent the XID command (Format 0 Type 2) to the FEP:

```
SDLLC: tx long xid, sa 4000.2000.1001, da C000.1020.1000, rif
 88C0.0011.00A1.0050, dsap 4 ssap 4
```

The following line is the SABME response to the XID command previously sent by the router to the FEP:

```
Rcvd SABME/LINKUP_REQ pak from TR host
```

debug serial interface

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

debug serial interface

no debug serial interface

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

If the **show interface serial** EXEC command shows that the line and protocol are down, you can use the **debug serial interface** command to isolate a timing problem as the cause of a connection failure. If the keepalive values in the mineseq, yourseen, and myseen fields are not incrementing in each subsequent line of output, there is a timing or line problem at one end of the connection.

Caution Although the **debug serial interface** command typically does not generate a substantial amount of output, nevertheless use it cautiously during production hours. When SMDS is enabled, for example, it can generate considerable output.

The output of the **debug serial interface** command can vary, depending on the type of WAN configured for an interface: Frame Relay, HDLC, HSSI, SMDS, or X.25. The output also can vary depending on the type of encapsulation configured for that interface. The hardware platform also can affect **debug serial interface** output.

Examples

The following sections show and describe sample **debug serial interface** output for various configurations.

Debug Serial Interface for Frame Relay Encapsulation

The following message is displayed if the encapsulation for the interface is Frame Relay (or HDLC) and the router attempts to send a packet containing an unknown packet type:

```
Illegal serial link type code xxx
```

Debug Serial Interface for HDLC

The following is sample output from the **debug serial interface** command for an HDLC connection when keepalives are enabled. This output shows that the remote router is not receiving all the keepalives the router is sending. When the difference in the values in the myseq and mineseen fields exceeds three, the line goes down and the interface is reset.

```
router# debug serial interface

Serial1: HDLC myseq 636119, mineseen 636119, yourseen 515032, line up
Serial1: HDLC myseq 636120, mineseen 636120, yourseen 515033, line up
Serial1: HDLC myseq 636121, mineseen 636121, yourseen 515034, line up
Serial1: HDLC myseq 636122, mineseen 636122, yourseen 515035, line up
Serial1: HDLC myseq 636123, mineseen 636123, yourseen 515036, line up
Serial1: HDLC myseq 636124, mineseen 636124, yourseen 515037, line up
Serial1: HDLC myseq 636125, mineseen 636125, yourseen 515038, line up
Serial1: HDLC myseq 636126, mineseen 636126, yourseen 515039, line up

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

Serial1: HDLC myseq 636130, mineseen 636130, yourseen 515043, line up
Serial1: HDLC myseq 636131, mineseen 636130, yourseen 515044, line up
Serial1: HDLC myseq 636132, mineseen 636130, yourseen 515045, line up
Serial1: HDLC myseq 636133, mineseen 636130, yourseen 515046, line down
Serial1: HDLC myseq 636127, mineseen 636127, yourseen 515040, line up
Serial1: HDLC myseq 636128, mineseen 636127, yourseen 515041, line up
Serial1: HDLC myseq 636129, mineseen 636129, yourseen 515042, line up
```

1 missed keepalive

3 missed keepalives; line goes down and interface is reset

S2561

Table 168 describes the significant fields.

Table 168 debug serial interface Field Descriptions for HDLC

Field	Description
Serial 1	Interface through which the serial connection is taking place.
HDLC	Serial connection is an HDLC connection.
myseq 636119	Myseq counter increases by one each time the router sends a keepalive packet to the remote router.
mineseen 636119	Value of the mineseen counter reflects the last myseq sequence number the remote router has acknowledged receiving from the router. The remote router stores this value in its yourseen counter and sends that value in a keepalive packet to the router.
yourseen 515032	Yourseen counter reflects the value of the myseq sequence number the router has received in a keepalive packet from the remote router.
line up	Connection between the routers is maintained. Value changes to "line down" if the values of the myseq and myseen fields in a keepalive packet differ by more than three. Value returns to "line up" when the interface is reset. If the line is in loopback mode, ("looped") appears after this field.

The previous example shows that after missing three keepalives, the line goes down and the interface is reset. However, the interface is also reset when two keepalives are missed, but the line is not marked as "down." This is done in an attempt to restart traffic on the interface without bringing the line down, as shown in the following output:

```
*Mar 18 08:07:29.057: Serial3/2: HDLC myseq 604562, mineseen 604562, yourseen 259336, line up
*Mar 18 08:07:39.053: Serial3/2: HDLC myseq 604563, mineseen 604563, yourseen 259337, line up
*Mar 18 08:07:49.081: Serial3/2: HDLC myseq 604564, mineseen 604564, yourseen 259338, line up
*Mar 18 08:07:59.057: Serial3/2: HDLC myseq 604565, mineseen 604565, yourseen 259339, line up
*Mar 18 08:08:09.073: Serial3/2: HDLC myseq 604566, mineseen 604565, yourseen 259340, line up
*Mar 18 08:08:19.057: Serial3/2: Reset from PC 0x6DEA0
*Mar 18 08:08:19.061: Serial3/2: HDLC myseq 604567, mineseen 604565, yourseen 259341, line up
*Mar 18 08:08:29.057: Serial3/2: HDLC myseq 604568, mineseen 604568, yourseen 259342, line up
*Mar 18 08:08:39.061: Serial3/2: HDLC myseq 604569, mineseen 604569, yourseen 259343, line up
*Mar 18 08:08:49.065: Serial3/2: HDLC myseq 604570, mineseen 604570, yourseen 259344, line up
*Mar 18 08:08:59.053: Serial3/2: HDLC myseq 604571, mineseen 604571, yourseen 259345, line up
```

Even though the "Reset from PC" message appears to occur when there is only a difference of 1 between myseq and mineseen, this message applies to the condition shown in the immediately following line (notice that the timestamp is only a few milliseconds later) where the difference is 2. After the reset, the line has recovered and the difference between myseq and mineseen is zero.

Table 169 describes additional error messages that the **debug serial interface** command can generate for HDLC.

Table 169 debug serial interface Error Messages for HDLC

Field	Description
Illegal serial link type code <*xxx*>, PC = 0x*nnnnnn*	Router attempted to send a packet containing an unknown packet type.
Illegal HDLC serial type code <*xxx*>, PC = 0x*nnnnn*	Unknown packet type is received.
Serial 0: attempting to restart	Interface is down. The hardware is then reset to correct the problem, if possible
Serial 0: Received bridge packet sent to <*nnnnnnnnn*>	Bridge packet is received over a serial interface configured for HDLC, and bridging is not configured on that interface.

Debug Serial Interface for HSSI

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

```
HSSI0: Reset from 0xnnnnnnn
```

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

Debug Serial Interface for ISDN Basic Rate

Table 170 describes error messages that the **debug serial interface** command can generate for ISDN Basic Rate.

Table 170 debug serial interface Error Messages for ISDN Basic Rate

Message	Description
BRI: D-chan collision	Collision on the ISDN D channel has occurred; the software will retry transmission.
Received SID Loss of Frame Alignment int.	ISDN hardware has lost frame alignment. This usually indicates a problem with the ISDN network.
Unexpected IMP int: ipr = 0x*nn*	ISDN hardware received an unexpected interrupt. The 0x*nn* variable indicates the value returned by the interrupt register.
BRI(d): RX Frame Length Violation. Length=*n* BRI(d): RX Nonoctet Aligned Frame BRI(d): RX Abort Sequence BRI(d): RX CRC Error BRI(d): RX Overrun Error BRI(d): RX Carrier Detect Lost	Any of these messages can be displayed when a receive error occurs on one of the ISDN channels. The (d) indicates which channel it is on. These messages can indicate a problem with the ISDN network connection.
BRI0: Reset from 0x*nnnnnnn*	BRI hardware has been reset. The 0x*nnnnnnn* variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers.
BRI(d): Bad state in SCMs scm1=*x* scm2=*x* scm3=*x* BRI(d): Bad state in SCONs scon1=*x* scon2 =*x* scon3=*x* BRI(d): Bad state ub SCR; SCR=*x*	Any of these messages can be displayed if the ISDN hardware is not in the proper state. The hardware is then reset. If the message is displayed constantly, it usually indicates a hardware problem.
BRI(d): Illegal packet encapsulation=*n*	Packet is received, but the encapsulation used for the packet is not recognized. The interface might be misconfigured.

Debug Serial Interface for an MK5025 Device

Table 171 describes the additional error messages that the **debug serial interface** command can generate for an MK5025 device.

Table 171 debug serial interface Error Messages for an MK5025 Device

Message	Description
MK5(d): Reset from 0x*nnnnnnnn*	Hardware has been reset. The 0x*nnnnnnn* variable is the address of the routine that requested that the hardware be reset; it is useful only to development engineers.
MK5(d): Illegal packet encapsulation=*n*	Packet is received, but the encapsulation used for the packet is not recognized. Interface might be misconfigured.
MK5(d): No packet available for packet realignment	Serial driver attempted to get a buffer (memory) and was unable to do so.
MK5(d): Bad state in CSR0=(*x*)	This message is displayed if the hardware is not in the proper state. The hardware is reset. If this message is displayed constantly, it usually indicates a hardware problem.

Table 171 debug serial interface Error Messages for an MK5025 Device (continued)

Message	Description
MK5(d): New serial state=*n*	Hardware has interrupted the software. It displays the state that the hardware is reporting.
MK5(d): DCD is down. MK5(d): DCD is up.	If the interrupt indicates that the state of carrier has changed, one of these messages is displayed to indicate the current state of DCD.

Debug Serial Interface for SMDS Encapsulation

When encapsulation is set to SMDS, the **debug serial interface** command displays SMDS packets that are sent and received, and any error messages resulting from SMDS packet transmission.

The error messages that the **debug serial interface** command can generate for SMDS follow.

The following message indicates that a new protocol requested SMDS to encapsulate the data for transmission. SMDS is not yet able to encapsulate the protocol.

```
SMDS: Error on Serial 0, encapsulation bad protocol = x
```

The following message indicates that SMDS was asked to encapsulate a packet, but no corresponding destination E.164 SMDS address was found in any of the static SMDS tables or in the ARP tables:

```
SMDS send: Error in encapsulation, no hardware address, type = x
```

The following message indicates that a protocol such as CLNS or IP has been enabled on an SMDS interface, but the corresponding multicast addresses have not been configured. The *n* variable displays the link type for which encapsulation was requested.

```
SMDS: Send, Error in encapsulation, type=n
```

The following messages can occur when a corrupted packet is received on an SMDS interface. The router expected *x*, but received *y*.

```
SMDS: Invalid packet, Reserved NOT ZERO, x y
SMDS: Invalid packet, TAG mismatch x y
SMDS: Invalid packet, Bad TRAILER length x y
```

The following messages can indicate an invalid length for an SMDS packet:

```
SMDS: Invalid packet, Bad BA length x
SMDS: Invalid packet, Bad header extension length x
SMDS: Invalid packet, Bad header extension type x
SMDS: Invalid packet, Bad header extension value x
```

The following messages are displayed when the **debug serial interface** command is enabled:

```
Interface Serial 0 Sending SMDS L3 packet:
SMDS: dgsize:x type:0xn src:y dst:z
```

If the **debug serial interface** command is enabled, the following message can be displayed when a packet is received on an SMDS interface, but the destination SMDS address does not match any on that interface:

```
SMDS: Packet n, not addressed to us
```

debug serial packet

To display more detailed serial interface debugging information than you can obtain using the **debug serial interface** command, use the **debug serial packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug serial packet

no debug serial packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug serial packet** command generates output that is dependent on the type of serial interface and the encapsulation running on that interface. The hardware platform also can impact **debug serial packet** output.

The **debug serial packet** command displays output for only SMDS encapsulations.

Examples

The following is sample output from the **debug serial packet** command when SMDS is enabled on the interface:

```
Router# debug serial packet

Interface Serial2 Sending SMDS L3 packet:
SMDS Header: Id: 00 RSVD: 00 BEtag: EC Basize: 0044
Dest:E18009999999FFFF Src:C12015804721FFFF Xh:04030000030001000000000000000000
SMDS LLC: AA AA 03 00 00 00 80 38
SMDS Data: E1 19 01 00 00 80 00 00 0C 00 38 1F 00 0A 00 80 00 00 0C 01 2B 71
SMDS Data: 06 01 01 0F 1E 24 00 EC 00 44 00 02 00 00 83 6C 7D 00 00 00 00 00
SMDS Trailer: RSVD: 00 BEtag: EC Length: 0044
```

As the output shows, when encapsulation is set to SMDS, the **debug serial packet** command displays the entire SMDS header (in hexadecimal notation), and some payload data on transmit or receive. This information is useful only when you have an understanding of the SMDS protocol. The first line of the output indicates either Sending or Receiving.

debug service-module

To display debugging information that monitors the detection and clearing of network alarms on the integrated channel service unit/data service unit (CSU/DSU) modules, use the **debug service-module** privileged EXEC command. The **no** form of this command disables debugging output.

debug service-module

no debug service-module

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to enable and disable debug logging for the serial 0 and serial 1 interfaces when an integrated CSU/DSU is pre-sent. This command enables debugging on all interfaces.

Network alarm status can also be viewed through the use of the **show service-module** command.

Note The debug output varies depending on the type of service module installed in the router.

Examples

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

```
Router# debug service-module

SERVICE_MODULE(1): loss of signal ended after duration 00:05:36
SERVICE_MODULE(1): oos/oof ended after duration 01:05:14
SERVICE_MODULE(0): Unit has no clock
SERVICE_MODULE(0): detects loss of signal
SERVICE_MODULE(0): loss of signal ended after duration 00:00:33
```

debug sgbp dial-bids

To display large-scale dial-out negotiations between the primary network access server (NAS) and alternate NASs, use the **debug sgbp dial-bids** privileged EXEC command. The **no** form of this command disables debugging output.

debug sgbp dial-bids

no debug sgbp dial-bids

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command only when the **sgbp dial-bids** command has been configured.

Examples

The following is sample output from the **debug sgbp dial-bids** command:

```
Router# debug sgbp dial-bids

*Jan  1 00:25:03.643: SGBP-RES:  New bid add request: 4B0 8 2 1 DAC0 1 1
This indicates a new dialout bid has started.
*Jan  1 00:25:03.643: SGBP-RES: Sent Discover message to ID 7B09B71E  49 bytes
The bid request has been sent.
*Jan  1 00:25:03.647: SGBP-RES: Received Message of 49 length:

*Jan  1 00:25:03.647: SGBP-RES: header 5  30  0  31
2  0  0  2D  0  0  0  0  0  0  0  3  0  0  0  1  1E  AF  3A  41  7B  9  B7  1E  8  15  B
3  2  C  6  0  0  DA  C0  D  4  0  0  E  3  1  F  3  1
*Jan  1 00:25:03.647:
*Jan  1 00:25:03.647: SGBP RES: Scan: Message type: Offer
*Jan  1 00:25:03.647: SGBP RES: Scan: Len is 45
*Jan  1 00:25:03.647: SGBP RES: Scan: Transaction ID: 3
*Jan  1 00:25:03.647: SGBP RES: Scan: Message ID: 1
*Jan  1 00:25:03.647: SGBP RES: Scan: Client ID: 1EAF3A41
*Jan  1 00:25:03.651: SGBP RES: Scan: Server ID: 7B09B71E
*Jan  1 00:25:03.651: SGBP RES: Scan: Resource type 8  length 21
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Media type: ISDN
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Min BW: 56000
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Num Links: 0
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port User class: 1
*Jan  1 00:25:03.651: SGBP RES: Scan: Phy-Port Priority: 1
*Jan  1 00:25:03.651: SGBP-RES: received 45 length Offer packet
*Jan  1 00:25:03.651: SGBP-RES: Offer from 7B09B71E for Transaction 3 accepted
*Jan  1 00:25:03.651: SGBP RES: Server is uncongested. Immediate win
An alternate network access server has responded and won the bid.
*Jan  1 00:25:03.651: SGBP-RES: Bid Succeeded  handle 7B09B71E  Server-id 4B0
*Jan  1 00:25:03.651: SGBP-RES: Sent Dial-Req message to ID 7B09B71E  66 bytes
The primary network access server has asked the alternate server to dial.
*Jan  1 00:25:04.651: SGBP-RES: QScan: Purging entry
*Jan  1 00:25:04.651: SGBP-RES: deleting entry 6112E204 1EAF3A41 from list...
```

debug sgbp error

To enable the display of debug messages about routing problems between members of a stack group, use the **debug sgbp error** command in privileged EXEC mode. To disable debug messages about routing problems between members of a stack group, use the **no** form of this command.

debug sgbp error

no debug sgbp error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2(9)	This command was introduced.

Usage Guidelines

Enable the **debug sgbp error** command to enable the display of debug messages about routing problems between members of a stack group.

Note In unusual cases you may see debug messages not documented on this command reference page. These debug messages are intended for expert diagnostic interpretation by the Cisco Technical Assistance Center (TAC).

Examples

One common configuration error is setting a source IP address for a stack member that does not match the locally defined IP address for the same stack member. The following debug output shows the error message that results from this misconfiguration:

```
Systema# debug sgbp error

%SGBP-7-DIFFERENT - systemb's addr 10.1.1.2 is different from hello's addr 10.3.4.5
```

This error means that the source IP address of the Stack Group Bidding Protocol (SGBP) hello message received from systemb does not match the IP address configured locally for systemb (through the **sgbp member** command). Correct this configuaration error by going to systemb and checking for multiple interfaces by which the SGBP hello can send the message.

Another common error message is:

```
Systema# debug sgbp error

%SGBP-7-MISCONF, Possible misconfigured member routerk (10.1.1.6)
```

This error message means that routerk is not defined locally, but is defined on another stack member. Correct this configuration error by defining routerk across all members of the stack group using the **sgbp member** command.

The following error message indicates that an SGBP peer is leaving the stack group:

```
Systema# debug sgbp error

%SGBP-7-LEAVING:Member systemc leaving group stack1
```

This error message indicates that the peer systemc is leaving the stack group. Systemc could be leaving the stack group intentionally, or a connectivity problem may exist.

The following error message indicates that an SGBP event was detected from an unknown peer:

```
Systema# debug sgbp error

%SGBP-7-UNKNOWPEER:Event 0x10 from peer at 172.21.54.3
```

An SGBP event came from a network host that was not recognizable as an SGBP peer. Check to see if a network media error could have corrupted the address, or if peer equipment is malfunctioning to generate corrupted packets. Depending on the network topology and firewalling of your network, SGBP packets from a nonpeer host could indicate probing and attempts to breach security.

Caution If there is a chance your network is under attack, obtain knowledgeable assistance from TAC.

Related Commands

Command	Description
debug sgbp hellos	Enables the display of debug messages for authentication between stack members.
sgbp group	Defines a named stack group and makes this router a member of that stack group.
sgbp member	Specifies the hostname and IP address of a router or access server that is a peer member of a stack group.
show sgbp	Displays the status of the stack group members.
username	Establishes a username-based authentication system.

debug sgbp hellos

To enable the display of debug messages for authentication between stack group members, use the **debug sgbp hellos** command in privileged EXEC mode. To disable debug messages about authentication between stack group members, use the **no** form of this command

debug sgbp hellos

no debug sgbp hellos

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2(9)	This command was introduced.

Usage Guidelines

Enable the **debug sgbp hellos** command to enable the display of debug messages for authentication between routers configured as members of a stack group.

Note In unusual cases you may see debug messages not documented on this command reference page. These debug messages are intended for expert diagnostic interpretation by the Cisco Technical Assistance Center (TAC).

Examples

The following output from the **debug sgbp hellos** command shows systema sending a successful Challenge Handshake Authentication Protocol (CHAP) challenge to and receiving a response from systemb. Similarly, systemb sends out a challenge and receives a response from systema:

```
systema# debug sgbp hellos

%SGBP-7-CHALLENGE: Send Hello Challenge to systemb group stack1
%SGBP-7-CHALLENGED: Hello Challenge message from member systemb (10.1.1.2)
%SGBP-7-RESPONSE: Send Hello Response to systemb group stack1
%SGBP-7-CHALLENGE: Send Hello Challenge to systemb group stack1
%SGBP-7-RESPONDED: Hello Response message from member systemb (10.1.1.2)
%SGBP-7-AUTHOK: Send Hello Authentication OK to member systemb (10.1.1.2)
%SGBP-7-INFO: Addr = 10.1.1.2 Reference = 0xC347DF7
%SGBP-5-ARRIVING: New peer event for member systemb
```

This debug output is self-explanitory.

If authentication fails, you may see one of the following messages in your debug output:

```
%SGBP-7-AUTHFAILED - Member systemb failed authentication
```

This error message means that the remote systemb password for the stack group does not match the password defined on systema. To correct this error, make sure that both systema and systemb have the same password defined using the **username** command.

```
%SGBP-7-NORESP -Fail to respond to systemb group stack1, may not have password.
```

This error message means that systema does not have a username or password defined. To correct this error, define a common group password across all stack members using the **username** command.

Related Commands

Command	Description
debug sgbp error	Enables the display of debug messages about routing problems between members of a stack group.
sgbp group	Defines a named stack group and makes this router a member of that stack group.
sgbp member	Specifies the hostname and IP address of a router or access server that is a peer member of a stack group.
show sgbp	Displays the status of the stack group members.
username	Establishes a username-based authentication system.

debug sgcp

To debug the Simple Gateway Control Protocol (SGCP), use the **debug sgcp** privileged EXEC command. To turn off debugging, use the **no** form of the command.

debug sgcp {**errors** | **events** | **packet**}

no debug sgcp {**errors** | **events** | **packet**}

Syntax Description

errors	Displays debug information about SGCP errors.
events	Displays debug information about SGCP events.
packet	Displays debug information about SGCP packets.

Command History

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

Examples

See the following examples to enable and disable debugging at the specified level:

```
Router# debug sgcp errors

Simple Gateway Control Protocol errors debugging is on

Router# no debug sgcp errors

Simple Gateway Control Protocol errors debugging is off
Router#

Router# debug sgcp events

Simple Gateway Control Protocol events debugging is on
Router# no debug sgcp events

Simple Gateway Control Protocol events debugging is off
Router#

Router# debug sgcp packet

Simple Gateway Control Protocol packets debugging is on
Router# no debug sgcp packet

Simple Gateway Control Protocol packets debugging is off
Router#
```

Related Commands

Command	Description
sgcp	Starts and allocates resources for the SCGP daemon.

debug sgcp errors

To debug Simple Gateway Control Protocol (SGCP) errors, use the **debug sgcp errors** EXEC command. Use the **no** form of this command to turn off debugging.

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

no debug sgcp errors

Syntax Description

endpoint *string*	(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint. On the Cisco MC3810 router, the endpoint string syntax takes the following forms: • DS1 endpoint: **DS1-***slot/port* • POTS endpoint: **aaln/***slot/port* On the Cisco 3600 router, the endpoint string syntax takes the following forms: • DS1 endpoint: *slot/subunit/***DS1-***ds1 number/ds0 number* • POTS endpoint: **aaln/***slot/subunit/port*

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced on the Cisco AS5300 access server in a private release not generally available.
12.0(7)XK	Support for this command was extended to the Cisco MC3810 and the Cisco 3600 series routers (except for the Cisco 3620) in a private release that was not generally available. Also, the **endpoint** keyword was added.

Examples

The following example shows the debugging of SGCP errors being enabled:

```
Router# debug sgcp errors

Simple Gateway Control Protocol errors debugging is on
no errors since call went through successfully.
```

The following example shows a debug trace for SGCP errors on a specific endpoint:

```
Router# debug sgcp errors endpoint DS1-0/1

End point name for error debug:DS1-0/1 (1)
00:08:41:DS1 = 0, DS0 = 1
```

```
00:08:41:Call record found
00:08:41:Enable error end point debug for (DS1-0/1)
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug sgcp events

To debug Simple Gateway Control Protocol (SGCP) events, use the **debug sgcp events** EXEC command. Use the **no debug sgcp events** command to turn off debugging.

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

no debug sgcp events

Syntax Description

endpoint *string*	(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint. On the Cisco MC3810 router, the endpoint string syntax takes the following forms: • DS1 endpoint: **DS1-***slot*/*port* • POTS endpoint: **aaln/***slot*/*port* On the Cisco 3600 router, the endpoint string syntax takes the following forms: • DS1 endpoint: *slot*/*subunit*/**DS1-***ds1 number*/*ds0 number* • POTS endpoint: **aaln/***slot*/*subunit*/*port*

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced on the Cisco AS5300 access server in a private release not generally available.
12.0(7)XK	Support for this command was extended to the Cisco MC3810 and the Cisco 3600 series routers (except for the Cisco 3620 router) in a private release that was not generally available. Also, the **endpoint** keyword was added.

Examples

The following example shows a debug trace for SGCP events on a specific endpoint:

```
Router# debug sgcp events endpoint DS1-0/1

End point name for event debug:DS1-0/1 (1)
00:08:54:DS1 = 0, DS0 = 1
00:08:54:Call record found
00:08:54:Enable event end point debug for (DS1-0/1)
```

The following example shows a debug trace for all SGCP events on a gateway:

```
Router# debug sgcp events

*Mar  1 01:13:31.035:callp :19196BC, state :0, call ID :-1, event :23

*Mar  1 01:13:31.035:voice_if->call_agent_ipaddr used as Notify entityNotify entity
available for Tx SGCP msg
NTFY send to ipaddr=1092E01 port=2427
*Mar  1 01:13:31.039:Push msg into SGCP wait ack queue* (1)[25]
*Mar  1 01:13:31.039:Timed Out interval [1]:(2000)
*Mar  1 01:13:31.039:Timed Out interval [1]:(2000)(0):E[25]
*Mar  1 01:13:31.075:Removing msg :
NTFY 25 ds1-1/13@mc1 SGCP 1.1
X:358258758
O:hd


*Mar  1 01:13:31.075:Unqueue msg from SGCP wait ack q** (0)[25]DS1 = 1, DS0 = 13

*Mar  1 01:13:31.091:callp :19196BC, vdbptr :1964EEC, state :1
*Mar  1 01:13:31.091:Checking ack (trans ID 237740140) :

*Mar  1 01:13:31.091:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:31.091:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                        event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                        event=0x20000004, event2=0xC
*Mar  1 01:13:31.091:Same digit map is download (ds1-1/13@mc1)

*Mar  1 01:13:31.091:R:requested trans_id (237740140)

*Mar  1 01:13:31.091:process_signal_ev:seizure possible=1, signal mask=0x4, mask2=0x0
*Mar  1 01:13:32.405:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.489:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:32.610:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.670:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:32.766:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.810:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:32.931:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:32.967:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.087:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.132:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.240:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.280:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.389:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.433:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.537:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.581:callp :19196BC, state :1, call ID :16, event :9
```

```
*Mar  1 01:13:33.702:SGCP Session Appl:ignore CCAPI event 10

*Mar  1 01:13:33.742:callp :19196BC, state :1, call ID :16, event :9

*Mar  1 01:13:33.742:voice_if->call_agent_ipaddr used as Notify entityNotify entity
available for Tx SGCP msg
NTFY send to ipaddr=1092E01 port=2427
*Mar  1 01:13:33.742:Push msg into SGCP wait ack queue* (1)[26]
*Mar  1 01:13:33.742:Timed Out interval [1]:(2000)
*Mar  1 01:13:33.742:Timed Out interval [1]:(2000)(0):E[26]
*Mar  1 01:13:33.786:Removing msg :
NTFY 26 ds1-1/13@mc1 SGCP 1.1
X:440842371
O:k0, 4081037, s0


*Mar  1 01:13:33.786:Unqueue msg from SGCP wait ack q** (0)[26]DS1 = 1, DS0 = 13

*Mar  1 01:13:33.802:callp :19196BC, vdbptr :1964EEC, state :1
*Mar  1 01:13:33.802:Checking ack (trans ID 698549528) :

*Mar  1 01:13:33.802:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:33.802:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                        event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                        event=0x4, event2=0x0
*Mar  1 01:13:33.802:R:requested trans_id (698549528)

*Mar  1 01:13:33.802:set_up_voip_call_leg:peer_addr=0, peer_port=0.
*Mar  1 01:13:33.806:call_setting_crcx:Enter CallProceeding state rc = 0, call_id=16

*Mar  1 01:13:33.806:callp :19196BC, state :4, call ID :16, event :31

*Mar  1 01:13:33.810:callp :1AF5798, state :2, call ID :17, event :8
call_pre_bridge!

*Mar  1 01:13:33.810:send_oc_create_ack:seizure_possiblle=1, ack-lready-sent=0, ack_send=0
*Mar  1 01:13:33.814:callp :1AF5798, state :4, call ID :17, event :28

*Mar  1 01:13:33.814:Call Connect:Raw Msg ptr=0x1995360, no-offhook=0; call-id=17
*Mar  1 01:13:33.814:SGCP Session Appl:ignore CCAPI event 37

*Mar  1 01:13:33.947:callp :19196BC, state :5, call ID :16, event :32
process_nse_on_orig
DS1 = 1, DS0 = 13

*Mar  1 01:13:34.007:callp :19196BC, vdbptr :1964EEC, state :5
*Mar  1 01:13:34.007:Checking ack (trans ID 123764791) :

*Mar  1 01:13:34.007:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:34.007:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                        event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                        event=0x4, event2=0x0
*Mar  1 01:13:34.007:R:requested trans_id (123764791)

*Mar  1 01:13:34.007:process_signal_ev:seizure possible=1, signal mask=0x0, mask2=0x0
*Mar  1 01:13:34.007:modify_connection:echo_cancel=1.
*Mar  1 01:13:34.007:modify_connection:vad=0.
*Mar  1 01:13:34.007:modify_connection:peer_addr=6000001, peer_port=0->16500.
*Mar  1 01:13:34.007:modify_connection:conn_mode=2.
*Mar  1 01:13:34.011:callp :19196BC, state :5, call ID :16, event :31
```

```
*Mar  1 01:13:34.011:callp :1AF5798, state :5, call ID :17, event :31
process_nse_event

*Mar  1 01:13:34.051:callp :19196BC, state :5, call ID :16, event :39

*Mar  1 01:13:34.051:call_id=16, ignore_ccapi_ev:ignore 19 for state 5
DS1 = 1, DS0 = 13

*Mar  1 01:13:39.497:callp :19196BC, vdbptr :1964EEC, state :5
*Mar  1 01:13:39.497:Checking ack (trans ID 553892443) :

*Mar  1 01:13:39.497:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:13:39.497:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                        event=0x6003421F, event2=0x3FD
requested signal=0x8, signal2=0x0,
                        event=0x4, event2=0x0
*Mar  1 01:13:39.497:R:requested trans_id (553892443)

*Mar  1 01:13:39.497:process_signal_ev:seizure possible=1, signal mask=0x0, mask2=0x0
*Mar  1 01:13:39.497:modify_connection:echo_cancel=1.
*Mar  1 01:13:39.497:modify_connection:vad=0.
*Mar  1 01:13:39.497:modify_connection:peer_addr=6000001, peer_port=16500->16500.
*Mar  1 01:13:39.497:modify_connection:conn_mode=3.
*Mar  1 01:13:39.497:callp :19196BC, state :5, call ID :16, event :31

*Mar  1 01:13:39.501:callp :1AF5798, state :5, call ID :17, event :31

*Mar  1 01:14:01.168:Removing ack (trans ID 237740140) :
 200 237740140 OK


*Mar  1 01:14:03.883:Removing ack (trans ID 698549528) :
 200 698549528 OK
I:7

v=0
c=IN IP4 5.0.0.1
m=audio 16400 RTP/AVP 0


*Mar  1 01:14:04.087:Removing ack (trans ID 123764791) :
 200 123764791 OK
I:7

v=0
c=IN IP4 5.0.0.1
m=audio 16400 RTP/AVP 0


*Mar  1 01:14:09.573:Removing ack (trans ID 553892443) :
 200 553892443 OK
I:7

v=0
c=IN IP4 5.0.0.1
m=audio 16400 RTP/AVP 0


*Mar  1 01:14:48.091:callp :19196BC, state :5, call ID :16, event :12

*Mar  1 01:14:48.091:voice_if->call_agent_ipaddr used as Notify entityNotify entity
available for Tx SGCP msg
NTFY send to ipaddr=1092E01 port=2427
*Mar  1 01:14:48.091:Push msg into SGCP wait ack queue* (1)[27]
```

```
*Mar  1 01:14:48.091:Timed Out interval [1]:(2000)
*Mar  1 01:14:48.091:Timed Out interval [1]:(2000)(0):E[27]
*Mar  1 01:14:48.128:Removing msg :
NTFY 27 ds1-1/13@mc1 SGCP 1.1
X:97849341
O:hu


*Mar  1 01:14:48.128:Unqueue msg from SGCP wait ack q** (0)[27]DS1 = 1, DS0 = 13

*Mar  1 01:14:48.212:callp :19196BC, vdbptr :1964EEC, state :5
*Mar  1 01:14:48.212:Checking ack (trans ID 79307869) :

*Mar  1 01:14:48.212:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:14:48.212:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                        event=0x6003421F, event2=0x3FD
requested signal=0x4, signal2=0x0,
                        event=0x0, event2=0x0
*Mar  1 01:14:48.212:delete_call:callp:19196BC, call ID:16
*Mar  1 01:14:48.212:sgcp delete_call:Setting disconnect_by_dlcx to 1
*Mar  1 01:14:48.216:callp :1AF5798, state :6, call ID :17, event :29

*Mar  1 01:14:48.216:Call disconnect:Raw Msg ptr = 0x0, call-id=17
*Mar  1 01:14:48.216:disconnect_call_leg O.K. call_id=17
*Mar  1 01:14:48.216:SGCP:Call disconnect:No need to send onhook
*Mar  1 01:14:48.216:Call disconnect:Raw Msg ptr = 0x19953B0, call-id=16
*Mar  1 01:14:48.216:disconnect_call_leg O.K. call_id=16
*Mar  1 01:14:48.220:callp :1AF5798, state :7, call ID :17, event :13

*Mar  1 01:14:48.220:Processing DLCX signal request :4, 0, 0

*Mar  1 01:14:48.220:call_disconnected:call_id=17, peer 16 is not idle yet.DS1 = 1, DS0 =
13

*Mar  1 01:14:48.272:callp :19196BC, vdbptr :1964EEC, state :7
*Mar  1 01:14:48.272:Checking ack (trans ID 75540355) :

*Mar  1 01:14:48.272:is_capability_ok:caps.codec=5, caps.pkt=10, caps.nt=8
*Mar  1 01:14:48.272:is_capability_ok:supported signal=0x426C079C, signal2=0x80003,
                        event=0x6003421F, event2=0x3FD
requested signal=0x0, signal2=0x0,
                        event=0x8, event2=0x0
*Mar  1 01:14:48.272:R:requested trans_id (75540355)

*Mar  1 01:14:48.272:process_signal_ev:seizure possible=1, signal mask=0x4, mask2=0x0
*Mar  1 01:14:49.043:callp :19196BC, state :7, call ID :16, event :27

*Mar  1 01:14:49.043:process_call_feature:Onhook event
*Mar  1 01:14:49.043:callp :19196BC, state :7, call ID :16, event :13

*Mar  1 01:15:18.288:Removing ack (trans ID 79307869) :
 250 79307869 OK


*Mar  1 01:15:18.344:Removing ack (trans ID 75540355) :
 200 75540355 OK
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp packet	Debugs SGCP packets.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug sgcp packet

To debug the Simple Gateway Control Protocol (SGCP), use the **debug sgcp packet** EXEC command. Use the **no debug sgcp packet** command to turn off debugging.

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

no debug sgcp packet

Syntax Description

endpoint *string*	(Optional) Specifies the endpoint string if you want to debug SGCP errors for a specific endpoint. On the Cisco MC3810, the endpoint string syntax takes the following forms: • DS1 endpoint: **DS1-***slot*/*port* • POTS endpoint: **aaln/***slot*/*port* On the Cisco 3600, the endpoint string syntax takes the following forms: • DS1 endpoint: *slot*/*subunit*/**DS1-***ds1 number*/*ds0 number* • POTS endpoint: **aaln**/*slot*/*subunit*/*port*

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(5)T	This command was introduced on the Cisco AS5300 in a private release not generally available.
12.0(7)XK	Support for this command was extended to the Cisco MC3810 and the Cisco 3600 series routers (except for the Cisco 3620) in a private release that was not generally available. Also, the **endpoint** keyword was added

Examples

The following example shows a debug trace for SGCP packets on a specific endpoint:

```
Router# debug sgcp packet endpoint DS1-0/1

End point name for packet debug:DS1-0/1 (1)
00:08:14:DS1 = 0, DS0 = 1
00:08:14:Enable packet end point debug for (DS1-0/1)
```

The following example shows a debug trace for all SGCP packets on a gateway:

```
Router# debug sgcp packet

*Mar  1 01:07:45.204:SUCCESS:Request ID string building is OK
```

```
*Mar  1 01:07:45.204:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:45.204:SUCCESS:SGCP message building OK
*Mar  1 01:07:45.204:SUCCESS:END of building
*Mar  1 01:07:45.204:SGCP Packet sent --->
NTFY 22 ds1-1/13@mc1 SGCP 1.1
X:550092018
O:hd

<---

*Mar  1 01:07:45.204:NTFY Packet sent successfully.
*Mar  1 01:07:45.240:Packet received -

200 22


*Mar  1 01:07:45.244:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:45.244:SUCCESS:END of Parsing
*Mar  1 01:07:45.256:Packet received -

RQNT 180932866 ds1-1/13@mc1 SGCP 1.1
X:362716780
R:hu,k0(A),s0(N),[0-9T](A) (D)
D:(9xx|xxxxxxx)


*Mar  1 01:07:45.256:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:45.256:SUCCESS:Request ID string(362716780) parsing is OK
*Mar  1 01:07:45.260:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:45.260:SUCCESS:Digit Map parsing is OK
*Mar  1 01:07:45.260:SUCCESS:END of Parsing
*Mar  1 01:07:45.260:SUCCESS:SGCP message building OK
*Mar  1 01:07:45.260:SUCCESS:END of building
*Mar  1 01:07:45.260:SGCP Packet sent --->
200 180932866 OK

<---

*Mar  1 01:07:47.915:SUCCESS:Request ID string building is OK
*Mar  1 01:07:47.915:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:47.919:SUCCESS:SGCP message building OK
*Mar  1 01:07:47.919:SUCCESS:END of building
*Mar  1 01:07:47.919:SGCP Packet sent --->
NTFY 23 ds1-1/13@mc1 SGCP 1.1
X:362716780
O:k0, 4081037, s0

<---

*Mar  1 01:07:47.919:NTFY Packet sent successfully.
*Mar  1 01:07:47.955:Packet received -

200 23


*Mar  1 01:07:47.955:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:47.955:SUCCESS:END of Parsing
*Mar  1 01:07:47.971:Packet received -

CRCX 938694984 ds1-1/13@mc1 SGCP 1.1
M:recvonly
L:p:10,e:on,s:off, a:G.711u
R:hu
C:6
```

```
*Mar  1 01:07:47.971:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:47.971:SUCCESS:Connection Mode parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Packet period parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Echo Cancellation parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Silence Supression parsing is OK
*Mar  1 01:07:47.971:SUCCESS:CODEC strings parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Local Connection option parsing is OK
*Mar  1 01:07:47.971:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:47.975:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:07:47.975:SUCCESS:END of Parsing
*Mar  1 01:07:47.979:SUCCESS:Conn ID string building is OK
*Mar  1 01:07:47.979:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:47.979:SUCCESS:SGCP message building OK
*Mar  1 01:07:47.979:SUCCESS:END of building
*Mar  1 01:07:47.979:SGCP Packet sent --->
200 938694984 OK
I:6

v=0
c=IN IP4 5.0.0.1
m=audio 16538 RTP/AVP 0

<---

*Mar  1 01:07:48.188:Packet received -

MDCX 779665338 ds1-1/13@mc1 SGCP 1.1
I:6
M:recvonly
L:p:10,e:on,s:off,a:G.711u
R:hu
C:6

v=0
c=IN IP4 6.0.0.1
m=audio 16392 RTP/AVP 0


*Mar  1 01:07:48.188:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:48.188:SUCCESS:Conn ID string(6) parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Connection Mode parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Packet period parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Echo Cancellation parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Silence Supression parsing is OK
*Mar  1 01:07:48.192:SUCCESS:CODEC strings parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Local Connection option parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:48.192:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:07:48.192:SUCCESS:SDP Protocol version parsing OK
*Mar  1 01:07:48.192:SUCCESS:SDP Conn Data OK
*Mar  1 01:07:48.192:SUCCESS:END of Parsing
*Mar  1 01:07:48.200:SUCCESS:Conn ID string building is OK
*Mar  1 01:07:48.200:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:48.200:SUCCESS:SGCP message building OK
*Mar  1 01:07:48.200:SUCCESS:END of building
*Mar  1 01:07:48.200:SGCP Packet sent --->
200 779665338 OK
I:6

v=0
c=IN IP4 5.0.0.1
m=audio 16538 RTP/AVP 0
```

```
<---

*Mar  1 01:07:53.674:Packet received -

MDCX 177780432 ds1-1/13@mc1 SGCP 1.1
I:6
M:sendrecv
X:519556004
L:p:10,e:on, s:off,a:G.711u
C:6
R:hu
S:hd

v=0
c=IN IP4 6.0.0.1
m=audio 16392 RTP/AVP 0


*Mar  1 01:07:53.674:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:07:53.674:SUCCESS:Conn ID string(6) parsing is OK
*Mar  1 01:07:53.674:SUCCESS:Connection Mode parsing is OK
*Mar  1 01:07:53.674:SUCCESS:Request ID string(519556004) parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Packet period parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Echo Cancellation parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Silence Supression parsing is OK
*Mar  1 01:07:53.678:SUCCESS:CODEC strings parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Local Connection option parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Requested Event parsing is OK
*Mar  1 01:07:53.678:SUCCESS:Signal Requests parsing is OK
*Mar  1 01:07:53.678:SUCCESS:SDP Protocol version parsing OK
*Mar  1 01:07:53.678:SUCCESS:SDP Conn Data OK
*Mar  1 01:07:53.678:SUCCESS:END of Parsing
*Mar  1 01:07:53.682:SUCCESS:Conn ID string building is OK
*Mar  1 01:07:53.682:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:07:53.682:SUCCESS:SGCP message building OK
*Mar  1 01:07:53.682:SUCCESS:END of building
*Mar  1 01:07:53.682:SGCP Packet sent --->
200 177780432 OK
I:6

v=0
c=IN IP4 5.0.0.1
m=audio 16538 RTP/AVP 0

<---

*Mar  1 01:09:02.401:SUCCESS:Request ID string building is OK
*Mar  1 01:09:02.401:SUCCESS:Building SGCP Parameter lines is OK
*Mar  1 01:09:02.401:SUCCESS:SGCP message building OK
*Mar  1 01:09:02.401:SUCCESS:END of building
*Mar  1 01:09:02.401:SGCP Packet sent --->
NTFY 24 ds1-1/13@mc1 SGCP 1.1
X:519556004
O:hu

<---

*Mar  1 01:09:02.401:NTFY Packet sent successfully.
*Mar  1 01:09:02.437:Packet received -

200 24
```

```
*Mar  1 01:09:02.441:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:09:02.441:SUCCESS:END of Parsing
*Mar  1 01:09:02.541:Packet received -

DLCX 865375036 ds1-1/13@mc1 SGCP 1.1
C:6
S:hu


*Mar  1 01:09:02.541:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:09:02.541:SUCCESS:Call ID string(6) parsing is OK
*Mar  1 01:09:02.541:SUCCESS:Signal Requests parsing is OK
*Mar  1 01:09:02.541:SUCCESS:END of Parsing
*Mar  1 01:09:02.545:SUCCESS:SGCP message building OK
*Mar  1 01:09:02.545:SUCCESS:END of building
*Mar  1 01:09:02.545:SGCP Packet sent --->
250 865375036 OK

<---

*Mar  1 01:09:02.577:Packet received -

RQNT 254959796 ds1-1/13@mc1 SGCP 1.1
X:358258758
R:hd


*Mar  1 01:09:02.577:SUCCESS:SGCP Header parsing was OK
*Mar  1 01:09:02.577:SUCCESS:Request ID string(358258758) parsing is OK
*Mar  1 01:09:02.577:SUCCESS:Requested Event parsing is OK
*Mar  1 01:09:02.581:SUCCESS:END of Parsing
*Mar  1 01:09:02.581:SUCCESS:SGCP message building OK
*Mar  1 01:09:02.581:SUCCESS:END of building
*Mar  1 01:09:02.581:SGCP Packet sent --->
200 254959796 OK
```

Command History

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug vtsp send-nse	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

debug smrp all

To display information about Simple Multicast Routing Protocol (SMRP) activity, use the **debug smrp all** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp all

no debug smrp all

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Because the **debug smrp all** command displays all SMRP debugging output, it is processor intensive and should not be enabled when memory is scarce or in very high traffic situations.

For general debugging, use the **debug smrp all** command and turn off excessive transactions with the **no debug smrp transaction** command. This combination of commands will display various state changes and events without displaying every transaction packet. For debugging a specific feature such as a routing problem, use the **debug smrp route** and **debug smrp transaction** commands to learn if packets are sent and received and which specific routes are affected. The **show smrp traffic** EXEC command is highly recommended as a troubleshooting method because it displays the SMRP counters.

For examples of the type of output you may see, refer to each of the commands listed in the "Related Commands" section.

Related Commands

Command	Description
debug smrp group	Displays information about SMRP group activity.
debug smrp mcache	Displays information about SMRP multicast fast-switching cache entries.
debug smrp neighbor	Displays information about SMRP neighbor activity.
debug smrp port	Displays information about SMRP port activity.
debug smrp route	Displays information about SMRP routing activity.
debug smrp transaction	Displays information about SMRP transactions.

debug smrp group

To display information about SMRP group activity, use the **debug smrp group** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp group

no debug smrp group

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug smrp group** command displays information when a group is created or deleted and when a forwarding entry for a group is created, changed, or deleted. For more information, refer to the **show smrp group** command described in the *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Examples

The following is sample output from the **debug smrp group** command showing a port being created and deleted on group AT 20.34. (AT signifies that this is an AppleTalk network group.)

```
Router# debug smrp group

SMRP: Group AT 20.34, created on port 20.1 by 20.2
SMRP: Group AT 20.34, deleted on port 20.1
```

Table 172 lists the messages that may be generated with the **debug smrp group** command concerning the forwarding table.

Table 172 debug smrp group Message Descriptions

Messages	Descriptions
Group <*address*>, deleted on port <*address*>	Group entry was deleted from the group table for the specified port.
Group <*address*>, forward state changed from *state* to *state*	State of the group changed. States are join, forward, and leave.
Group <*address*>, deleted forward entry	Group was deleted from the forwarding table.
Group <*address*>, created on port <*address*> by <*address*>	Group entry was created in the table for the specified port.
Group <*address*>, added by <*address*> to the group	Secondary router has added this group to its group table.
Group <*address*>, discard join request from <*address*>, not responsible	Discard Join Group request if the router is not the primary router on the local connected network or if it is not the port parent of the route.
Group <*address*>, join request from <*address*>	Request to join the group was received.

Table 172 debug smrp group Message Descriptions (continued)

Messages	Descriptions
Group <*address*>, forward is found	Forward entry for the group was found in the forwarding table.
Group <*address*>, forward state is already joining, ignored	Request to join the group is in progress, so the second request was discarded.
Group <*address*>, no forward found	Forward entry for the group was not found in the forwarding table.
Group <*address*>, join request discarded, fw discarded, fwd parent port not operational	Request to join the group was discarded because the parent port is not available.
Group <*address*>, created forward entry - parent <*address*> child <*address*>	Forward entry was created in the forwarding table for the parent and child address.
Group <*address*>, creator no longer up on <*address*>	Group creator has not been heard from for a specified time and is deemed no longer available.
Group <*address*>, pruning duplicate path on <*address*>	Duplicate path was removed. If we are forwarding and we are a child port, and our port parent address is not pointing to our own port address, we are in a duplicate path.
Group <*address*>, member no longer up on <*address*>	Group member has not been heard from for a specified time and is deemed no longer available.
Group <*address*>, no more child ports in forward entry	Forward entry for group no longer has any child ports. As a result, the forward entry is no longer necessary.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp mcache

To display information about SMRP multicast fast-switching cache entries, use the **debug smrp mcache** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp mcache

no debug smrp mcache

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **show smrp mcache** EXEC command (described in the *Cisco IOS AppleTalk and Novell IPX Command Reference* to display the entries in the SMRP multicast cache, and use the **debug smrp mcache** command to learn whether the cache is being populated and invalidated.

Examples

The following is sample output from the **debug smrp mcache** command. In this example, the cache is created and populated for group AT 11.124. (AT signifies that this is an AppleTalk network group.)

```
Router# debug smrp mcache

SMRP: Cache created
SMRP: Cache populated for group AT 11.124
        mac - 090007400b7c00000c1740d9
        net - 001fef7500000014ff020a0a0a
SMRP: Forward cache entry created for group AT 11.124
SMRP: Forward cache entry validated for group AT 11.124
SMRP: Forward cache entry invalidated for group AT 11.124
SMRP: Forward cache entry deleted for group AT 11.124
```

Table 173 lists all the messages that can be generated with the **debug smrp mcache** command concerning the multicast cache.

Table 173 debug smrp mcache Message Descriptions

Messages	Descriptions
Cache populated for group <*address*>	SMRP packet was received on a parent port that has fast switching enabled. As a result, the cache was created and the MAC and network headers were stored for all child ports that have fast switching enabled. Use the **show smrp port appletalk** EXEC command with the optional interface type and number to display the switching path.
Cache memory allocated	Memory was allocated for the multicast cache.
Forward cache entry created/deleted for group <*address*>	Forward cache entry for the group was added to or deleted from the cache.
Forward cache entry validated for group <*address*>	Forward cache entry is validated and is now ready for fast switching.
Forward cache entry invalidated for group <*address*>	Cache entry is invalidated because some change (such as port was shut down) occurred to one of the ports.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp neighbor

To display information about SMRP neighbor activity, use the **debug smrp neighbor** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp neighbor

no debug smrp neighbor

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug smrp neighbor** command displays information when a neighbor operating state changes. A neighbor is an adjacent router. For more information, refer to the **show smrp neighbor** EXEC command described in the *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Examples

The following is sample output from the **debug smrp neighbor** command. In this example, the neighbor on port 30.02 has changed state from normal operation to secondary operation.

```
Router# debug smrp neighbor

SMRP: Neighbor 30.2, state changed from "normal op" to "secondary op"
```

Table 174 lists all the messages that can be generated with the **debug smrp neighbor** command concerning the neighbor table.

Table 174 debug smrp neighbor Message Descriptions

Messages	Descriptions
Neighbor <*address*>, state changed from *state* to *state*	State of the neighbor changed. States are primary operation, secondary operation, normal operation, primary negotiation, secondary negotiation, and down.
Neighbor <*address*>, neighbor added/deleted	Neighbor was added to or removed from the neighbor table.
SMRP neighbor up/down	Neighbor is available for service or unavailable.
Neighbor <*address*>, no longer up	Neighbor is unavailable because it has not been heard from for a specified duration.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp port

To display information about SMRP port activity, use the **debug smrp port** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp port

no debug smrp port

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug smrp port** command displays information when a port operating state changes. For more information, refer to the **show smrp port** command described in the *Cisco IOS AppleTalk and Novell IPX Command Reference*.

Examples

The following is sample output from the **debug smrp port** command. In this example, port 30.1 has changed state from secondary negative to secondary operation to primary negative:

```
Router# debug smrp port

SMRP: Port 30.1, state changed from "secondary neg" to "secondary op"
SMRP: Port 30.1, secondary router changed from 0.0 to 30.1
SMRP: Port 30.1, state changed from "secondary op" to "primary neg"
```

Table 175 lists all the messages that can be generated with the **debug smrp port** command concerning the port table.

Table 175 debug smrp port Message Descriptions

Messages	Descriptions
Port <*address*>, port created/deleted	Port entry was added to or removed from the port table.
Port <*address*>, line protocol changed to *state*	Line protocol for the port is up or down.
Port <*address*>, state changed from *state* to *state*	State of the port changed. States are primary operation, secondary operation, normal operation, primary negotiation, secondary negotiation, and down.
Port <*address*>, primary/secondary router changed from <*address*> to <*address*>	Primary or secondary port address of the router changed.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp route

To display information about SMRP routing activity, use the **debug smrp route** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp route

no debug smrp route

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

For more information, refer to the **show smrp route** EXEC command described in the *Cisco IOS AppleTalk and Novell IPX Command Reference.*

Examples

The following is sample output from the **debug smrp route** command. In this example, poison notification is received from port 30.2. Poison notification is the receipt of a poisoned route on a nonparent port.

```
Router# debug smrp route

SMRP: Route AT 20-20, poison notification from 30.2
SMRP: Route AT 30-30, poison notification from 30.2
```

Table 176 lists all the messages that can be generated with the **debug smrp route** command concerning the routing table. In Table 176, the term *route* does not refer to an address but rather to a network range.

Table 176 debug smrp route Message Descriptions

Messages	Descriptions
Route address, deleted/created as local network	Route entry was removed from or added to the routing table.
Route address, from address has invalid distance value	Route entry from the specified address has an incorrect distance value and was ignored.
Route address, unknown route poisoned by address ignored	Route entry received from the specified address is bad and was ignored.
Route address, created via address - hop number tunnel number	New route entry added to the routing table with the specified number of hops and tunnels.
Route address, from address - overlaps existing route	Route entry received from the specified address overlaps an existing route and was ignored.
Route address, poisoned by address	Route entry has been poisoned by neighbor. Poisoned routes have distance of 255.
Route address, poison notification from address	Poisoned route is received from a nonparent port.
Route address, worsened by parent address	Distance to the route has worsened (become higher), received from the parent neighbor.

Table 176 debug smrp route Message Descriptions (continued)

Messages	Descriptions
Route address, improved via address - number -> number hop, number -> number tunnel	Distance to the route has improved (become lower), received from a neighbor.
Route address, switched to address - higher address than address	Tie condition exists, and because this router had the highest network address, it was used to forward the packet.
Route address, parent port changed address -> address	Parent port address change occurred. The parent port address of a physical network segment determines which router should handle Join Group and Leave Group requests.
SMRP bad distance vector	Packet has an invalid distance vector and was ignored.
Route address, has been poisoned	Route has been poisoned. Poisoned routes are purged from the routing table after a specified time.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug smrp transaction

To display information about SMRP transactions, use the **debug smrp transaction** privileged EXEC command. The **no** form of this command disables debugging output.

debug smrp transaction

no debug smrp transaction

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug smrp transaction** command. In this example, a secondary node request is sent out to all routers on port 30.1.

```
Router# debug smrp transaction

SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
SMRP: Transaction for port 30.1, secondary node request (seq 8435) sent to all routers
```

Table 177 lists all the messages that can be generated with the **debug smrp route** command.

Table 177 debug smrp Transaction Message Descriptions

Messages	Descriptions
Transaction for port address, packet-type command-type (grp/sec number) sent to/received from address	Port message concerning a packet or command was sent to or received from the specified address.
Transaction for group address on port address, (seq number) sent to/received from address	Group message for a specified port was sent to or received from the specified address.
Unrecognized transaction for port address	Unrecognized message was received and ignored by the port.
Discarded incomplete request	Incomplete message was received and ignored.
Response in wrong state in HandleRequest	Message was received with the wrong state and was ignored.
SMRP bad packet type	SMRP packet was received with a bad packet type and was ignored.
Packet discarded, Bad Port ID	Packet was received with a bad port ID and was ignored.
Packet discarded, Check Packet failed	Packet was received with a failed check packet and was ignored.

Related Commands

Command	Description
debug sgbp dial-bids	Displays large-scale dial-out negotiations between the primary NAS and alternate NASs.

debug snasw dlc

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

debug snasw dlc detail

Syntax Description

detail	Indicates that in addition to a one-line description of the frame being displayed, an entire hexadecimal dump of the frame will follow.

Defaults

By default, a one-line description of the frame is displayed.

Caution The **debug snasw dlc** command displays the same trace information available via the **snasw dlctrace** command. The **snasw dlctrace** command is the preferred method for gathering this trace information because it is written to a capture buffer instead of directly to the console. The **debug snasw dlc** command should only be used when it is certain that the output will not cause excessive data to be output to the console.

Command History

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

Examples

The following is an example of the **debug snasw dlc** command output:

```
Router# debug snasw dlc

Sequence
Number              Size of ISR/
       Link         SNA BTU HPR  Description of frame

343    MVSD     In  sz:134  ISR fmh5 DLUR Rq ActPU NETA.APPNRA29
344    MVSD     Out sz:12   ISR +Rsp IPM     slctd nws:0008
345    @I000002 Out sz:18   ISR Rq ActPU
346    MVSD     Out sz:273  ISR fmh5 TOPOLOGY UPDATE
347    @I000002 In  sz:9    ISR +Rsp Data
348    @I000002 In  sz:12   ISR +Rsp IPM     slctd nws:0002
349    @I000002 In  sz:29   ISR +Rsp ActPU
350    MVSD     Out sz:115  ISR fmh5 DLUR +Rsp ActPU
351    MVSD     In  sz:12   ISR +Rsp IPM     slctd nws:0007
352    MVSD     In  sz:88   ISR fmh5 DLUR Rq ActLU NETA.MARTLU1
353    MVSD     Out sz:108  ISR fmh5 REGISTER
354    @I000002 Out sz:27   ISR Rq ActLU NETA.MARTLU1
```

Related Commands

Command	Description
snasw dlctrace	Captures trace frames entering and leaving the SNA switching Services feature.
snasw dlcfilter	Filters frames traced by the **snasw dlctrace** or **debug snasw dlc** command.

debug snasw ips

To display internal signal information between the SNA switch and the console in real time, use the **debug snasw ips** privileged EXEC command.

debug snasw dlc

Syntax Description

This command has no arguments or keywords.

Defaults

By default, a one-line description of the interprocess signal is displayed.

Caution The **debug snasw ips** command displays the same trace information available via the **snasw ipstrace** command. Output from this **debug** command can be large. The **snasw ipstrace** command is the preferred method for gathering this trace information because it is written to a capture buffer instead of directly to the console. The **debug snasw ips** command should only be used when it is certain that the output will not cause excessive data to be output to the console. The **debug snasw dlc** command displays the same trace information available via the **snasw dlctrace** command.

Command History

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

Examples

The following is an example of the **debug snasw ips** command output:

```
Router# debug snasw ips

Sequence
Number                 Sending     Receiving
        Signal Name    Process     Process    Queue

11257 : DEALLOCATE_RCB : --(0) -> RM(2130000) Q 4
11258 : RCB_DEALLOCATED : RM(2130000) -> PS(22E0000) Q 2
11259 : RCB_DEALLOCATED : --(0) -> PS(22E0000) Q 2
11260 : VERB_SIGNAL : PS(22E0000) -> DR(20F0000) Q 2
11261 : FREE_SESSION : --(0) -> RM(2130000) Q 2
11262 : BRACKET_FREED : RM(2130000) -> HS(22FB0001) Q 2
11263 : BRACKET_FREED : --(0) -> HS(22FB0001) Q 2
11264 : VERB_SIGNAL : --(0) -> DR(20F0000) Q 2
11265 : DLC_MU : DLC(2340000) -> PC(22DD0001) Q 2
11266 : DLC_MU : --(0) -> PC(22DD0001) Q 2
```

Related Commands

Command	Description
snasw ipstrace	Captures interprocess signal information between Switching Services components.

debug snmp packet

To display information about every SNMP packet sent or received by the router, use the **debug snmp packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug snmp packet

no debug snmp packet

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug snmp packet** command. In this example, the router receives a get-next request from the host at 172.16.63.17 and responds with the requested information.

```
Router# debug snmp packet

SNMP: Packet received via UDP from 172.16.63.17 on Ethernet0
SNMP: Get-next request, reqid 23584, errstat 0, erridx 0
 sysUpTime = NULL TYPE/VALUE
 system.1 = NULL TYPE/VALUE
 system.6 = NULL TYPE/VALUE
SNMP: Response, reqid 23584, errstat 0, erridx 0
 sysUpTime.0 = 2217027
 system.1.0 = Cisco Internetwork Operating System Software
 system.6.0 =
SNMP: Packet sent via UDP to 172.16.63.17
```

Based on the kind of packet sent or received, the output may vary. For get-bulk requests, a line similar to the following is displayed:

```
SNMP: Get-bulk request, reqid 23584, nonrptr 10, maxreps 20
```

For traps, a line similar to the following is displayed:

```
SNMP: V1 Trap, ent 1.3.6.1.4.1.9.1.13, gentrap 3, spectrap 0
```

Table 178 describes the significant fields shown in the display.

Table 178 debug snmp packet Field Descriptions

Field	Description
Get-next request	Indicates what type of SNMP PDU the packet is. Possible types are as follows: • Get request • Get-next request • Response • Set request • V1 Trap • Get-bulk request • Inform request • V2 Trap Depending on the type of PDU, the rest of this line displays different fields. The indented lines following this line list the MIB object names and corresponding values.
reqid	Request identification number. This number is used by the SNMP manager to match responses with requests.
errstat	Error status. All PDU types other than response will have an errstat of 0. If the agent encounters an error while processing the request, it will set errstat in the response PDU to indicate the type of error.
erridx	Error index. This value will always be 0 in all PDUs other than responses. If the agent encounters an error, the erridx will be set to indicate which varbind in the request caused the error. For example, if the agent had an error on the second varbind in the request PDU, the response PDU will have an erridx equal to 2.
nonrptr	Nonrepeater value. This value and the maximum repetition value are used to determine how many varbinds are returned. Refer to RFC 1905 for details.
maxreps	Maximum repetition value. This value and the nonrepeater value are used to determine how many varbinds are returned. Refer to RFC 1905 for details.
ent	Enterprise object identifier. Refer to RFC 1215 for details.
gentrap	Generic trap value. Refer to RFC 1215 for details.
spectrap	Specific trap value. Refer to RFC 1215 for details.

debug snmp requests

To display information about every SNMP request made by the SNMP manager, use the **debug snmp requests** privileged EXEC command. The **no** form of this command disables debugging output.

debug snmp requests

no debug snmp requests

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug snmp requests** command:

```
Router# debug snmp requests

SNMP Manager API: request
  dest: 171.69.58.33.161, community: public
  retries: 3, timeout: 30, mult: 2, use session rtt
  userdata: 0x0
```

Table 179 describes the significant fields shown in the display.

Table 179 debug snmp requests field Field Descriptions

Field	Description
SNMP Manager API	Indicates that the router sent an SNMP request.
dest	Destination of the request.
community	Community string sent with the request.
retries	Number of times the request has been re-sent.
timeout	Request timeout, or how long the router will wait before resending the request.
mult	Timeout multiplier. The timeout for a re-sent request will be equal to the previous timeout multiplied by the timeout multiplier.
use session rtt	Indicates that the average round-trip time of the session should be used in calculating the timeout value.
userdata	Internal Cisco IOS software data.

debug sntp adjust

To display information about SNTP clock adjustments, use the **debug sntp adjust** privileged EXEC command. The **no** form of this command disables debugging output.

debug sntp adjust

no debug sntp adjust

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug sntp adjust** command output when an offset to the time reported by the configured NTP server is calculated. The offset indicates the difference between the router time and the actual time (as kept by the server) and is displayed in milliseconds. The clock time is then successfully changed to the accurate time by adding the offset to the current router time.

```
Router# debug sntp adjust

Delay calculated, offset 3.48
Clock slewed.
```

The following is sample output from the **debug sntp adjust** command when an offset to the time reported by a broadcast server is calculated. Because the packet is a broadcast packet, no transmission delay can be calculated. However, in this case, the offset is too large, so the clock is reset to the correct time.

```
Router# debug sntp adjust

No delay calculated, offset 11.18
Clock stepped.
```

debug sntp packets

To display information about SNTP packets sent and received, use the **debug sntp packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug sntp packets

no debug sntp packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug sntp packets

Received SNTP packet from 172.16.186.66, length 48
 leap 0, mode 1, version 3, stratum 4, ppoll 1024
 rtdel 00002B00, rtdsp 00003F18, refid AC101801 (172.16.24.1)
 ref B7237786.ABF9CDE5 (23:28:06.671 UTC Tue May 13 1997)
 org 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
 rec 00000000.00000000 (00:00:00.000 UTC Mon Jan 1 1900)
 xmt B7237B5C.A7DE94F2 (23:44:28.655 UTC Tue May 13 1997)
 inp AF3BD529.810B66BC (00:19:53.504 UTC Mon Mar 1 1993)
```

The following is sample output from the **debug sntp packets** command when a message is sent:

```
Router# debug sntp packets

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

Table 180 describes the significant fields shown in the display.

Table 180 debug sntp packets Field Descriptions

Field	Description
length	Length of the SNTP packet.
leap	Indicates if a leap second will be added or subtracted.
mode	Indicates the mode of the router relative to the server sending the packet.
version	SNTP version number of the packet.
stratum	Stratum of the server.
ppoll	Peer polling interval.
rtdel	Total delay along the path to the root clock.
rtdsp	Dispersion of the root path.
refid	Address of the server that the router is currently using for synchronization.
ref	Reference time stamp.
org	Originate time stamp. This value indicates the time the request was sent by the router.

Table 180 debug sntp packets Field Descriptions (continued)

Field	Description
rec	Receive time stamp. This value indicates the time the request was received by the SNTP server.
xmt	Transmit time stamp. This value indicates the time the reply was sent by the SNTP server.
inp	Destination time stamp. This value indicates the time the reply was received by the router.

debug sntp select

To display information about SNTP server selection, use the **debug sntp select** privileged EXEC command. The **no** form of this command disables debugging output.

debug sntp select

no debug sntp select

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the show sample **debug sntp select** command. In this example, the router will synchronize its time to the server at 172.16.186.66.

```
Router# debug sntp select

SNTP: Selected 172.16.186.66
```

debug source bridge

To display information about packets and frames transferred across a source-route bridge, use the **debug source bridge** privileged EXEC command. The **no** form of this command disables debugging output.

debug source bridge

no debug source bridge

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug source bridge** output for peer bridges using TCP as a transport mechanism. The remote source-route bridging (RSRB) network configuration has ring 2 and ring 1 bridged together through remote peer bridges. The remote peer bridges are connected via a serial line and use TCP as the transport mechanism.

```
Router# debug source bridge

RSRB: remote explorer to 5/192.108.250.1/1996 srn 2 [C840.0021.0050.0000]
RSRB: Version/Ring XReq sent to peer 5/192.108.250.1/1996
RSRB: Received version reply from 5/192.108.250.1/1996 (version 2)
RSRB: DATA: 5/192.108.250.1/1996 Ring Xchg Rep, trn 2, vrn 5, off 18, len 10
RSRB: added bridge 1, ring 1 for 5/192.108.240.1/1996
RSRB: DATA: 5/192.108.250.1/1996 Explorer trn 2, vrn 5, off 18, len 69
RSRB: DATA: 5/192.108.250.1/1996 Forward trn 2, vrn 5, off 0, len 92
RSRB: DATA: forward Forward srn 2, br 1, vrn 5 to peer 5/192.108.250.1/1996
```

The following line indicates that a remote explorer frame has been sent to IP address 192.108.250.1 and, like all RSRB TCP connections, has been assigned port 1996. The bridge belongs to ring group 5. The explorer frame originated from ring 2. The routing information field (RIF) descriptor has been generated by the local station and indicates that the frame was sent out via bridge 1 onto virtual ring 5.

```
RSRB: remote explorer to 5/192.108.250.1/1996 srn 2 [C840.0021.0050.0000]
```

The following line indicates that a request for remote peer information has been sent to IP address 192.108.250.1, TCP port 1996. The bridge belongs to ring group 5.

```
RSRB: Version/Ring XReq sent to peer 5/192.108.250.1/1996
```

The following line is the response to the version request previously sent. The response is sent from IP address 192.108.250.1, TCP port 1996. The bridge belongs to ring group 5.

```
RSRB: Received version reply from 5/192.108.250.1/1996 (version 2)
```

The following line is the response to the ring request previously sent. The response is sent from IP address 192.108.250.1, TCP port 1996. The target ring number is 2, virtual ring number is 5, the offset is 18, and the length of the frame is 10 bytes.

```
RSRB: DATA: 5/192.108.250.1/1996 Ring Xchg Rep, trn 2, vrn 5, off 0, len 10
```

The following line indicates that bridge 1 and ring 1 were added to the source-bridge table for IP address 192.108.250.1, TCP port 1996:

```
RSRB: added bridge 1, ring 1 for 5/192.108.250.1/1996
```

The following line indicates that a packet containing an explorer frame came across virtual ring 5 from IP address 192.108.250.1, TCP port 1996. The packet is 69 bytes in length. This packet is received after the Ring Exchange information was received and updated on both sides.

```
RSRB: DATA: 5/192.108.250.1/1996 Explorer trn 2, vrn 5, off 18, len 69
```

The following line indicates that a packet containing data came across virtual ring 5 from IP address 192.108.250.1 over TCP port 1996. The packet is being placed on the local target ring 2. The packet is 92 bytes in length.

```
RSRB: DATA: 5/192.108.250.1/1996 Forward trn 2, vrn 5, off 0, len 92
```

The following line indicates that a packet containing data is being forwarded to the peer that has IP address 192.108.250.1 address belonging to local ring 2 and bridge 1. The packet is forwarded via virtual ring 5. This packet is sent after the Ring Exchange information was received and updated on both sides.

```
RSRB: DATA: forward Forward srn 2, br 1, vrn 5 to peer 5/192.108.250.1/1996
```

The following is sample output from the **debug source bridge** command for peer bridges using direct encapsulation as a transport mechanism. The RSRB network configuration has ring 1 and ring 2 bridged together through peer bridges. The peer bridges are connected via a serial line and use TCP as the transport mechanism.

```
Router# debug source bridge

RSRB: remote explorer to 5/Serial1 srn 1 [C840.0011.0050.0000]
RSRB: Version/Ring XReq sent to peer 5/Serial1
RSRB: Received version reply from 5/Serial1 (version 2)
RSRB: IFin: 5/Serial1 Ring Xchg, Rep trn 0, vrn 5, off 0, len 10
RSRB: added bridge 1, ring 1 for 5/Serial1
```

The following line indicates that a remote explorer frame was sent to remote peer Serial1, which belongs to ring group 5. The explorer frame originated from ring 1. The RIF descriptor 0011.0050 was generated by the local station and indicates that the frame was sent out via bridge 1 onto virtual ring 5.

```
RSRB: remote explorer to 5/Serial1 srn 1 [C840.0011.0050.0000]
```

The following line indicates that a request for remote peer information was sent to Serial1. The bridge belongs to ring group 5.

```
RSRB: Version/Ring XReq sent to peer 5/Serial1
```

The following line is the response to the version request previously sent. The response is sent from Serial 1. The bridge belongs to ring group 5 and the version is 2.

```
RSRB: Received version reply from 5/Serial1 (version 2)
```

The following line is the response to the ring request previously sent. The response is sent from Serial1. The target ring number is 2, virtual ring number is 5, the offset is 0, and the length of the frame is 39 bytes.

```
RSRB: IFin: 5/Serial1 Ring Xchg Rep, trn 2, vrn 5, off 0, len 39
```

The following line indicates that bridge 1 and ring 1 were added to the source-bridge table for Serial1:

```
RSRB: added bridge 1, ring 1 for 5/Serial1
```

debug source error

To display source-route bridging (SRB) errors, use the **debug source error** privileged EXEC command. The **no** form of this command disables debugging output.

debug source error

no debug source error

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The debug source error command displays some output also found in the **debug source bridge** output. See the **debug source bridge** command for other possible output.

Examples

In all of the following examples of **debug source error** command messages, the variable *number* is the Token Ring interface. For example, if the line of output starts with SRB1, the output relates to the Token Ring 1 interface. SRB indicates a source-route bridging message. RSRB indicates a remote source-route bridging message. SRTLB indicates a source-route translational bridging (SR/TLB) message.

In the following example, a packet of protocol *protocol-type* was dropped:

```
SRBnumber drop: Routed protocol protocol-type
```

In the following example, an Address Resolution Protocol (ARP) packet was dropped. ARP is defined in RFC 826.

```
SRBnumber drop:TYPE_RFC826_ARP
```

In the following example, the current Cisco IOS version does not support Qualified Logical Link Control (QLLC). Reconfigure the router with an image that has the IBM feature set.

```
RSRB: QLLC not supported in version version
Please reconfigure.
```

In the following example, the packet was dropped because the outgoing interface of the router was down:

```
RSRB IF: outgoing interface not up, dropping packet
```

In the following example, the router received an out-of-sequence IP sequence number in a Fast Sequenced Transport (FST) packet. FST has no recovery for this problem like TCP encapsulation does.

```
RSRB FST: bad sequence number dropping.
```

In the following example, the router was unable to locate the virtual interface:

```
RSRB: couldn't find virtual interface
```

In the following example, the TCP queue of the peer router is full. TCPD indicates that this is a TCP debug.

```
RSRB TCPD: tcp queue full for peer
```

In the following example, the router was unable to send data to the *peer* router. A *result* of 1 indicates that the TCP queue is full. A *result* of —1 indicates that the RSRB peer is closed.

```
RSRB TCPD: tcp send failed for peer result
```

In the following example, the routing information identifier (RII) was not set in the explorer packet going forward. The packet will not support SRB, so it is dropped.

```
vrforward_explorer - RII not set
```

In the following example, a packet sent to a virtual bridge in the router did not include a routing information field (RIF) to tell the router which route to use:

```
RSRB: no RIF on packet sent to virtual bridge
```

The following example indicates that the RIF did not contain any information or the length field was set to zero:

```
RSRB: RIF length of zero sent to virtual bridge
```

The following message occurs when the local service access point (LSAP) is out of range. The variable *lsap-out* is the value, *type* is the type of RSRB peer, and *state* is the state of the RSRB peer.

```
VRP: rsrb_lsap_out = lsap-out, type = type, state = state
```

In the following message, the router is unable to find another router with which to exchange bridge protocol data units (BPDUs). BPDUs are exchanged to set up the spanning tree and determine the forwarding path.

```
RSRB(span): BPDU's peer not found
```

Related Commands

Command	Description
debug source bridge	Displays information about packets and frames transferred across a source-route bridge.
debug source event	Displays information on SRB activity.

debug source event

To display information on source-route bridging activity, use the **debug source event** privileged EXEC command. The **no** form of this command disables debugging output.

debug source event

no debug source event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Some of the output from the **debug source bridge** and **debug source error** commands is identical to the output of this command.

Note In order to use the **debug source event** command to display traffic source-routed through an interface, you first must disable fast switching of SRB frames with the **no source bridge route-cache** interface configuration command.

Examples

The following is sample output from the **debug source event** command:

```
Router# debug source event

RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
RSRB0: forward (srn 5 bn 1 trn 10), src: 8110.2222.33c1 dst: 1000.5a59.04f9
[0800.3201.00A1.0050]
```

Table 181 describes the significant fields shown in the display.

Table 181 debug source event Field Descriptions

Field	Description
RSRB0:	Indication that this RIF cache entry is for the Token Ring interface 0, which has been configured for remote source-route bridging (SRB). (SRB1, in contrast, would indicate that this RIF cache entry is for Token Ring 1, configured for SRB.)
forward	Forward (normal data) packet, in contrast to a control packet containing proprietary Cisco bridging information.
srn 5	Ring number of the source ring of the packet.
bn 1	Bridge number of the bridge this packet traverses.
trn 10	Ring number of the target ring of the packet.

Table 181 debug source event Field Descriptions (continued)

Field	Description
src: 8110.2222.33c1	Source address of the route in this RIF cache entry.
dst: 1000.5a59.04f9	Destination address of the route in this RIF cache entry.
[0800.3201.00A1.0050]	RIF string in this RIF cache entry.

In the following example messages, SRBnumber or RSRBnumber denotes a message associated with interface Token Ring number. A number of 99 denotes the remote side of the network.

```
SRBnumber: no path, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

In the preceding example, a bridgeable packet came in on interface Token Ring number but there was nowhere to send it. This is most likely a configuration error. For example, an interface has source bridging turned on, but it is not connected to another source bridging interface or a ring group.

In the following example, a bridgeable packet has been forwarded from Token Ring number to the target ring. The two interfaces are directly linked.

```
SRBnumber: direct forward (srn ring bn bridge trn ring)
```

In the following examples, a proxy explorer reply was not generated because the address could not be reached from this interface. The packet came from the node with the first address.

```
SRBnumber: br dropped proxy XID, address for address, wrong vring (rem)
SRBnumber: br dropped proxy TEST, address for address, wrong vring (rem)
SRBnumber: br dropped proxy XID, address for address, wrong vring (local)
SRBnumber: br dropped proxy TEST, address for address, wrong vring (local)
SRBnumber: br dropped proxy XID, address for address, no path
SRBnumber: br dropped proxy TEST, address for address, no path
```

In the following example, an appropriate proxy explorer reply was generated on behalf of the second address. It is sent to the first address.

```
SRBnumber: br sent proxy XID, address for address[rif]
SRBnumber: br sent proxy TEST, address for address[rif]
```

The following example indicates that the broadcast bits were not set, or that the routing information indicator on the packet was not set:

```
SRBnumber: illegal explorer, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that the direction bit in the RIF field was set, or that an odd packet length was encountered. Such packets are dropped.

```
SRBnumber: bad explorer control, D set or odd
```

The following example indicates that a spanning explorer was dropped because the spanning option was not configured on the interface:

```
SRBnumber: span dropped, input off, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that a spanning explorer was dropped because it had traversed the ring previously:

```
SRBnumber: span violation, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that an explorer was dropped because the maximum hop count limit was reached on that interface:

```
SRBnumber: max hops reached - hop-cnt, s: source-MAC-addr d: dst-MAC-addr rif: rif
```

The following example indicates that the ring exchange request was sent to the indicated peer. This request tells the remote side which rings this node has and asks for a reply indicating which rings that side has.

```
RSRB: sent RingXreq to ring-group/ip-addr
```

The following example indicates that a message was sent to the remote peer. The label variable can be AHDR (active header), PHDR (passive header), HDR (normal header), or DATA (data exchange), and op can be Forward, Explorer, Ring Xchg, Req, Ring Xchg, Rep, Unknown Ring Group, Unknown Peer, or Unknown Target Ring.

```
RSRB: label: sent op to ring-group/ip-addr
```

The following example indicates that the remote bridge and ring pair were removed from or added to the local ring group table because the remote peer changed:

```
RSRB: removing bn bridge rn ring from ring-group/ip-addr
RSRB: added bridge bridge, ring ring for ring-group/ip-addr
```

The following example shows miscellaneous remote peer connection establishment messages:

```
RSRB: peer ring-group/ip-addr closed [last state n]
RSRB: passive open ip-addr(remote port) -> local port
RSRB: CONN: opening peer ring-group/ip-addr, attempt n
RSRB: CONN: Remote closed ring-group/ip-addr on open
RSRB: CONN: peer ring-group/ip-addr open failed, reason[code]
```

The following example shows that an explorer packet was propagated onto the local ring from the remote ring group:

```
RSRBn: sent local explorer, bridge bridge trn ring, [rif]
```

The following messages indicate that the RSRB code found that the packet was in error:

```
RSRBn: ring group ring-group not found
RSRBn: explorer rif [rif] not long enough
```

The following example indicates that a buffer could not be obtained for a ring exchange packet (this is an internal error):

```
RSRB: couldn’t get pak for ringXchg
```

The following example indicates that a ring exchange packet was received that had an incorrect length (this is an internal error):

```
RSRB: XCHG: req/reply badly formed, length pak-length, peer peer-id
```

The following example indicates that a ring entry was removed for the peer; the ring was possibly disconnected from the network, causing the remote router to send an update to all its peers.

```
RSRB: removing bridge bridge ring ring from peer-id ring-type
```

The following example indicates that a ring entry was added for the specified peer; the ring was possibly added to the network, causing the other router to send an update to all its peers.

```
RSRB: added bridge bridge, ring ring for peer-id
```

The following example indicates that no memory was available to add a ring number to the ring group specified (this is an internal error):

```
RSRB: no memory for ring element ring-group
```

The following example indicates that memory was corrupted for a connection block (this is an internal error):

```
RSRB: CONN: corrupt connection block
```

The following example indicates that a connector process started, but that there was no packet to process (this is an internal error):

```
RSRB: CONN: warning, no initial packet, peer: ip-addr peer-pointer
```

The following example indicates that a packet was received with a version number different from the one pre-sent on the router:

```
RSRB: IF New version. local=local-version, remote=remote-version,pak-op-code peer-id
```

The following example indicates that a packet with a bad op code was received for a direct encapsulation peer (this is an internal error):

```
RSRB: IFin: bad op op-code (op code string) from peer-id
```

The following example indicates that the virtual ring header will not fit on the packet to be sent to the peer (this is an internal error):

```
RSRB: vrif_sender, hdr won't fit
```

The following example indicates that the specified peer is being opened. The retry count specifies the number of times the opening operation is attempted.

```
RSRB: CONN: opening peer peer-id retry-count
```

The following example indicates that the router, configured for FST encapsulation, received a version reply to the version request packet it had sent previously:

```
RSRB: FST Rcvd version reply from peer-id (version version-number)
```

The following example indicates that the router, configured for FST encapsulation, sent a version request packet to the specified peer:

```
RSRB: FST Version Request. op = opcode, peer-id
```

The following example indicates that the router received a packet with a bad op code from the specified peer (this is an internal error):

```
RSRB: FSTin: bad op opcode (op code string) from peer-id
```

The following example indicates that the TCP connection between the router and the specified peer is being aborted:

```
RSRB: aborting ring-group/peer-id (vrtcpd_abort called)
```

The following example indicates that an attempt to establish a TCP connection to a remote peer timed out:

```
RSRB: CONN: attempt timed out
```

The following example indicates that a packet was dropped because the ring group number in the packet did not correlate with the ring groups configured on the router:

```
RSRBnumber: ring group ring-group not found
```

debug span

To display information on changes in the spanning-tree topology when debugging a transparent bridge, use the **debug span** privileged EXEC command. The **no** form of this command disables debugging output.

debug span

no debug span

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is useful for tracking and verifying that the spanning-tree protocol is operating correctly.

Examples

The following is sample output from the **debug span** command for an IEEE BPDU packet:

```
Router# debug span

ST: Ether4 000000000000A080002A02D6700000000000A080002A02D67800100001400020 00F00
```

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

```
ST: Ether4 000000000000A080002A02D6700000000000A080002A02D67800100001400020 00F00
          A   B C D E   F          G       H   I          J K L   M   N   O
```

Table 182 describes the significant fields shown in the display.

Table 182 debug span Field Descriptions—IEEE BPDU Packet

Field	Description
ST:	Indication that this is a spanning tree packet.
Ether4	Interface receiving the packet.
(A) 0000	Indication that this is an IEEE BPDU packet.
(B) 00	Version.
(C) 00	Command mode: • 00 indicates config BPDU. • 80 indicates the Topology Change Notification (TCN) BPDU.
(D) 00	Topology change acknowledgment: • 00 indicates no change. • 80 indicates a change notification.
(E) 000A	Root priority.
(F) 080002A02D67	Root ID.
(G) 00000000	Root path cost (0 means the sender of this BPDU packet is the root bridge).
(H) 000A	Bridge priority.

Table 182 debug span Field Descriptions—IEEE BPDU Packet (continued)

Field	Description
(I) 080002A02D67	Bridge ID.
(J) 80	Port priority.
(K) 01	Port Number 1.
(L) 0000	Message age in 256ths of a second (0 seconds, in this case).
(M) 1400	Maximum age in 256ths of a second (20 seconds, in this case).
(N) 0200	Hello time in 256ths of a second (2 seconds, in this case).
(O) 0F00	Forward delay in 256ths of a second (15 seconds, in this case).

The following is sample output from the **debug span** command for a DEC BPDU packet:

```
Router# debug span

ST: Ethernet4 E1190100000200000C01A2C90064008000000C0106CE0A01050F1E6A
```

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

```
E1 19 01 00 0002 00000C01A2C9 0064 0080 00000C0106CE 0A 01 05 0F 1E 6A
A  B  C  D  E    F            G    H    I            J  K  L  M  N  O
```

Table 183 describes the significant fields.

Table 183 debug span Field Descriptions for a DEC BPDU Packet

Field	Description
ST:	Indication that this is a spanning tree packet.
Ethernet4	Interface receiving the packet.
(A) E1	Indication that this is a DEC BPDU packet.
(B) 19	Indication that this is a DEC hello packet. Possible values are as follows: • 0x19—DEC Hello • 0x02—TCN
(C) 01	DEC version.
(D) 00	Flag that is a bit field with the following mapping: • 1—TCN • 2—TCN acknowledgment • 8—Use short timers
(E) 0002	Root priority.
(F) 00000C01A2C9	Root ID (MAC address).
(G) 0064	Root path cost (translated as 100 in decimal notation).
(H) 0080	Bridge priority.
(I) 00000C0106CE	Bridge ID.
(J) 0A	Port ID (in contrast to interface number).

Table 183 debug span Field Descriptions for a DEC BPDU Packet (continued)

Field	Description
(K) 01	Message age (in seconds).
(L) 05	Hello time (in seconds).
(M) 0F	Maximum age (in seconds).
(N) 1E	Forward delay (in seconds).
(O) 6A	Not applicable.

debug sse

To display information for the silicon switching engine (SSE) processor, use the **debug sse** privileged EXEC command. The **no** form of this command disables debugging output.

debug sse

no debug sse

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug sse** command to display statistics and counters maintained by the SSE.

Examples

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

```
Router# debug sse

SSE: IP number of cache entries changed 273 274
SSE: bridging enabled
SSE: interface Ethernet0/0 icb 0x30 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/1 icb 0x33 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/2 icb 0x36 addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/3 icb 0x39 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/4 icb 0x3C addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/5 icb 0x3F addr 0x29 status 0x21A040 protos 0x11
SSE: interface Hssi1/0 icb 0x48 addr 0x122 status 0x421E080 protos 0x11
SSE: cache update took 316ms, elapsed 320ms
```

The following line indicates that the SSE cache is being updated due to a change in the IP fast-switching cache:

```
SSE: IP number of cache entries changed 273 274
```

The following line indicates that bridging functions were enabled on the SSE:

```
SSE: bridging enabled
```

The following lines indicate that the SSE is now loaded with information about the interfaces:

```
SSE: interface Ethernet0/0 icb 0x30 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/1 icb 0x33 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/2 icb 0x36 addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/3 icb 0x39 addr 0x29 status 0x21A040 protos 0x11
SSE: interface Ethernet0/4 icb 0x3C addr 0x29 status 0x21A040 protos 0x10
SSE: interface Ethernet0/5 icb 0x3F addr 0x29 status 0x21A040 protos 0x11
SSE: interface Hssi1/0 icb 0x48 addr 0x122 status 0x421E080 protos 0x11
```

The following line indicates that the SSE took 316 ms of processor time to update the SSE cache. The value of 320 ms represents the total time elapsed while the cache updates were performed.

```
SSE: cache update took 316ms, elapsed 320ms
```

debug standby errors

To display error messages related to Host Standby Router Protocol (HSRP), use the **debug standby errors** command in privileged EXEC mode. To disable the display of these messages, use the **no** form of this command.

debug standby errors

no debug standby errors

Syntax Description

This command has no arguments or keywords

Defaults

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.

Usage Guidelines

You can filter the **debug** output using interface and HSRP group conditional debugging. To enable interface conditional debugging, use the **debug condition interface** command. To enable HSRP conditional debugging, use the **debug condition standby** command.

Examples

The following example enables the display of HSRP errors:

```
debug standby errors
```

Related Commands

Command	Description
debug standby events icmp	Displays HSRP errors.
debug standby events	Displays HSRP events

debug standby events

To display events related to Host Standby Router Protocol (HSRP), use the **debug standby events** command in privileged EXEC mode. To disable the display of these messages, use the **no** form of this command.

debug standby events [[**all**] | [**hsrp** | **redundancy** | **track**]] [**detail**]

no debug standby events

Syntax Description

all	(Optional) Specifies all HSRP events
hsrp	(Optional) Specifies HSRP protocol events
redundancy	(Optional) Specifies HSRP redundancy events
track	(Optional) Specifies HSRP tracking events
detail	(Optional) Specifies detailed debugging information

Defaults

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.

Usage Guidelines

You can filter the **debug** output using interface and HSRP group conditional debugging. To enable interface conditional debugging, use the **debug condition interface** command. To enable HSRP conditional debugging, use the **debug condition standby** command.

Examples

The following example enables the display of all HSRP events:

```
debug standby events all
```

Related Commands

Command	Description
debug standby errors	Displays HSRP errors.
debug standby events icmp	Displays HSRP packets

debug standby events icmp

To display debug messages for the Hot Standby Router Protocol (HSRP) Internet Control Message Protocol (ICMP) redirects filter, use the **debug standby events icmp** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug standby events icmp

no debug standby events icmp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

This command helps you determine whether HSRP is filtering an outgoing ICMP redirect message.

Examples

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

```
Router# debug standby events icmp

10:35:20: SB: changing ICMP redirect sent to 20.0.0.4 for dest 30.0.0.2
10:35:20: SB:   gw 20.0.0.2 -> 20.0.0.12, src 20.0.0.11
10:35:20: SB: Use HSRP virtual address 20.0.0.11 as ICMP src
```

If the router being redirected to is passive (HSRP enabled but no active groups), the following debug message is displayed:

```
10:41:22: SB: ICMP redirect not sent to 20.0.0.4 for dest 40.0.0.3
10:41:22: SB:  20.0.0.3 does not contain an active HSRP group
```

If HSRP could not uniquely determine the gateway used by the host, then the following message is displayed:

```
10:43:08: SB: ICMP redirect not sent to 20.0.0.4 for dest 30.0.0.2
10:43:08: SB: could not uniquely determine IP address for mac 00d0.bbd3.bc22
```

The following messages are also displayed if the **debug ip icmp** command is enabled, in which case the message prefix is changed:

```
10:39:09: ICMP: HSRP changing redirect sent to 20.0.0.4 for dest 30.0.0.2
10:39:09: ICMP:   gw 20.0.0.2 -> 20.0.0.12, src 20.0.0.11
10:39:09: ICMP: Use HSRP virtual address 20.0.0.11 as ICMP src
10:39:09: ICMP: redirect sent to 20.0.0.4 for dest 30.0.0.2, use gw 20.0.0.12
```

Related Commands

Command	Description
debug ip icmp	Displays information on ICMP transactions.

debug standby packets

To display debugging information for packets related to Host Standby Router Protocol (HSRP), use the **debug standby packets** command in privileged EXEC mode. To disable the display of these messages, use the **no** form of this command.

debug standby packets [[**all** | **terse**] | [**hsrp** | **coup** | **hello** | **resign**]] [**detail**]

no debug standby packet

Syntax Description

all	(Optional) Specifies all HSRP packets
terse	(Optional) Specifies all HSRP packets, except hellos and advertisements
hsrp	(Optional) Specifies HSRP packets
coup	(Optional) Specifies HSRP coup packets
hello	(Optional) Specifies HSRP hello packets
resign	(Optional) Specifies HSRP resign packets
detail	(Optional) Specifies HSRP packets in detail

Defaults

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1	This command was introduced.

Usage Guidelines

You can filter the **debug** output using interface and HSRP group conditional debugging. To enable interface conditional debugging, use the **debug condition interface** command. To enable HSRP conditional debugging, use the **debug condition standby** command.

Examples

The following example enables the display of all HSRP packets:

```
debug standby packets all
```

Related Commands

Command	Description
debug standby errors	Displays HSRP errors.
debug standby events	Displays HSRP events

debug stun packet

To display information on packets traveling through the serial tunnel (STUN) links, use the **debug stun packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug stun packet [*group*] [*address*]

no debug stun packet [*group*] [*address*]

Syntax Description

group	(Optional) A decimal integer assigned to a group. Using this option limits output to packets associated with the specified STUN group.
address	(Optional) The output is further limited to only those packets containing the specified STUN address. The *address* argument is in the appropriate format for the STUN protocol running for the specified group.

Usage Guidelines

Because using this command is processor intensive, it is best to use it after regular business hours, rather than in a production environment. It is also best to turn this command on by itself, rather than use it in conjunction with other **debug** commands.

Examples

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

```
router# debug stun packet
X1 type
of packet  STUN sdlc: 0:00:04 Serial3        NDI: (0C2/008) U: SNRM    PF:1
           STUN sdlc: 0:00:04 Serial3        NDI: (0C2/008) U: SNRM    PF:1
           STUN sdlc: 0:00:01 Serial3        SDI: (0C2/008) U: UA      PF:1
X2 type    STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
of packet  STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:000
X3 type    STUN sdlc: 0:00:00 Serial3        NDI: (0C2/008) I:         PF:1  NR:000  NS:000
of packet  STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) I:         PF:1  NR:001  NS:000
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:001
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:001
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:001
           STUN sdlc: 0:00:00 Serial3        SDI: (0C2/008) S: RR      PF:1  NR:001
                                                                                     S2563
```

The following line describes an X1 type of packet:

```
STUN sdlc: 0:00:04 Serial3         NDI: (0C2/008) U: SNRM    PF:1
```

Table 184 describes the significant fields in this line of **debug stun packet** output.

Table 184 debug stun packet Field Descriptions

Field	Description
STUN sdlc:	Indication that the STUN feature is providing the information.
0:00:04	Time elapsed since receipt of the previous packet.
Serial3	Interface type and unit number reporting the event.
NDI:	Type of cloud separating the SDLC end nodes. Possible values are as follows: • NDI—Network input • SDI—Serial link
0C2	SDLC address of the SDLC connection.
008	Modulo value of 8.
U: SNRM	Frame type followed by the command or response type. In this case it is an Unnumbered frame that contains a Set Normal Response Mode (SNRM) command. The possible frame types are as follows: • I—Information frame • S—Supervisory frame. The possible commands and responses are: RR (Receive Ready), RNR (Receive Not Ready), and REJ (Reject). • U—Unnumbered frame. The possible commands are: UI (Unnumbered Information), SNRM, DISC/RD (Disconnect/Request Disconnect), SIM/RIM, XID Exchange Identification), TEST. The possible responses are UA (unnumbered acknowledgment), DM (Disconnected Mode), and FRMR (Frame Reject Mode)
PF:1	Poll/Final bit. Possible values are as follows: • 0—Off • 1—On

The following line of output describes an X2 type of packet:

```
STUN sdlc: 0:00:00 Serial3         SDI: (0C2/008) S: RR      PF:1 NR:000
```

All the fields in the previous line of output match those for an X1 type of packet, except the last field, which is additional. NR:000 indicates a receive count of 0; the range for the receive count is 0 to 7.

The following line of output describes an X3 type of packet:

```
STUN sdlc: 0:00:00 Serial3         SDI: (0C2/008) S:I PF:1 NR:000 NS:000
```

All fields in the previous line of output match those for an X2 type of packet, except the last field, which is additional. NS:000 indicates a send count of 0; the range for the send count is 0 to 7.

debug sw56

To display debug information for switched 56K services, use the **debug sw56** privileged EXEC command.

debug sw56

Syntax Description

This command has no arguments or keywords.

debug syscon perfdata

To display messages related to performance data collection, use the **debug syscon perfdata** privileged EXEC command. The **no** form of this command disables debugging output.

debug syscon perfdata

no debug syscon perfdata

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command is primarily useful to your technical support representative.

Examples

The following is sample output from the **debug syscon perfdata** command. In this example, the CallFail poll group is configured and applied to shelf 1111. The system determines when the next polling cycle should occur and polls the shelf at the appropriate time. The data is stored in the file CallFail.891645120, and an older file is deleted.

```
Router# debug syscon perfdata

PERF: Applying 'CallFail' to shelf 1111
PERF: Setting up objects for SNMP polling: 'CallFail', shelf 1111
PERF: year hours mins secs msecs = 1998 15 11 1 5
PERF: Start 'CallFail' timer, next cycle in 0 mins, 59 secs
PERF: Timer event:  CallFail, 4 minutes
PERF: Polling 'CallFail', shelf 1111, pc 60AEFDF0
PERF: SNMP resp: Type 6, 'CallFail', shelf 1111, error_st 0
PERF: Logged polled data to disk0:/performance/shelf-1111/CallFail.891645120
PERF: Deleted disk0:/performance/shelf-1111/CallFail.891637469
```

debug syscon sdp

To display messages related to the Shelf Discovery Protocol (SDP), use the **debug syscon sdp** privileged EXEC command. The **no** form of this command disables debugging output.

debug syscon sdp

no debug syscon sdp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to display information about SDP packets exchanged between the shelf and the system controller.

Examples

The following sample output from the **debug syscon sdp** command shows the system controller discovering a managed shelf. In the first few lines, the system controller receives a hello packet from shelf 99 at 172.23.66.106. The system controller responds with a hello packet. When the shelf sends another hello packet, the system controller resets the timer and sends another packet.

```
Syscon# debug syscon sdp

SYSCTLR: Hello packet received via UDP from 172.23.66.106
%SYSCTLR-6-SHELF_ADD: Shelf 99 discovered located at address 172.23.66.106
Hello packet sent to the RS located at 172.23.66.106
SYSCTLR: Hello packet received via UDP from 172.23.66.106
Timer for shelf 99 updated, shelf is alive
Hello packet sent to the RS located at 172.23.66.106
```

The following sample output from the **debug syscon sdp** command shows the shelf contacting the system controller. The shelf sends a hello packet to the system controller at 172.23.66.111. The system controller responds with the autoconfiguration commands. The remaining lines show the Hello packets were exchanged between the shelf and the system controller.

```
Shelf# debug syscon sdp

SYSCTLR: Hello packet sent to the SYSCTLR at 172.23.66.111
SYSCTLR: Command packet received from SYSCTLR
Feb 24 17:24:16.713: %SHELF-6-SYSCTLR_ESTABLISHED: Configured via system controller
located at 172.23.66.111
SYSCTLR: Rcvd HELLO from SYSCTLR at 172.23.66.111
SYSCTLR: Hello packet sent to the SYSCTLR at 172.23.66.111
SYSCTLR: Rcvd HELLO from SYSCTLR at 172.23.66.111
```

debug syslog-server

To display information about the syslog server process, use the **debug syslog-server** privileged EXEC command. The **no** form of this command disables debugging output.

debug syslog-server

no debug syslog-server

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command outputs a message every time the syslog server receives a message. It also displays information about subfile creation, removal, and renaming.

Use this command when subfiles are not being created as configured or data is not being written to subfiles. This command is also useful for detecting syslog file size mismatches.

Examples

The following sample display shows output when the following command has been added to the configuration:

```
logging syslog-server 10 3 syslogs
```

This example shows the files being created. Use the **dir disk0:/syslogs.dir** command to display the contents of the newly created directory.

```
Router# debug syslog-server

SYSLOG_SERVER:Syslog file syslogs
SYSLOG_SERVER:Directory disk0:/syslogs.dir created.
SYSLOG_SERVER:Syslog file syslogs created successfully.
```

When a syslog message is received, the router checks to determine if the current file will be too large when the new data is added. In this example, two messages are added to the file.

```
SYSLOG_SERVER: Configured size : 10240 bytes
Current size : 0 bytes
Data size : 68 bytes
New size : 68 bytes
SYSLOG_SERVER: Wrote 68 bytes successfully.
SYSLOG_SERVER: Configured size : 10240 bytes
Current size : 68 bytes
Data size : 61 bytes
New size : 129 bytes
SYSLOG_SERVER: Wrote 61 bytes successfully.
```

Table 185 describes the significant fields shown in the display.

Table 185 debug syslog-server Field Descriptions

Field	Description
Configured size	Maximum subfile size, as set in the **logging syslog-server** command.
Current size	Size of the current subfile before the new message is added.

Table 185 debug syslog-server Field Descriptions (continued)

Field	Description
Data size	Size of the syslog message.
New size	Size of the current subfile after the syslog message is added.

The following output indicates that the current file is too full to fit the next syslog message. The oldest subfile is removed, and the remaining files are renamed. A new file is created and opened for writing syslog messages.

```
SYSLOG_SERVER:Last archive subfile disk0:/syslogs.dir/syslogs.2 removed.
SYSLOG_SERVER: Subfile disk0:/syslogs.dir/syslogs.1 renamed as
disk0:/syslogs.dir/syslogs.2.
SYSLOG_SERVER:subfile disk0:/syslogs.dir/syslogs.cur renamed as
disk0:/syslogs.dir/syslogs.1.
SYSLOG_SERVER:Current subfile disk0:/syslogs.dir/syslogs.cur has been opened.
```

debug tacacs

To display information associated with the TACACS, use the **debug tacacs** privileged EXEC command. The **no** form of this command disables debugging output.

debug tacacs

no debug tacacs

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

TACACS is a distributed security system that secures networks against unauthorized access. Cisco supports TACACS under the authentication, authorization, and accounting (AAA) security system.

Use the **debug aaa authentication** command to get a high-level view of login activity. When TACACS is used on the router, you can use the **debug tacacs** command for more detailed debugging information.

Examples

The following is sample output from the **debug aaa authentication** command for a TACACS login attempt that was successful. The information indicates that TACACS+ is the authentication method used.

```
Router# debug aaa authentication

14:01:17: AAA/AUTHEN (567936829): Method=TACACS+
14:01:17: TAC+: send AUTHEN/CONT packet
14:01:17: TAC+ (567936829): received authen response status = PASS
14:01:17: AAA/AUTHEN (567936829): status = PASS
```

The following is sample output from the **debug tacacs** command for a TACACS login attempt that was successful, as indicated by the status PASS:

```
Router# debug tacacs

14:00:09: TAC+: Opening TCP/IP connection to 192.168.60.15 using source 10.116.0.79
14:00:09: TAC+: Sending TCP/IP packet number 383258052-1 to 192.168.60.15 (AUTHEN/START)
14:00:09: TAC+: Receiving TCP/IP packet number 383258052-2 from 192.168.60.15
14:00:09: TAC+ (383258052): received authen response status = GETUSER
14:00:10: TAC+: send AUTHEN/CONT packet
14:00:10: TAC+: Sending TCP/IP packet number 383258052-3 to 192.168.60.15 (AUTHEN/CONT)
14:00:10: TAC+: Receiving TCP/IP packet number 383258052-4 from 192.168.60.15
14:00:10: TAC+ (383258052): received authen response status = GETPASS
14:00:14: TAC+: send AUTHEN/CONT packet
14:00:14: TAC+: Sending TCP/IP packet number 383258052-5 to 192.168.60.15 (AUTHEN/CONT)
14:00:14: TAC+: Receiving TCP/IP packet number 383258052-6 from 192.168.60.15
14:00:14: TAC+ (383258052): received authen response status = PASS
14:00:14: TAC+: Closing TCP/IP connection to 192.168.60.15
```

The following is sample output from the **debug tacacs** command for a TACACS login attempt that was unsuccessful, as indicated by the status FAIL:

```
Router# debug tacacs

13:53:35: TAC+: Opening TCP/IP connection to 192.168.60.15 using source
192.48.0.79
13:53:35: TAC+: Sending TCP/IP packet number 416942312-1 to 192.168.60.15
(AUTHEN/START)
13:53:35: TAC+: Receiving TCP/IP packet number 416942312-2 from 192.168.60.15
```

```
13:53:35: TAC+ (416942312): received authen response status = GETUSER
13:53:37: TAC+: send AUTHEN/CONT packet
13:53:37: TAC+: Sending TCP/IP packet number 416942312-3 to 192.168.60.15
(AUTHEN/CONT)
13:53:37: TAC+: Receiving TCP/IP packet number 416942312-4 from 192.168.60.15
13:53:37: TAC+ (416942312): received authen response status = GETPASS
13:53:38: TAC+: send AUTHEN/CONT packet
13:53:38: TAC+: Sending TCP/IP packet number 416942312-5 to 192.168.60.15
(AUTHEN/CONT)
13:53:38: TAC+: Receiving TCP/IP packet number 416942312-6 from 192.168.60.15
13:53:38: TAC+ (416942312): received authen response status = FAIL
13:53:40: TAC+: Closing TCP/IP connection to 192.168.60.15
```

Related Commands

Command	Description
debug aaa accounting	Displays information on accountable events as they occur.
debug aaa authentication	Displays information on AAA/TACACS+ authentication.

debug tacacs events

To display information from the TACACS+ helper process, use the **debug tacacs events** privileged EXEC command. The **no** form of this command disables debugging output.

debug tacacs events

no debug tacacs events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use the **debug tacacs events** command only in response to a request from service personnel to collect data when a problem has been reported.

Caution Use the **debug tacacs events** command with caution because it can generate a substantial amount of output.

The TACACS protocol is used on routers to assist in managing user accounts. TACACS+ enhances the TACACS functionality by adding security features and cleanly separating out the authentication, authorization, and accounting (AAA) functionality.

Examples

The following is sample output from the **debug tacacs events** command. In this example, the opening and closing of a TCP connection to a TACACS+ server are shown, and the bytes read and written over the connection and the TCP status of the connection:

```
Router# debug tacacs events

%LINK-3-UPDOWN: Interface Async2, changed state to up
00:03:16: TAC+: Opening TCP/IP to 192.168.58.104/1049 timeout=15
00:03:16: TAC+: Opened TCP/IP handle 0x48A87C to 192.168.58.104/1049
00:03:16: TAC+: periodic timer started
00:03:16: TAC+: 192.168.58.104 req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (ESTAB)
expire=14 AUTHEN/START/SENDAUTH/CHAP queued
00:03:17: TAC+: 192.168.58.104 ESTAB 3BD868 wrote 46 of 46 bytes
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=12 wanted=12 alloc=12 got=12
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=61 wanted=61 alloc=61 got=49
00:03:22: TAC+: 192.168.58.104 received 61 byte reply for 3BD868
00:03:22: TAC+: req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (CLOSEWAIT) expire=9
AUTHEN/START/SENDAUTH/CHAP processed
00:03:22: TAC+: periodic timer stopped (queue empty)
00:03:22: TAC+: Closing TCP/IP 0x48A87C connection to 192.168.58.104/1049
00:03:22: TAC+: Opening TCP/IP to 192.168.58.104/1049 timeout=15
00:03:22: TAC+: Opened TCP/IP handle 0x489F08 to 192.168.58.104/1049
00:03:22: TAC+: periodic timer started
00:03:22: TAC+: 192.168.58.104 req=3BD868 id=299214410 ver=192 handle=0x489F08 (ESTAB)
expire=14 AUTHEN/START/SENDPASS/CHAP queued
00:03:23: TAC+: 192.168.58.104 ESTAB 3BD868 wrote 41 of 41 bytes
00:03:23: TAC+: 192.168.58.104 CLOSEWAIT read=12 wanted=12 alloc=12 got=12
00:03:23: TAC+: 192.168.58.104 CLOSEWAIT read=21 wanted=21 alloc=21 got=9
00:03:23: TAC+: 192.168.58.104 received 21 byte reply for 3BD868
```

```
00:03:23: TAC+: req=3BD868 id=299214410 ver=192 handle=0x489F08 (CLOSEWAIT) expire=13
AUTHEN/START/SENDPASS/CHAP processed
00:03:23: TAC+: periodic timer stopped (queue empty)
```

The TACACS messages are intended to be self-explanatory or for consumption by service personnel only. However, the messages shown are briefly explained in the following text.

The following message indicates that a TCP open request to host 192.168.58.104 on port 1049 will time out in 15 seconds if it gets no response:

```
00:03:16: TAC+: Opening TCP/IP to 192.168.58.104/1049 timeout=15
```

The following message indicates a successful open operation and provides the address of the internal TCP "handle" for this connection:

```
00:03:16: TAC+: Opened TCP/IP handle 0x48A87C to 192.168.58.104/1049
```

The following message indicates that a TACACS+ request has been queued:

```
00:03:16: TAC+: 192.168.58.104 req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (ESTAB)
expire=14 AUTHEN/START/SENDAUTH/CHAP queued
```

The message identifies the following:

- Server that the request is destined for
- Internal address of the request
- TACACS+ ID of the request
- TACACS+ version number of the request
- Internal TCP handle the request uses (which will be zero for a single-connection server)
- TCP status of the connection—which is one of the following:
  - CLOSED
  - LISTEN
  - SYNSENT
  - SYNRCVD
  - ESTAB
  - FINWAIT1
  - FINWAIT2
  - CLOSEWAIT
  - LASTACK
  - CLOSING
  - TIMEWAIT
- Number of seconds until the request times out
- Request type

The following message indicates that all 46 bytes were written to address 192.168.58.104 for request 3BD868:

```
00:03:17: TAC+: 192.168.58.104 ESTAB 3BD868 wrote 46 of 46 bytes
```

The following message indicates that 12 bytes were read in reply to the request:

```
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=12 wanted=12 alloc=12 got=12
```

The following message indicates that 49 more bytes were read, making a total of 61 bytes in all, which is all that was expected:

```
00:03:22: TAC+: 192.168.58.104 CLOSEWAIT read=61 wanted=61 alloc=61 got=49
```

The following message indicates that a complete 61-byte reply has been read and processed for request 3BD868:

```
00:03:22: TAC+: 192.168.58.104 received 61 byte reply for 3BD868 00:03:22: TAC+:
req=3BD868 id=-1242409656 ver=193 handle=0x48A87C (CLOSEWAIT) expire=9
AUTHEN/START/SENDAUTH/CHAP processed
```

The following message indicates that the TACACS+ server helper process switched itself off when it had no more work to do:

```
00:03:22: TAC+: periodic timer stopped (queue empty)
```

Related Commands

Command	Description
debug aaa accounting	Displays information on accountable events as they occur.
debug aaa authentication	Displays information on AAA/TACACS+ authentication.
debug aaa authorization	Displays information on AAA/TACACS+ authorization.
debug sw56	Displays debug information for switched 56 K services.

debug tag-switching adjacency

The **debug tag-switching adjacency** command is replaced by the **debug mpls adjancency** command. See the **debug mpls adjacency** command for more information.

debug tag-switching atm-cos

To display ATM label-VC bind or request activity based on the configuration of a CoS map, use the **debug tag-switching atm-cos** ATM privileged EXEC command.

debug tag-switching atm-cos [*bind* | *request*]

Syntax Description

bind	Specifies debug information about bind responses for a VC path.
request	Specifies debug information about bind requests for a VC path.

Command History

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

Examples

The following is sample output from the **debug tag-switching atm-cos** command.

```
Router# show tag forwarding

Local Outgoing  Prefix      Bytes tag Outgoing  Next Hop
tag  tag or VC  or Tunnel Id   switched  interface
26   28     17.17.17.17/32  0      PO6/0   point2point
27   Pop tag   11.11.11.11/32  1560    PO6/0   point2point
28   27     16.16.16.16/32  0      PO6/0   point2point
29   30     92.0.0.0/8    0     PO6/0   point2point
30   Pop tag   95.0.0.0/8    2600    PO6/0   point2point
31   2/34    10.10.10.10/32  0     AT2/0.1  point2point
32   Pop tag   14.14.14.14/32  0     Fa5/0   91.0.0.1
33   Pop tag   90.0.0.0/8    0     Fa5/0   91.0.0.1
34   Pop tag   96.0.0.0/8    0     Fa5/0   91.0.0.1
    2/36    96.0.0.0/8    0     AT2/0.1  point2point
35   35     93.0.0.0/8    0     PO6/0   point2point
36   36     12.12.12.12/32  0      PO6/0   point2point
37   37     15.15.15.15/32  0      PO6/0   point2point
38   37     18.18.18.18/32  0      Fa5/0   91.0.0.1
39   39     97.0.0.0/8    540    PO6/0   point2point
40   40     98.0.0.0/8    0     PO6/0   point2point

Router# debug tag atm-c
Router# debug tag atm-cos ?
 bind   Bind response for VC path
 request Requests for VC binds path

Router# debug tag atm-cos bind
ATM TAGCOS Bind response debugging is on

Router# debug tag atm-cos request
ATM TAGCOS VC requests debugging is on

Router# configure terminal

Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# interface a2/0.1
Router(config-subif)# tag atm multi
Router(config-subif)# end
Router#
19:59:14:%SYS-5-CONFIG_I:Configured from console by console
```

```
Router#
19:59:24:TAGCOS-REQ:vc request 10.10.10.10/32, available
19:59:24:TAGCOS-REQ:vc request 10.10.10.10/32, standard
19:59:24:TAGCOS-REQ:vc request 10.10.10.10/32, premium
19:59:24:TAGCOS-REQ:vc request 10.10.10.10/32, control
19:59:24:TAGCOS-REQ:vc request 96.0.0.0/8, available
19:59:24:TAGCOS-REQ:vc request 96.0.0.0/8, standard
19:59:24:TAGCOS-REQ:vc request 96.0.0.0/8, premium
19:59:24:TAGCOS-REQ:vc request 96.0.0.0/8, control
TAGCOS-REQ/TCATM:11.11.11.11/32,len=4352,band=1099528405504,class=0x700
TAGCOS-REQ/TCATM:12.12.12.12/32,len=4352,band=2199040033280,class=0x700
TAGCOS-REQ/TCATM:13.13.13.13/32,len=4352,band=3298551661056,class=0x700
TAGCOS-REQ/TCATM:14.14.14.14/32,len=4352,band=4398063288832,class=0x700
TAGCOS-REQ/TCATM:15.15.15.15/32,len=4352,band=5497574916608,class=0x700
TAGCOS-REQ/TCATM:16.16.16.16/32,len=4352,band=6597086544384,class=0x700
TAGCOS-REQ/TCATM:17.17.17.17/32,len=4352,band=7696598172160,class=0x700
TAGCOS-REQ/TCATM:18.18.18.18/32,len=4352,band=8796109799936,class=0x700
TAGCOS-REQ/TCATM:90.0.0.0/8,len=768,band=3940649674539009,class=0x2
TAGCOS-REQ/TCATM:91.0.0.0/8,len=768,band=3940649674604545,class=0x2
TAGCOS-REQ/TCATM:92.0.0.0/8,len=768,band=3940649674670081,class=0x2
TAGCOS-REQ/TCATM:93.0.0.0/8,len=768,band=3940649674735617,class=0x2
TAGCOS-REQ/TCATM:94.0.0.0/8,len=768,band=3940649674801153,class=0x2
TAGCOS-REQ/TCATM:95.0.0.0/8,len=768,band=3940649674866689,class=0x2
TAGCOS-REQ/TCATM:97.0.0.0/8,len=768,band=3940649674932225,class=0x2
TAGCOS-REQ/TCATM:98.0.0.0/8,len=768,band=3940649674997761,class=0x2
TAGCOS-BIND:binding_ok 10.10.10.10/32,VCD=41 - control 41,41,41,41
TAGCOS-BIND:binding_ok 10.10.10.10/32, Inform TFIB pidx=0, in_tag=31, idx=0x80000000
TAGCOS-BIND:binding_ok 96.0.0.0/8,VCD=42 - control 42,42,42,42
TAGCOS-BIND:binding_ok 96.0.0.0/8, Inform TFIB pidx=1, in_tag=34, idx=0x80000001
TAGCOS-BIND:binding_ok 10.10.10.10/32,VCD=43 - premium 43,43,43,41
TAGCOS-BIND:binding_ok 96.0.0.0/8,VCD=44 - premium 44,44,44,42
TAGCOS-BIND:binding_ok 10.10.10.10/32,VCD=45 - standard 45,45,43,41
TAGCOS-BIND:binding_ok 96.0.0.0/8,VCD=46 - standard 46,46,44,42
TAGCOS-BIND:binding_ok 10.10.10.10/32,VCD=47 - available 47,45,43,41
TAGCOS-BIND:binding_ok 96.0.0.0/8,VCD=48 - available 48,46,44,42
72k-41-5#
72k-41-5#
```

Related Commands

Command	Description
debug tag atm-tdp	Debugs label-controlled ATM TDP.
debug tag packets	Debugs tag switching packets.
debug tag tdp	Debugs tag distribution protocol items and information.

debug tag-switching atm-tdp api

To display information about the VCI allocation of tag VCs (TVCs), free, and cross-connect requests, use the **debug tag-switching atm-tdp api** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching atm-tdp api

no debug tag-switching atm-tdp api

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

You can use the **debug tag-switching atm-tdp api** command with the **debug tag-switching atm-tdp states** command to display more complete information about a TVC.

Examples

The following is sample output from the **debug tag-switching atm-tdp api** command:

```
Router# debug tag-switching atm-tdp api

Tailend Router Free tag Req 167.50.0.0 on ATM0/0.2 VPI/VCI 1/674
 TAGATM_API: received tag free request
       interface: ATM0/0.2 dir: in vpi: 1 vci: 674
 TAGATM_API: completed tag free
       interface: ATM0/0.2 vpi: 1 vci: 674
       result: TAGATM_OK
```

Table 186 describes the significant fields shown in the display.

Table 186 debug tag-switching atm-tdp api Field Descriptions

Field	Description
TAGATM_API	Subsystem that prints the message.
interface	Interface used by the driver to allocate or free VPI/VCI resources.
dir	Direction of the VC: • In—Input or receive VC • Out—Output VC
vpi	Virtual path identifier.
vci	Virtual channel identifier.
result	Return error code from the driver API.

Related Commands

Command	Description
debug tag-switching atm-tdp states	Displays information about TVC state transitions as they occur.

debug tag-switching atm-tdp routes

To display information about the state of the routes for which VCI requests are being made, use the **debug tag-switching atm-tdp routes** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching atm-tdp routes

no debug tag-switching atm-tdp routes

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

When there are many routes and system activities (that is, shutting down interfaces, learning of new routes, and so on), the **debug tag-switching atm-tdp routes** command displays a substantial amount of information that may interfere with system timing. Most commonly, this affects the normal operation of the Tag Distribution Protocol (TDP). You should increase the TDP hold-time value by using the **tag-switching tdp holdtime** command.

Examples

The following is sample output from the **debug tag-switching atm-tdp routes** command:

```
Router# debug tag-switching atm-tdp routes

CleanupRoutes,not deleting route of idb ATM0/0.2,rdbIndex 0
tcatmFindRouteTags,153.7.0.0/16,idb=ATM0/0.2,nh=134.111.102.98,index=0
AddNewRoute,153.7.0.0/16,idb=ATM0/0.2
CleanupRoutes,153.7.0.0/16
CleanupRoutes,not deleting route of idb ATM0/0.2,rdbIndex 0
tcatmFindRouteTags,153.8.0.0/16,idb=ATM0/0.2,nh=134.111.102.98,index=0
AddNewRoute,153.8.0.0/16,idb=ATM0/0.2
CleanupRoutes,153.8.0.0/16
CleanupRoutes,not deleting route of idb ATM0/0.2,rdbIndex 0
tcatmFindRouteTags,153.9.0.0/16,idb=ATM0/0.2,nh=134.111.102.98,index=0
AddNewRoute,153.9.0.0/16,idb=ATM0/0.2
CleanupRoutes,153.9.0.0/16
CleanupRoutes,not deleting route of idb ATM0/0.2,rdbIndex 0
tcatmFindRouteTags,153.10.0.0/16,idb=ATM0/0.2,nh=134.111.102.98,index=0
AddNewRoute,153.10.0.0/16,idb=ATM0/0.2
CleanupRoutes,153.10.0.0/16
CleanupRoutes,not deleting route of idb ATM0/0.2,rdbIndex 0
tcatmFindRouteTags,153.11.0.0/16,idb=ATM0/0.2,nh=134.111.102.98,index=0
AddNewRoute,153.11.0.0/16,idb=ATM0/0.2
CleanupRoutes,153.11.0.0/16
```

Table 187 describes the significant fields shown in the display.

Table 187 debug tag-switching atm-tdp routes Field Descriptions

Field	Description
CleanupRoutes	Cleans up the routing table after a route has been deleted.
not deleting route of idb ATM0/0.2	Route cleanup event has not removed the specified route.
rdbIndex	Index identifying the route.
tcatmFindRouteTags	Request a VC for the route.
idb	Internal descriptor for an interface.
nh	Next hop for the route.
index	Identifier for the route.
AddNewRoute	Action of adding routes for a prefix or address.

debug tag-switching atm-tdp states

To display information about TVC state transitions as they occur, use the **debug tag-switching atm-tdp states** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching atm-tdp states

no debug tag-switching atm-tdp states

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

When there are many routes and system activities (that is, shutting down interfaces, learning of new routes, and so on), the **debug tag-switching atm-tdp states** command outputs a substantial amount of information that may interfere with system timing. Most commonly, this affects the normal operation of the Tag Distribution Protocol (TDP). You should increase the TDP hold-time value by using the **tag-switching tdp holdtime** command.

Examples

The following is sample output from the **debug tag-switching atm-tdp states** command:

```
Router# debug tag-switching atm-tdp states

Transit Output 166.35.0.0 VPI/VCI 1/67 Active -> XmitRelease NoPath
Transit Input 166.35.0.0 VPI/VCI 1/466 Active -> ApiWaitParentLoss ParentLoss
Transit Input 166.35.0.0 VPI/VCI 1/466 ApiWaitParentLoss -> ParentWait ApiSuccess
Transit Input 166.35.0.0 VPI/VCI 1/466 ParentWait -> XmitWithdraw NoPath
Transit Input 166.35.0.0 VPI/VCI 1/466 XmitWithdraw -> XmitWithdraw Transmit
Transit Input 166.35.0.0 VPI/VCI 1/466 XmitWithdraw -> NonExistent Release
Transit Input 166.35.0.0 VPI/VCI 1/466 NonExistent -> NonExistent ApiSuccess
```

Table 188 describes the significant fields shown in the display.

Table 188 debug tag-switching atm-tdp states Field Descriptions

Field	Description
Transit Output	Output side of a TVC.
VPI/VCI	VC value.
Transit Input	Input side of a TVC.

debug tag-switching packets

The **debug tag-switching packets** command is replaced by the **debug mpls packets** command. See the **debug mpls packets** command for more information.

debug tag-switching tdp advertisements

To print information about the advertisement of tags and interface addresses to TDP peer devices, use the **debug tag-switching tdp advertisements** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp advertisements

no debug tag-switching tdp advertisements

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug tag-switching tdp advertisements** command:

```
Router# debug tag-switching tdp advertisements

tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 99.101.0.8
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 172.27.32.28
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 10.105.0.8
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 10.92.0.8
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 10.205.0.8
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 210.8.0.8
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 10.105.0.0/16, tag 1 (#2)
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 10.102.0.0/16, tag 26 (#4)
tagcon: adj 210.9.0.9:0 (pp 0x60D8E98C): advertise 10.227.0.0/16, tag 27 (#6)
```

Table 189 describes the significant fields shown in the display.

Table 189 debug tag-switching tdp advertisements Field Descriptions

Field	Description
tagcon:	Identifies the source of the message as the tag control subsystem.
adj <*a.b.c.d:e*>	TDP identifier of the peer device to which the advertisement has been made.
(pp 0x*nnnnnnnn*)	Identifier for the data structure used to represent the peer device at the tag distribution level. Useful for correlating debug output.
advertise *X*	What was advertised to the peer device—either an interface address ("*a.b.c.d*") or tag binding ("*a.b.c.d/m*, tag *t* (#*n*)").
(#n)	For a tag binding advertisement, the sequence number of the tag information base (TIB) modification that made it necessary to advertise the tag.

Related Commands

Command	Description
show tag-switching tdp neighbors	Displays the status of TDP sessions.

debug tag-switching tdp bindings

To print information about changes to the tag information base (TIB) used to keep track of tag bindings learned from TDP peer devices through TDP downstream tag distribution, use the **debug tag-switching tdp bindings** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp bindings

no debug tag-switching tdp bindings

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug tag-switching tdp bindings** command:

```
Router# debug tag-switching tdp bindings

tagcon: tibent(10.105.0.0/16): created; find route tags request
tagcon: tibent(10.105.0.0/16): lcl tag 1 (#2) assigned
tagcon: tibent(10.102.0.0/16): created; find route tags request
tagcon: tibent(10.102.0.0/16): lcl tag 26 (#4) assigned
tagcon: 210.9.0.9:0: 99.101.0.9 added to addr<->tdp ident map
tagcon: 210.9.0.9:0: 172.27.32.29 added to addr<->tdp ident map
tagcon: 210.9.0.9:0: 10.105.0.9 added to addr<->tdp ident map
tagcon: tibent(172.27.32.0/22): rem tag 1 from 210.9.0.9:0 added
tagcon: tibent(200.26.0.0/16): rem tag 30 from 210.9.0.9:0 added
tagcon: tibent(210.8.0.8/32): created; remote tag learned
tagcon: tibent(210.8.0.8/32): rem tag 31 from 210.9.0.9:0 added
```

Table 190 describes the significant fields shown in the display.

Table 190 debug tag-switching tdp bindings Field Descriptions

Field	Description
tagcon:	Identifies the source of the message as the tag control subsystem.
tibent(*network/mask*)	Destination that has a tag binding change.
created; *reason*	TIB entry has been created for the specified destination for the indicated reason.
rem tag ...	Describes a change to the tag bindings for the specified destination. The change is for a tag binding learned from the specified TDP peer device.
lcl tag ...	Describes a change to a locally assigned (incoming) tag for the specified destination.
(#n)	Sequence number of the modification to the TIB corresponding to the local tag change.
a.b.c.d:n: e.f.g.h added to addr<->tdp ident map	Address *e.f.g.h* has been added to the set of addresses associated with TDP identifier *a.b.c.d:n*.

Related Commands

Command	Description
show tag-switching tdp bindings	Displays the contents of the TIB.

debug tag-switching tdp directed-neighbors

To print information about the directed neighbor mechanism, use the **debug tag-switching tdp directed-neighbors** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp directed-neighbors

no debug tag-switching tdp directed-neighbors

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This mechanism establishes TDP adjacencies to peer devices that are not directly adjacent, such as peer devices at either end of a tunnel.

The directed neighbor mechanism starts TDP discovery between two TSRs that are not necessarily directly adjacent. This mechanism is used, for instance, to support two-level tagging across a TSP tunnel, and to support traffic engineering metric exchange across a TSP tunnel.

The mechanism is based on an IP address, such as the IP address of the last hop of a TSP tunnel. A TSR wanting to establish a TDP adjacency to some other TSR with a given IP address is the active TSR for that directed neighbor discovery. A TSR willing to respond to that discovery is the passive TSR for that discovery.

As with TDP discovery between adjacent TSRs, it is possible to have multiple directed neighbor discovery sessions can run between two TSRs, all supporting a single TDP adjacency.

The debug messages track discovery changes, such as discovery or loss of a directed neighbor. As a detail reflected in the debug prints, discovery of a directed neighbor with IP address X is complete when a TDP adjacency comes up and the far end announces that IP address X is one of its IP addresses.

Examples

The following is sample output from the **debug tag-switching tdp directed-neighbors** command:

```
Router# debug tag-switching tdp directed-neighbors

tdp_directednbr: TDPDirAdj 10.11.10.11 received address addition notification
tdp_directednbr: TDPDirAdj 10.11.10.11 TDP peer set
tdp_directednbr: TDPDirAdj 10.11.10.11 received address deletion notification
tdp_directednbr: TDPDirAdj 10.11.10.11 peer cleared
```

Table 191 describes the significant fields shown in the display.

Table 191 debug tag-switching tdp directed-neighbors Field Descriptions

Field	Description
tdp_directednbr:	Identifies this as a TDP directed neighbor debug statement.
TDPDirAdj <*address*>	Identifies the IP address to which a TDP adjacency is desired.

Related Commandss

Command	Description
show tag-switching tdp neighbors	Displays the status of TDP sessions.

debug tag-switching tdp peer state-machine

To print information about state transitions at the tag distribution level, use the **debug tag-switching tdp peer state-machine** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp peer state-machine

no debug tag-switching tdp peer state-machine

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

TDP sessions are supported by data structures and state machines at three levels:

- Transport—The transport level establishes and maintains TCP connections used to support TDP sessions.
- Protocol—The protocol level implements the TDP session setup protocol, and constructs and parses TDP PDUs and PIEs.
- Tag distribution—The tag distribution level uses TDP sessions to exchange tags with TDP peer devices.

The **debug tag-switching tdp transport** command provides visibility of activity at the transport level, the **debug tag-switching tdp session** command at the protocol level, and the **debug tag-switching tdp peer** command at the tag distribution level.

Examples

The following is sample output from the **debug tag-switching tdp peer state-machine** command:

```
Router# debug tag-switching tdp peer state-machine

tagcon: start TDP TCP timers for 202.0.0.1:1 (pp 0x60D8ABC8)
tagcon: adj 202.0.0.1:1-1 (pp 0x60D8ABC8): Event unsol open
    unsol op pdg -> estab
tagcon: start TDP TCP timers for 210.9.0.9:0 (pp 0x60D93608)
tagcon: adj 210.9.0.9:0 (pp 0x60D93608): Event unsol open
    unsol op pdg -> estab
tagcon: adj 210.9.0.9:0 (pp 0x60D93608): Event down
    estab -> dstroy
tagcon: adj 202.0.0.1:1 (pp 0x60D8ABC8): Event down
    estab -> dstroy
tagcon: start TDP TCP timers for 202.0.0.1:1 (pp 0x60DAC678)
tagcon: adj 202.0.0.1:1-1 (pp 0x60DAC678): Event unsol open
    unsol op pdg -> defrd
tagcon: start TDP TCP timers for 210.9.0.9:0 (pp 0x60D895C4)
tagcon: adj 210.9.0.9:0 (pp 0x60D895C4): Event unsol open
    unsol op pdg -> defrd
tagcon: adj 210.9.0.9:0 (pp 0x60D93608): Event cleanup done
    dstroy -> non-ex
tagcon: adj 210.9.0.9:0 (pp 0x60D895C4): Event undefer
    defrd -> estab
tagcon: adj 202.0.0.1:1 (pp 0x60D8ABC8): Event cleanup done
    dstroy -> non-ex
tagcon: adj 202.0.0.1:1-1 (pp 0x60DAC678): Event undefer
    defrd -> estab
```

Table 192 describes the significant fields shown in the display.

Table 192 debug tag-switching tdp peer state-machine Field Descriptions

Field	Description
tagcon:	Identifies the source of the message as the tag control subsystem.
adj *a.b.c.d:e*	TDP identifier of the peer device for the session with the state change.
(pp 0x*nnnnnnnn*)	Address of the data structure used to represent the peer device at the tag distribution level. It is useful for correlating debug output.
Event E	Event causing the state change.
S1 -> S2	State of the TDP session has changed from state S1 to state S2.

debug tag-switching tdp pies received

To print information about TDP protocol information elements (PIEs) received from TDP peer devices, use the **debug tag-switching tdp pies received** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp pies received [**all**]

no debug tag-switching tdp pies received [**all**]

Syntax Description

all	(Optional) TDP received PIEs, including periodic keepalive PIEs.

Usage Guidelines

TDP requires periodic transmission of keepalive PIEs. If you do not specify the **all** option, periodic keepalive PIEs are not displayed.

Examples

The following is sample output from the **debug tag-switching tdp pies received** command:

```
Router# debug tag-switching tdp pies received all

tdp: Rcvd open PIE from 202.0.0.1 (pp 0x0)
tdp: Rcvd keep_alive PIE from 202.0.0.1:1 (pp 0x0)
tdp: Rcvd request_bind PIE from 202.0.0.1:1 (pp 0x60DAC678)
tdp: Rcvd request_bind PIE from 202.0.0.1:1 (pp 0x60DAC678)
tdp: Rcvd open PIE from 210.9.0.9 (pp 0x0)
tdp: Rcvd keep_alive PIE from 210.9.0.9:0 (pp 0x0)
tdp: Rcvd bind PIE from 202.0.0.1:1 (pp 0x60DAC678)
tdp: Rcvd bind PIE from 202.0.0.1:1 (pp 0x60DAC678)
```

Table 193 describes the significant fields shown in the display.

Table 193 debug tag-switching tdp pies received all Field Descriptions

Field	Description
tdp:	Identifies the source of the message as TDP.
Rcvd *xxx* PIE	Type of PIE received.
from *a.b.c.d*	Host that sent the PIE. Used in the early stages of the opening of a TDP session, when the TDP identifier is not yet known.
from a.b.c.d:e	TDP identifier of the peer device that sent the PIE.
(pp 0x*nnnnnnnn*)	Identifies the data structure used to represent the peer device at the tag distribution level. Useful for correlating debug output.

Related Commands

Command	Description
debug tag-switching tdp pies sent	Prints information about state transitions at the tag distribution level.

debug tag-switching tdp pies sent

To print information about state transitions at the tag distribution level, use the **debug tag-switching tdp pies sent** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp pies sent [**all**]

no debug tag-switching tdp pies sent [**all**]

Syntax Description

all	(Optional) TDP sent PIEs, including periodic keepalive PIEs.

Usage Guidelines

TDP requires periodic transmission of keepalive PIEs. If you do not specify the **all** option, periodic keepalive PIEs are not displayed.

Examples

The following is sample output from the **debug tag-switching tdp pies sent all** command:

```
Router# debug tag-switching tdp pies sent all

tdp: Queued open PIE to 210.222.0.222:1 (pp 0x0)
tdp: Sent open PIE to 210.222.0.222:1 (pp 0x0)
tdp: Queued keep_alive PIE to 210.222.0.222:1 (pp 0x0)
tdp: Sent keep_alive PIE to 210.222.0.222:1 (pp 0x0)
tdp: Queued request_bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Sent request_bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Queued request_bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Sent request_bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Queued open PIE to 210.8.0.8 (pp 0x0)
tdp: Queued bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Sent bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Queued bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Sent bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Queued bind PIE to 210.222.0.222:1 (pp 0x60F264C8)
tdp: Queued open PIE to 210.8.0.8 (pp 0x0)
tdp: Sent open PIE to 210.8.0.8 (pp 0x0)
tdp: Queued keep_alive PIE to 210.8.0.8:0 (pp 0x0)
tdp: Sent keep_alive PIE to 210.8.0.8:0 (pp 0x0)
tdp: Queued address PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Sent address PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Queued bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Queued bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Queued bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Queued bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Queued bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Sent bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Sent bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Sent bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Sent bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
tdp: Sent bind PIE to 210.8.0.8:0 (pp 0x60F161AC)
```

Table 194 describes the significant fields shown in the display.

Table 194 debug tag-switching tdp sent all Field Descriptions

Field	Description
tdp:	Identifies the source of the message as TDP.
Queued *xxx* PIE	Indicates that a PIE of the specified type has been queued for transmission.
Sent *xxx* PIE	Indicates that a PIE of the specified type has been sent on the TDP session TCP connection.
to *a.b.c.d*	Host to which the PIE has been sent or for which it has been queued. Used in the early stages of opening a TDP session when the TDP identifier is not yet known.
to *a.b.c.d:e*	TDP identifier of the peer device to which the PIE has been sent or for which it has been queued.
(pp 0x*nnnnnnnn*)	Identifies the data structure used to represent the peer device at the tag distribution level. Useful for correlating debug output.

Related Commands

Command	Description
debug tag-switching tdp pies received	Prints information about TDP PIEs received from TDP peer devices.
debug tag-switching tdp session io	Prints the contents of TDP PIEs sent to and received from TDP peer devices.

debug tag-switching tdp session io

To print the contents of TDP PIEs sent to and received from TDP peer devices, use the **debug tag-switching tdp session io** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp session io [**all**]

no debug tag-switching tdp session io [**all**]

Syntax Description

all	(Optional) TDP session I/O activity, including I/O for periodic keepalives.

Usage Guidelines

TDP sessions are supported by data structures and state machines at three levels:

- Transport—The transport level establishes and maintains TCP connections used to support TDP sessions.
- Protocol—The protocol level implements the TDP session setup protocol, and constructs and parses TDP PDUs and PIEs.
- Tag distribution—The tag distribution level uses TDP sessions to exchange tags with TDP peer devices.

The **debug tag-switching tdp transport** command provides visibility of activity at the transport level, the **debug tag-switching tdp session** command at the protocol level, and the **debug tag-switching tdp peer** command at the tag distribution level.

TDP requires periodic transmission of keepalive PIEs. If you do not specify the **all** option, periodic keepalive PIEs are not displayed.

Examples

The following is sample output from the **debug tag-switching tdp session io all** command:

```
Router# debug tag-switching tdp session io all

tdp: Rcvd open PIE from 210.9.0.9 (pp 0x0)
tdp: TDP open PIE: PDU hdr: TDP Id: 210.9.0.9:0; PIE Contents:
 0x00 0x01 0x00 0x10 0xD2 0x09 0x00 0x09 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x04
 0x01 0x00 0x00 0x1E
tdp: Sent open PIE to 210.9.0.9:0 (pp 0x0)
tdp: TDP open PIE: PDU hdr: TDP Id: 172.27.32.28:0; PIE Contents:
 0x00 0x01 0x00 0x10 0xAC 0x1B 0x20 0x1C 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x04
 0x01 0x00 0x00 0x0F
tdp: Sent keep_alive PIE to 210.9.0.9:0 (pp 0x0)
tdp: TDP keep_alive PIE: PDU hdr: TDP Id: 172.27.32.28:0; PIE Contents:
 0x00 0x01 0x00 0x0C 0xAC 0x1B 0x20 0x1C 0x00 0x00 0x00 0x00 0x05 0x00 0x00 0x00
tdp: Rcvd keep_alive PIE from 210.9.0.9:0 (pp 0x0)
tdp: TDP keep_alive PIE: PDU hdr: TDP Id: 210.9.0.9:0; PIE Contents:
 0x00 0x01 0x00 0x0C 0xD2 0x09 0x00 0x09 0x00 0x00 0x00 0x00 0x05 0x00 0x00 0x00
tdp: Rcvd address PIE from 210.9.0.9:0 (pp 0x60E109F0)
tdp: TDP address PIE: PDU hdr: TDP Id: 210.9.0.9:0; PIE Contents:
 0x00 0x01 0x00 0x35 0xD2 0x09 0x00 0x09 0x00 0x00 0x00 0x00 0x08 0x00 0x00 0x29
 0x00 0x01 0x00 0x03 0x00 0x23 0x20 0x63 0x65 0x00 0x09 0x20 0xAC 0x1B 0x20 0x1D
 0x20 0x0A 0x69 0x00 0x09 0x20 0x0A 0x5C 0x00 0x09 0x20 0x0A 0x6F 0x00 0x09 0x20
 0x0A 0xCD 0x00 0x09 0x20 0xD2 0x09 0x00 0x09
tdp: Rcvd bind PIE from 210.9.0.9:0 (pp 0x60E109F0)
```

```
tdp: TDP bind PIE: PDU hdr: TDP Id: 210.9.0.9:0; PIE Contents:
 0x00 0x01 0x00 0xFC 0xD2 0x09 0x00 0x09 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0xF0
 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x02 0x00 0xE6 0x00 0x00 0x00 0x00 0x01 0x10
 0x0A 0x6F 0x00 0x00 0x00 0x00 0x01 0x16 0xAC 0x1B 0x20 0x00 0x00 0x00 0x00 0x01
 0x10 0xD2 0x09 0x00 0x00 0x00 0x00 0x1A 0x20 0x0A 0x0B 0x00 0x0B 0x00 0x00 0x00
```

Table 195 describes the significant fields shown in the display.

Table 195 debug tag-switching tdp session io Field Descriptions

Field	Description
tdp:	Identifies the source of the message as TDP.
Rcvd *xxx* PIE	Indicates that a PIE of the specified type has been received.
from *a.b.c.d*	Host to which the PIE has been sent. Used in the early stages of the opening of a TDP session when the TDP identifier is not yet known.
Sent *xxx* PIE	Indicates that a PIE of the specified type has been sent.
to *a.b.c.d*	Host to which the PIE has been sent. Used in the early stages of opening a TDP session when the TDP identifier is not yet known.
to *a.b.c.d:e*	TDP identifier of the peer device to which the PIE has been sent.
(pp 0x*nnnnnnnn*)	Identifies the data structure used to represent the peer device at the tag distribution level. Useful for correlating debug output.
--TDP *xxx* PIE	Type of PIE that has been sent.
PDU_hdr: TDP Id: *a.b.c.d:e*	TDP identifier of the sender included in the TDP PDU header.
PIE contents: 0x*nn* ... 0x*nn*	Contents of the PIE represented as a sequence of bytes.

Related Commands

Command	Description
debug tag-switching tdp pies received	Prints information about TDP PIEs received from TDP peer devices.
debug tag-switching tdp pies sent	Prints information about state transitions at the tag distribution level.

debug tag-switching tdp session state-machine

To print information about state transitions at the protocol level, use the **debug tag-switching tdp session state-machine** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp session state-machine

no debug tag-switching tdp session state-machine

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

TDP sessions are supported by data structures and state machines at three levels:

- Transport—The transport level establishes and maintains TCP connections used to support TDP sessions.
- Protocol—The protocol level implements the TDP session setup protocol, and constructs and parses TDP PDUs and PIEs.
- Tag distribution—The tag distribution level uses TDP sessions to exchange tags with TDP peer devices.

The **debug tag-switching tdp transport** command provides visibility of activity at the transport level, the **debug tag-switching tdp session** command at the protocol level, and the **debug tag-switching tdp peer** command at the tag distribution level.

Examples

The following is sample output from the **debug tag-switching tdp session state-machine** command:

```
Router# debug tag-switching tdp session state-machine

tdp: adj:210.9.0.9(0x60DDBB4C): Event: Xport opened;
   Non-existent -> Init pasv
tdp: tdp_create_ptcl_adj: tp = 0x60DDBB4C, ipaddr = 210.9.0.9
tdp: adj:210.9.0.9(0x60DDBB4C): Event: Xport opened;
   Init pasv -> Init pasv
tdp: adj:10.105.0.9(0x60DDBB4C): Event: Rcv TDP Open;
   Init pasv -> Open rcvd pasv
tdp: adj:10.105.0.9(0x60DDBB4C): Event: Rcv TDP KA;
   Open rcvd pasv -> Oper
tdp: adj:unknown(0x60DDBB4C): Event: Xport closed;
   Oper -> Non-existent
```

Table 196 describes the significant fields shown in the display.

Table 196 debug tag-switching tdp session state-machine Field Descriptions

Field	Description
tdp:	Identifies the source of the message as TDP.
adj:*a.b.c.d*	Identifies the network address of the TDP peer device.
(0x*nnnnnnnn*)	Identifies the data structure used to represent the peer device at the protocol level. Useful for correlating debug output.

Table 196 debug tag-switching tdp session state-machine Field Descriptions (continued)

Field	Description
Event: E	Event that caused the state transition.
S1 -> S2	State of the TDP session has changed from state S1 to state S2.

debug tag-switching tdp transport connections

To print information about the TCP connections used to support TDP sessions, use the **debug tag-switching tdp transport connections** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp transport connections

no debug tag-switching tdp transport connections

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

TDP sessions are supported by data structures and state machines at three levels:

- Transport—The transport level establishes and maintains TCP connections used to support TDP sessions.
- Protocol—The protocol level implements the TDP session setup protocol, and constructs and parses TDP PDUs and PIEs.
- Tag distribution—The tag distribution level uses TDP sessions to exchange tags with TDP peer devices.

The **debug tag-switching tdp transport** command provides visibility of activity at the transport level, the **debug tag-switching tdp session** command at the protocol level, and the **debug tag-switching tdp peer** command at the tag distribution level.

When two devices establish a TCP connection for a TDP session, the device with the larger transport address plays an active role and the other plays a passive role. The active device attempts to establish a TCP connection to the well-known TDP port at the passive device. The passive device waits for the connection to the well-known port to be established.

Examples

The following is sample output from the **debug tag-switching transport connections** command:

```
Router# debug tag-switching tdp transport connections

Debug output at active peer:

tdp: Opening conn; adj 0x60F7C604, 210.9.0.9 <-> 172.27.32.28
tdp: Conn is up; adj 0x60F7C604, 210.9.0.9:11018 <-> 172.27.32.28:711
tdp: hold-timer expired for adj 0x60F7C604, will close conn
tdp: Closing conn 210.9.0.9:11018 <-> 172.27.32.28:711, adj 0x60F7C604

Debug output at passive peer:

tdp: Incoming conn 172.27.32.28:711 <-> 210.9.0.9:11018
tdp: Conn closed by peer; adj 0x60EB5FD4
   172.27.32.28:711 <-> 210.9.0.9:11018, Ethernet1/1/1
tdp: Closing conn 172.27.32.28:711 <-> 210.9.0.9:11018, adj 0x60EB5FD4
```

Table 197 describes the significant fields shown in the display.

Table 197 debug tag-switching tdp transport connections Field Descriptions

Field	Description
tdp:	Identifies the source of the message as TDP.
adj 0x*nnnnnnnn*	Identifies the data structure used to represent the peer device at the transport level. Useful for correlating debug output.
a.b.c.d -> *p.q.r.s*	Indicates a TCP connection between a.b.c.d and p.q.r.s.
a.b.c.d:x -> *p.q.r.s:y*	Indicates a TCP connection between a.b.c.d, port x and p.q.r.s, port y.

Related Commands

Command	Description
debug tag-switching tdp transport events	Prints information about the events related to the TDP peer discovery mechanism, which is used to determine the devices with which to establish TDP sessions.

debug tag-switching tdp transport events

To print information about the events related to the TDP peer discovery mechanism, which is used to determine the devices with which to establish TDP sessions, use the **debug tag-switching tdp transport events** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp transport events

no debug tag-switching tdp transport events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

TDP sessions are supported by data structures and state machines at three levels:

- Transport—The transport level establishes and maintains TCP connections used to support TDP sessions.
- Protocol—The protocol level implements the TDP session setup protocol, and constructs and parses TDP PDUs and PIEs.
- Tag distribution—The tag distribution level uses TDP sessions to exchange tags with TDP peer devices.

The **debug tag-switching tdp transport** command provides visibility of activity at the transport level, the **debug tag-switching tdp session** command at the protocol level, and the **debug tag-switching tdp peer** command at the tag distribution level.

Examples

The following is sample output from the **debug tag-switching tdp transport events** command:

```
Router# debug tag-switching tdp transport events

tdp: Rcvd hello; Ethernet1/1/1, from 10.105.0.9 (210.9.0.9:0), intf_id 0, opt 0x4
tdp: Hello from 10.105.0.9 (210.9.0.9:0) to 255.255.255.255, opt 0x4
tdp: New adj 0x60DF6E50 from 10.105.0.9 (210.9.0.9:0), Ethernet1/1/1
tdp: Rcvd hello; ATM3/0.1, from 200.26.0.4 (202.0.0.1:1), intf_id 1, opt 0x4, tcatm
tdp: Rcvd hello; Ethernet1/1/1, from 10.105.0.9 (210.9.0.9:0), intf_id 0, opt 0x4
tdp: Hello from 10.105.0.9 (210.9.0.9:0) to 255.255.255.255, opt 0x4
tdp: Ignore Hello Timer for Ethernet1/1/1; intf not TDP ready
tdp: Send hello; Ethernet1/1/1, src/dst 10.105.0.8/255.255.255.255, inst_id 0
tdp: Incoming conn 172.27.32.28:711 <-> 210.9.0.9:11019
tdp: Found adj 0x60DF6E50 for 210.9.0.9 (Hello xport addr opt)
tdp: New temporary adj 0x61033D38 from 210.9.0.9
tdp: Real adj 0x60DF6E50 bound to 210.9.0.9:0, replacing temp adj 0x61033D38
tdp: Adj 0x61033D38; state set to closed
tdp: Rcvd hello; Ethernet1/1/1, from 10.105.0.9 (210.9.0.9:0), intf_id 0, opt 0x4
tdp: Rcvd hello; ATM3/0.1, from 200.26.0.4 (202.0.0.1:1), intf_id 1, opt 0x4, tcatm
tdp: Send hello; ATM3/0.1, src/dst 99.101.0.8/255.255.255.255, inst_id 1, tcatm
tdp: Rcvd hello; Ethernet1/1/1, from 10.105.0.9 (210.9.0.9:0), intf_id 0, opt 0x4
tdp: Send hello; Ethernet1/1/1, src/dst 10.105.0.8/255.255.255.255, inst_id 0
tdp: Rcvd hello; ATM3/0.1, from 200.26.0.4 (202.0.0.1:1), intf_id 1, opt 0x4, tcatm
```

Table 198 describes the significant fields shown in the display.

Table 198 debug tag-switching tdp transport events Field Descriptions

Field	Description
tdp:	Identifies the source of the message as TDP.
adj 0x*nnnnnnnn*	Identifies the data structure used to represent the peer device at the transport level. Useful for correlating debug output.
a.b.c.d (*p.q.r.s:n*)	Network address and TDP identifier of the peer device.
intf_id	Interface identifier (nonzero for TC-ATM interfaces, 0 otherwise).
opt 0x*n*	Bits that describe options in the TDP discovery hello packet: • 0x1—Directed hello option • 0x2—Send directed hello option • 0x4—Transport address option

Related Commands

Command	Description
debug tag-switching tdp transport connections	Prints information about the TCP connections used to support TDP sessions.

debug tag-switching tdp transport timers

To print information about events that restart the "hold" timers that are part of the TDP discovery mechanism, use the **debug tag-switching tdp transport timers** privileged EXEC command. The **no** form of this command disables debugging output.

debug tag-switching tdp transport timers

no debug tag-switching tdp transport timers

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

TDP sessions are supported by data structures and state machines at three levels:

- Transport—The transport level establishes and maintains TCP connections used to support TDP sessions.
- Protocol—The protocol level implements the TDP session setup protocol. The construction and parsing of TDP PDUs and PIEs occur at this level.
- Tag distribution—The tag distribution level uses TDP sessions to exchange tags with TDP peer devices.

The **debug tag-switching tdp transport** command provides visibility of activity at the transport level, the **debug tag-switching tdp session** command at the protocol level, and the **debug tag-switching tdp peer** command at the tag distribution level.

Examples

The following is sample output from the **debug tag-switching tdp transport timers** command:

```
Router# debug tag-switching tdp transport timers

tdp: Start holding timer; adj 0x60D5BC10, 200.26.0.4
tdp: Start holding timer; adj 0x60EA9360, 10.105.0.9
tdp: Start holding timer; adj 0x60D5BC10, 200.26.0.4
tdp: Start holding timer; adj 0x60EA9360, 10.105.0.9
tdp: Start holding timer; adj 0x60D5BC10, 200.26.0.4
tdp: Start holding timer; adj 0x60EA9360, 10.105.0.9
```

Table 199 describes the significant fields shown in the display.

Table 199 debug tag-switching tdp transport timers Field Descriptions

Field	Description
tdp	Identifies the source of the message as TDP.
adj 0x*nnnnnnnn*	Identifies the data structure used to represent the peer device at the transport level.
a.b.c.d	Network address of the peer device.

Related Commands

Command	Description
debug tag-switching tdp transport events	Prints information about the events related to the TDP peer discovery mechanism, which is used to determine the devices with which to establish TDP sessions.

debug tag-switching tfib cef

The **debug tag-switching tfib cef** command is replaced by the **debug mpls lfib cef** command. See the **debug mpls lfib cef** command for more information.

debug tag-switching tfib enc

The **debug tag-switching tfib enc** command is replaced by the **debug mpls lfib enc** command. See the **debug mpls lfib enc** command for more information.

debug tag-switching tfib state

The **debug tag-switching tfib state** command is replaced by the **debug mpls lfib state** command. See the **debug mpls lfib state** command for more information.

debug tag-switching tfib struct

The **debug tag-switching tfib struct** command is replaced by the **debug mpls lfib struct** command. See the **debug mpls lfib struct** command for more information.

debug tag-switching tfib tsp

The **debug tag-switching tfib tsp** command is replaced by the **debug mpls lfib lsp** command. See the **debug mpls lfib lsp** command for more information.

debug tag-switching tsp-tunnels events

The **debug tag-switching tsp-tunnels events** command is replaced by the **debug mpls traffic-eng tunnels events** command. See the **debug mpls traffic-eng tunnels events** command for more information.

debug tag-switching tsp-tunnels signalling

The **debug tag-switching tsp-tunnels signalling** command is replaced by the **debug mpls traffic-eng tunnels signalling** command. See the **debug mpls traffic-eng tunnels signalling** command for more information.

debug tag-switching tsp-tunnels tagging

The **debug tag-switching tsp-tunnels tagging** command is replaced by the **debug mpls traffic-eng tunnels labels** command. See the **debug mpls traffic-eng tunnels labels** command for more information.

debug tag-switching xtagatm cross-connect

To display requests and responses for establishing and removing cross-connects on the controlled ATM switch, use the **debug tag-switching xtagatm cross-connect** command. The **no** form of this command disables debugging output.

debug tag-switching xtagatm cross-connect

no debug tag-switching xtagatm cross-connect

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

Use the **debug tag-switching xtagatm cross-connect** command to monitor requests to establish or remove cross-connects from XTagATM interfaces to the VSI master, and the VSI master's responses to these requests.

Note Use this command with care, because it generates output for each cross-connect operation performed by the LSC. In a network configuration with a large number of label virtual circuits (LVCs), the volume of output generated may interfere with system timing and the proper operation of other router functions. Use this command only in situations in which the LVC setup or teardown rate is low.

Examples

The following is sample output from the **debug tag-switching xtagatm cross-connect** command:

```
Router# debug tag-switching xtagatm cross-connect

XTagATM: cross-conn request; SETUP, userdata 0x17, userbits 0x1, prec 7
    0xC0100 (Ctl-If) 1/32 <-> 0xC0200 (XTagATM0) 0/32
XTagATM: cross-conn response; DOWN, userdata 0x60CDCB5C, userbits 0x2, result
OK
    0xC0200 1/37 --> 0xC0300 1/37
```

Table 200 describes the significant fields shown in the sample command output shown above.

Table 200 debug tag-switching xtagatm cross-connect Field Descriptions

Field	Description
XTagATM	Identifies the source of the debug message as an XTagATM interface.
cross-conn	Indicates that the debug message pertains to a cross-connect setup or teardown operation.
request	A request from an XTagATM interface to the VSI master to set up or tear down a cross-connect.
response	Response from the VSI master to an XTagATM interface that a cross-connect was set up or removed.
SETUP	A request for the setup of a cross-connect.
TEARDOWN	A request for the teardown of a cross-connect.
UP	The cross-connect is established.
DOWN	The cross-connect is not established.
userdata, userbits	Values passed with the request that are returned in the corresponding fields shown in the matching response.
prec	The precedence for the cross-connect.
result	Indicates the status of the completed request.
0xC0100 (Ctl-If) 1/32	Indicates the following: that one endpoint of the cross-connect is on the interface whose logical interface number is 0xC0100; that this interface is the VSI control interface; that the VPI value at this endpoint is 1; and that the VCI value at this end of the cross-connect is 32.
<->	Indicates that this is a bidirectional cross-connect.
0xC0200 (XTagATM0) 0/32	Indicates the following: that the other endpoint of the cross-connect is on the interface whose logical interface number is 0xC0200; that this interface is associated with XTagATM interface 0; that the VPI value at this endpoint is 0; and that the VCI value at this end of the cross-connect is 32.
->	Indicates that this response pertains to a unidirectional cross-connect.

Related Commands

Command	Description
show xtagatm cross-connect	Displays information about remotely connected ATM switches.

debug tag-switching xtagatm errors

To display information about error and abnormal conditions that occur on XTagATM interfaces, use the **debug tag-switching xtagatm errors** command. The **no** form of this command disables debugging output.

debug tag-switching xtagatm errors

no debug tag-switching xtagatm errors

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

Use the **debug tag-switching xtagatm errors** command to display information about abnormal conditions and events that occur on XTagATM interfaces.

Examples

The following is sample output from the **debug tag-switching xtagatm errors** command:

```
Router# debug tag-switching xtagatm errors

XTagATM VC: XTagATM0 1707 2/352 (ATM1/0 1769 3/915): Cross-connect setup
failed NO_RESOURCES
```

This message indicates that an attempt to set up a cross-connect for a terminating VC on XTagATM interface 0 failed, and that the reason for the failure was a lack of resources on the controlled ATM switch.

debug tag-switching xtagatm events

To display information about major events that occur on XTagATM interfaces, not including events for specific XTagATM VCs and switch cross-connects, use the following **debug tag-switching xtagatm events** command. The **no** form of this command disables debugging output.

debug tag-switching xtagatm events

no debug tag-switching xtagatm events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

Use the **debug tag-switching xtagatm events** command to monitor major events that occur on XTagATM interfaces. This command monitors only events that pertain to XTagATM interfaces as a whole and does not include any events that pertain to individual XTagATM VCs or individual switch cross-connects. The specific events monitored when the **debug tag-switching xtagatm events** command is in effect include the following:

- Receipt of asynchronous notifications sent by the VSI master through the external ATM API (ExATM API) to an XTagATM interface.
- Resizing of the table that is used to store switch cross-connect information. This table is resized automatically as the number of cross-connects increases.
- Marking of XTagATM VCs as stale when an XTagATM interface shuts down, thereby ensuring that the stale interfaces are refreshed before new XTagATM VCs can be created on the interface.

Examples

The following is sample output from the **debug tag-switching xtagatm events** command:

```
Router# debug tag-switching xtagatm events

XTagATM: desired cross-connect table size set to 256
XTagATM: ExATM API intf event Up, port 0xA0100 (None)
XTagATM: ExATM API intf event Down, port 0xA0100 (None)
XTagATM: marking all VCs stale on XTagATM0
```

Table 201 describes the significant fields shown in the sample command output shown above.

Table 201 debug tag-switching xtagatm events Field Descriptions

Field	Description
XTagATM	Identifies the source of the debug message as an XTagATM interface.
desired cross-connect table size set to 256	Indicates that the table of cross-connect information has been set to hold 256 entries. A single cross-connect table is shared among all XTagATM interfaces. The cross-connect table is automatically resized as the number of cross-connects increases.
ExATM API	Indicates that the information in the debug output pertains to an asynchronous notification sent by the VSI master to the XTagATM driver.
event Up/Down	Indicates the specific event that was sent by the VSI master to the XTagATM driver.
port 0xA0100 (None)	Indicates that the event pertains to the VSI interface whose logical interface number is 0xA0100, and that this logical interface is not bound (through the **extended-port** interface configuration command) to any XTagATM interface.
marking all VCs stale on XTagATM0	Indicates that all existing XTagATM VCs on interface XTagATM0 are marked as stale, and that XTagATM0 remains down until all of these VCs are refreshed.

debug tag-switching xtagatm vc

To display information about events that affect individual XTagATM terminating VCs, use the **debug tag-switching xtagatm vc** command. The **no** form of this command disables debugging output.

debug tag-switching xtagatm vc

no debug tag-switching xtagatm vc

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

Use the **debug tag-switching xtagatm vc** command to display detailed information about all events that affect individual XTagATM terminating VCs.

Note Use this command with care, because it results in extensive output when many XTagATM VCs are set up or torn down. This output can interfere with system timing and normal operation of other router functions. Use the **debug tag-switching xtagatm vc** command only when a few XTagATM VCs are created or removed.

Examples

The following is sample output from the **debug tag-switching xtagatm vc** command:

```
Router# debug tag-switching xtagatm vc

XTagATM VC: XTagATM1 18 0/32 (ATM1/0 0 0/0): Setup, Down --> UpPend
XTagATM VC: XTagATM1 18 0/32 (ATM1/0 88 1/32): Complete, UpPend --> Up
XTagATM VC: XTagATM1 19 1/33 (ATM1/0 0 0/0): Setup, Down --> UpPend
XTagATM VC: XTagATM0 43 0/32 (ATM1/0 67 1/84): Teardown, Up --> DownPend
```

Table 202 describes the significant fields shown in the display.

Table 202 debug tag-switching stagatm vc Field Descriptions

Field	Description
XTagATM VC	Identifies the source of the debug message as the XTagATM interface terminating VC facility.
XTagATM <ifnum>	Identifies the particular XTagATM interface number for the terminating VC.
vcd vpi/vci	Indicates the VCD and VPI/VCI values for the terminating VC.
(ctl-if vcd vpi/vci)	Indicates the control interface, the VCD, and the VPI and VCI values for the private VC corresponding to the XTagATM VC on the control interface.
Setup, Complete, Teardown	Indicates the name of the particular event that has occurred for the indicated VC.
oldstate -> newstate	Indicates the state of the terminating VC before and after the processing of the indicated event.

debug tarp events

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

debug tarp events

no debug tarp events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

```
Router# debug tarp events

Router# debug tarp packets

Router# tarp resolve artemis

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

 NET corresponding to TID artemis is 49.0001.1111.1111.1111.00

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

Table 203 describes the significant fields in this display.

Table 203 debug tarp events Field Descriptions—tarp resolve Command

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

Related Commands

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

debug tarp packets

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

debug tarp packets

no debug tarp packets

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

```
Router# debug tarp packets

Router# debug tarp events

Router# tarp query 49.0001.3333.3333.3333.00

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

 TID corresponding to NET 49.0001.3333.3333.3333.00 is cerdiwen

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

Table 204 describes the significant fields shown in the display.

Table 204 debug tarp packets Field Descriptions—tarp query Command

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

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

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

Related Commands

Command	Modification
debug tarp events	Displays information on TARP activity.

debug tccs signaling

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

debug tccs signaling

no debug tccs signaling

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command Modes

EXEC

Command History

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

Usage Guidelines

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

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

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

Examples

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

```
Router# debug tccs signaling

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

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

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

debug tdm

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

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

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

Syntax Description

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

Usage Guidelines

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

Examples

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

```
Router# debug tdm api

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

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

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

Dynamic Backplane Timeslot Pool:

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


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


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


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

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

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


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

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

debug telco-return msg

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

debug cable telco-return msg

no debug cable telco-return msg

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for telco-return messages is not enabled.

Command History

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

Examples

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

Related Commands

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

debug telnet

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

debug telnet

no debug telnet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
8.1	This command was introduced.

Examples

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

```
Router# debug telnet

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

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

Table 205 describes the significant fields shown in the display.

Table 205 debug telnet Field Descriptions

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

Related Commands

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

debug text-to-fax

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

debug text-to-fax

[no] debug text-to-fax

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

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

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

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

debug tftp

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

debug tftp

no debug tftp

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug tftp

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

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

Table 206 debug tftp Field Descriptions

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

debug tgrm

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

debug tgrm

no debug tgrm

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

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

Examples

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

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

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

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

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

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

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

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

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

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

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

Related Commands

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

debug tiff reader

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

debug tiff reader

[no] debug tiff-reader

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Related Commands

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

debug tiff writer

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

debug tiff writer

[no] debug tiff-writer

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

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

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

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

Related Commands

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

debug token ring

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

debug token ring

no debug token ring

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

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

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

Examples

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

```
Router# debug token ring

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

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

Table 207 debug token ring Field Descriptions

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

Table 207 debug token ring Field Descriptions (continued)

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

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

Table 208 debug token ring Field Descriptions

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

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

Table 209 debug token ring Field Descriptions

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

debug tsp

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

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

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

Syntax Description

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

Defaults

Disabled

Command History

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

Examples

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

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

Related Commands

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

debug txconn all

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

debug txconn all

no debug txconn all

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

```
Router# debug txconn all

All possible TXConn debugging has been turned on
```

Related Commands

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

debug txconn appc

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

debug txconn appc

no debug txconn appc

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

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

Related Commands

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

debug txconn config

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

debug txconn config

no debug txconn config

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

```
Router# debug txconn config

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

Related Commands

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

debug txconn data

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

debug txconn data

no debug txconn data

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

```
Router# debug txconn data

TXConn DATA debugging is on

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

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

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

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

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

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

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

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

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

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

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

Related Commands

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

debug txconn event

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

debug txconn event

no debug txconn event

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

```
Router# debug txconn event

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

Related Commands

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

debug txconn tcp

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

debug txconn tcp

no debug txconn tcp

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

```
Router# debug txconn tcp

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

Related Commands

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

debug txconn timer

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

debug txconn timer

no debug txconn timer

Syntax Description

This command has no arguments or keywords.

Defaults

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

Command History

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

Examples

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

```
Router# debug txconn timer

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

Related Commands

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

debug udptn

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

debug udptn

no debug udptn

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

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

Examples

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

```
terrapin# debug udptn

terrapin# udptn 172.16.1.1
Trying 172.16.1.1 ... Open

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

Related Commands

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

debug v120 event

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

debug v120 event

no debug v120 event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

Examples

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

```
Router# debug v120 event

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

Related Commands

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

debug v120 packet

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

debug v120 packet

no debug v120 packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

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

Examples

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

```
Router# debug v120 packet

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

Table 210 describes the significant fields in the display.

Table 210 debug v.120 packet Field Descriptions

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

Related Commands

Command	Description
debug tarp events	Displays information on TARP activity.

debug vg-anylan

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

debug vg-anylan

no debug vg-anylan

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command could create a substantial amount of command output.

Examples

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

```
Router# debug vg-anylan

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

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

Table 211 debug vg-anylan Message Descriptions

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

Table 211 debug vg-anylan Message Descriptions (continued)

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

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

debug video vicm

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

debug video vicm

no debug video vicm

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for the ViCM is not enabled.

Command History

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

Examples

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

```
Router# debug video vicm

Video ViCM FSM debugging is on

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

Router# SVC HANDLE in rcvd:0x80001B:

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

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

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

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

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

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

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

debug vines arp

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

debug vines arp

no debug vines arp

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug vines arp

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

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

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

Table 212 describes the significant fields shown in the display.

Table 212 debug vines arp Field Descriptions

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

debug vines echo

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

debug vines echo

no debug vines echo

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

These echo packets do not include network-layer addresses.

Examples

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

```
Router# debug vines echo

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

Table 213 describes the significant fields shown in the display.

Table 213 debug vines echo Field Descriptions

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

debug vines ipc

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

debug vines ipc

no debug vines ipc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

```
Router# debug vines ipc

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

Table 214 describes the significant fields shown in the display.

Table 214 debug vines ipc Field Descriptions

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

Table 214 debug vines ipc Field Descriptions (continued)

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

debug vines netrpc

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

debug vines netrpc

no debug vines netrpc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

```
Router# debug vines netrpc

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

Table 215 describes the significant fields shown in the display.

Table 215 debug vines netrpc Field Descriptions

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

debug vines packet

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

debug vines packet

no debug vines packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug vines packet

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

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

Table 216 debug vines packet Field Descriptions

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

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

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

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

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

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

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

debug vines routing

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

debug vines routing [**verbose**]

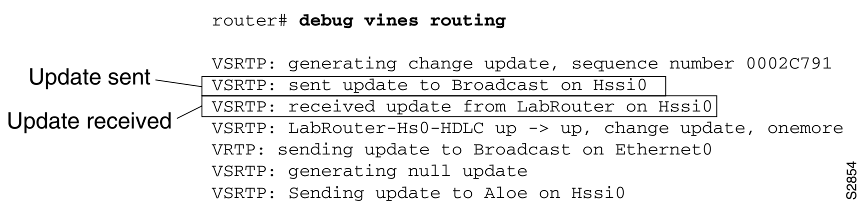
no debug vines routing [**verbose**]

Syntax Description

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

Examples

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



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

```
Router# debug vines routing verbose

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
network LabRouter from the server
```

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

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

debug vines service

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

debug vines service

no debug vines service

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

Examples

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

```
router# debug vines service
```

Sent/ Response pair

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

S2565

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

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

Table 217 debug vines service Field Descriptions

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

Table 217 debug vines service Field Descriptions (continued)

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

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

Table 218 debug vines service Field Descriptions

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

debug vines state

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

debug vines state

no debug vines state

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

debug vines table

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

debug vines table

no debug vines table

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

```
Router# debug vines table

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

Table 219 describes the significant fields in the display.

Table 219 debug vines table Field Descriptions

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

debug vlan packet

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

debug vlan packet

no debug vlan packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

```
Router# debug vlan packet

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

debug voice all

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

debug voice all [*slot/port*]

no debug voice all [*slot/port*]

Syntax Description

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

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

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

Related Commands

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

debug voice cp

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

debug voice cp [*slot/port*]

no debug voice cp [*slot/port*]

Syntax Description

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

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

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

```
Router# debug voice cp 1/1

Voice Call Processing State Machine debugging is on

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

Related Commands

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

debug voice eecm

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

debug voice eecm [*slot/port*]

no debug voice eecm [*slot/port*]

Syntax Description

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

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

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

```
Router# debug voice eecm

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

Related Commands

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

debug voice protocol

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

debug voice protocol [*slot/port*]

no debug voice protocol [*slot/port*]

Syntax Description

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

Usage Guidelines

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

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

Command History

This command is valid on the Cisco MC3810 device only.

Examples

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

```
Router# debug voice protocol

Voice Line protocol State machine debugging is on

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

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

Related Commands

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

debug voice signaling

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

debug voice signaling [*slot*/*port*]

no debug voice signaling [*slot*/*port*]

Syntax Description

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

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

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

```
Router# debug voice signaling

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

Related Commands

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

debug voice tdsm

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

debug voice tdsm [*slot*/*port*]

no debug voice tdsm [*slot*/*port*]

Syntax Description

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

Usage Guidelines

This command is valid on the Cisco MC3810 device only.

Examples

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

```
Router# debug voice tdsm

Voice tandem switch debugging is on

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

Related Commands

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

debug voice vofr

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

debug voice vofr

no debug voice vofr

Syntax Description

This command has no arguments or keywords.

Command History

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

Usage Guidelines

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

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

Examples

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

```
Router# debug voice vofr

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

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

```
Router# debug voice vofr

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

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

Related Commands

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

debug voip aaa

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

debug voip aaa

no debug voip aaa

Syntax Description

This command has no arguments or keywords.

Command History

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

debug voip ccapi

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

debug voip ccapi

no debug voip ccapi

Syntax Description

This command has no arguments or keywords.

Command History

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

Examples

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

```
Router# show debug

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

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

debug voip ccapi error

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

debug voip ccapi error

no debug voip ccapi error

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

debug voip ccapi inout

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

debug voip ccapi inout

no debug voip ccapi inout

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

Examples

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

```
Router# debug voip ccapi inout

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

debug voip ivr

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

debug voip ivr

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

Syntax Description

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

Defaults

Debug is not enabled.

Command History

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

Examples

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

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

```
Router# debug voip ivr applib

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

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

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

```
Router# debug voip ivr callsetup

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

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

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

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

```
Router# debug voip ivr digitcollect

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

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

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

```
Router# deb voip ivr script

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

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

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

```
Router# debug voip ivr tclcommands

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

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

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

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

Related Commands

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

debug voip ivr settlement

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

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

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

Syntax Description

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

Defaults

Not enabled

Usage Guidelines

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

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

Command History

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

Examples

Example On the Originating Gateway

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

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

Example On the Terminating Gateway

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

debug voip rawmsg

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

debug voip rawmsg [*detail*]

no debug voip rawmsg [*detail*]

Syntax Description

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

Defaults

Disabled.

Command History

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

Examples

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

```
as5300# debug voip rawmsg

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

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

```
as5300# debug voip rawmsg detail

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

Related Commands

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

debug voip settlement all

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

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

Syntax Description

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

Defaults

Not enabled

Command History

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

Usage Guidelines

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

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

debug voip settlement enter

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

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

Defaults

Not enabled

Command History

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

Examples

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

debug voip settlement error

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

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

Defaults

Not enabled

Command History

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

Examples

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

Usage Guidelines

See "Error Code Definitions" section on page 1019.

Error Code Definitions

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

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

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

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

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

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

12200:OSP security - CA certificate index overflow.

13010:OSP error message - failed to allocate memory.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

22510:OSP audit error - failed to allocate memory.

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

debug voip settlement exit

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

debug voip settlement exit

no debug voip settlement exit

Defaults

Not enabled

Command History

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

Examples

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

debug voip settlement misc

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

debug voip settlement misc

no debug voip settlement misc

Defaults

Not enabled

Command History

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

Examples

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

debug voip settlement network

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

debug voip settlement network

no debug voip settlement network

Defaults

Not enabled

Command History

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

Usage Guidelines

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

Examples

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

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

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

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

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

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

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

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

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

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

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

This is your response signature

--bar--
```

debug voip settlement security

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

debug voip settlement security

no debug voip settlement security

Defaults

Not enabled

Command History

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

Examples

Not available due to security issues.

debug voip settlement transaction

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

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

Defaults

Not enabled

Command History

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

Examples

Sample output from the originating gateway:

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

Sample output from the terminating gateway:

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

debug vpdn

To troubleshoot Layer 2 Forwarding (L2F) or Layer 2 Tunnel Protocol (L2TP) virtual private dialup network (VPDN) tunneling events and infrastructure, use the **debug vpdn** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpdn {**error** | **event** [**disconnect**] | **l2tp-sequencing** | **l2x-data** | **l2x-errors** | **l2x-events** | **l2x-packets** | **packet** [**errors**]}

no debug vpdn {**error** | **event** [**disconnect**] | **l2tp-sequencing** | **l2x-data** | **l2x-errors** | **l2x-events** | **l2x-packets** | **packet** [**errors**]}

Syntax Description

error	Displays VPDN errors.
event	Displays VPDN events.
disconnect	(Optional) Displays VPDN disconnect events.
l2tp-sequencing	Displays significant events related to L2TP sequence numbers such as mismatches, resend queue flushes, and drops.
l2x-data	Displays errors that occur in data packets.
l2x-errors	Displays errors that occur in protocol-specific conditions.
l2x-events	Displays events resulting from protocol-specific conditions.
l2x-packets	Displays detailed information about control packets in protocol-specific conditions.
packet	Displays information about VPDN packets.
errors	(Optional) Displays errors that occur in packet processing.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2	This command was introduced.
12.0(5)T	Support was added for L2TP debugging messages. The **l2tp-sequencing** and **errors** keywords were added. The **l2f-errors**, **l2f-events**, and **l2f-packets** keywords were changed to **l2x-errors**, **l2x-events**, and **l2x-packets**.

Usage Guidelines

Note that the **debug vpdn packet** and **debug vpdn packet detail** commands generate several debug operations per packet. Depending on the L2TP traffic pattern, these commands may cause the CPU load to increase to a high level that impacts performance.

Examples

This section contains the following examples:

- Debugging VPDN Events on a NAS—Normal L2F Operations
- Debugging VPDN Events on the Tunnel Server—Normal L2F Operations

- Debugging VPDN Events on the NAS—Normal L2TP Operations
- Debugging VPDN Events on the Tunnel Server—Normal L2TP Operations
- Debugging Protocol-Specific Events on the NAS—Normal L2F Operations
- Debugging Protocol-Specific Events on the Tunnel Server—Normal L2F Operations
- Debugging Errors on the NAS—L2F Error Conditions
- Debugging L2F Control Packets for Complete Information

Debugging VPDN Events on a NAS—Normal L2F Operations

The network access server (NAS) has the following VPDN configuration:

```
vpdn-group 1
 request-dialin
  protocol l2f
  domain cisco.com
 initiate-to ip 172.17.33.125
username nas1 password nas1
```

The following is sample output from the **debug vpdn event** command on a NAS when an L2F tunnel is brought up and Challenge Handshake Authentication Protocol (CHAP) authentication of the tunnel succeeds:

```
Router# debug vpdn event

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:26:05.537: looking for tunnel -- cisco.com --
*Mar 2 00:26:05.545: Async6 VPN Forwarding...
*Mar 2 00:26:05.545: Async6 VPN Bind interface direction=1
*Mar 2 00:26:05.553: Async6 VPN vpn_forward_user user6@cisco.com is forwarded
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:26:06.289: L2F: Chap authentication succeeded for nas1.
```

The following is sample output from the **debug vpdn event** command on a NAS when the L2F tunnel is brought down normally:

```
Router# debug vpdn event

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:27:18.865: Async6 VPN cleanup
*Mar 2 00:27:18.869: Async6 VPN reset
*Mar 2 00:27:18.873: Async6 VPN Unbind interface
%LINK-3-UPDOWN: Interface Async6, changed state to down
```

Table 220 describes the significant fields shown in the two previous displays. The output describes normal operations when an L2F tunnel is brought up or down on a NAS.

Table 220 debug vpdn event Field Descriptions for the NAS

Field	Description
Asynchronous interface coming up	
%LINK-3-UPDOWN: Interface Async6, changed state to up	Asynchronous interface 6 came up.
looking for tunnel -- cisco.com -- Async6 VPN Forwarding...	Domain name is identified.

Table 220 debug vpdn event Field Descriptions for the NAS (continued)

Field	Description
Async6 VPN Bind interface direction=1	Tunnel is bound to the interface. These are the direction values: • 1—From the NAS to the tunnel server • 2—From the tunnel server to the NAS
Async6 VPN vpn_forward_user user6@cisco.com is forwarded	Tunnel for the specified user and domain name is forwarded.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol is up.
L2F: Chap authentication succeeded for nas1.	Tunnel was authenticated with the tunnel password nas1.
Virtual access interface coming down	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down	Normal operation when the virtual access interface is taken down.
Async6 VPN cleanup Async6 VPN reset Async6 VPN Unbind interface	Normal cleanup operations performed when the line or virtual access interface goes down.

Debugging VPDN Events on the Tunnel Server—Normal L2F Operations

The tunnel server has the following VPDN configuration, which uses nas1 as the tunnel name and the tunnel authentication name. The tunnel authentication name might be entered in a users file on an authentication, authorization, and accounting (AAA) server and used to define authentication requirements for the tunnel.

```
vpdn-group 1
 accept-dialin
  protocol l2f
  virtual-template 1
 terminate-from hostname nas1
```

The following is sample output from the **debug vpdn event** command on the tunnel server when an L2F tunnel is brought up successfully:

```
Router# debug vpdn event

L2F: Chap authentication succeeded for nas1.
Virtual-Access3 VPN Virtual interface created for user6@cisco.com
Virtual-Access3 VPN Set to Async interface
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up
Virtual-Access3 VPN Bind interface direction=2
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up
```

The following is sample output from the **debug vpdn event** command on a tunnel server when an L2F tunnel is brought down normally:

```
Router# debug vpdn event

%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down
Virtual-Access3 VPN cleanup
```

```
Virtual-Access3 VPN reset
Virtual-Access3 VPN Unbind interface
Virtual-Access3 VPN reset
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down
```

Table 221 describes the fields shown in two previous outputs. The output describes normal operations when an L2F tunnel is brought up or down on a tunnel server.

Table 221 debug vpdn event Field Descriptions for the Tunnel Server

Field	Description
Tunnel coming up	
L2F: Chap authentication succeeded for nas1.	PPP CHAP authentication status for the tunnel named nas1.
Virtual-Access3 VPN Virtual interface created for user6@cisco.com	Virtual access interface was set up on the tunnel server for the user user6@cisco.com.
Virtual-Access3 VPN Set to Async interface	Virtual access interface 3 was set to asynchronous for character-by-character transmission.
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0	Virtual template 1 was applied to virtual access interface 3.
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up	Link status is set to up.
Virtual-Access3 VPN Bind interface direction=2	Tunnel is bound to the interface. These are the direction values: • 1—From the NAS to the tunnel server • 2—From the tunnel server to the NAS
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK	PPP link control protocol (LCP) configuration settings (negotiated between the remote client and the NAS) were copied to the tunnel server and acknowledged.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up	Line protocol is up; the line can be used.
Tunnel coming down	
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down	Virtual access interface is coming down.
Virtual-Access3 VPN cleanup Virtual-Access3 VPN reset Virtual-Access3 VPN Unbind interface Virtual-Access3 VPN reset	Router is performing normal cleanup operations when a virtual access interface used for an L2F tunnel comes down.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down	Line protocol is down for virtual access interface 3; the line cannot be used.

Debugging VPDN Events on the NAS—Normal L2TP Operations

The following is sample output from the **debug vpdn event** command on the NAS when an L2TP tunnel is brought up successfully:

```
Router# debug vpdn event

20:19:17: L2TP: I SCCRQ from ts1 tnl 8
20:19:17: L2X: Never heard of ts1
20:19:17: Tnl 7 L2TP: New tunnel created for remote ts1, address 172.21.9.4
20:19:17: Tnl 7 L2TP: Got a challenge in SCCRQ, ts1
20:19:17: Tnl 7 L2TP: Tunnel state change from idle to wait-ctl-reply
20:19:17: Tnl 7 L2TP: Got a Challenge Response in SCCCN from ts1
20:19:17: Tnl 7 L2TP: Tunnel Authentication success
20:19:17: Tnl 7 L2TP: Tunnel state change from wait-ctl-reply to established
20:19:17: Tnl 7 L2TP: SM State established
20:19:17: Tnl/Cl 7/1 L2TP: Session FS enabled
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from idle to wait-for-tunnel
20:19:17: Tnl/Cl 7/1 L2TP: New session created
20:19:17: Tnl/Cl 7/1 L2TP: O ICRP to ts1 8/1
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from wait-for-tunnel to wait-connect
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from wait-connect to established
20:19:17: Vi1 VPDN: Virtual interface created for bum1@cisco.com
20:19:17: Vi1 VPDN: Set to Async interface
20:19:17: Vi1 VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
20:19:18: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
20:19:18: Vi1 VPDN: Bind interface direction=2
20:19:18: Vi1 VPDN: PPP LCP accepting rcv CONFACK
20:19:19: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to
up
```

Debugging VPDN Events on the Tunnel Server—Normal L2TP Operations

The following is sample output from the **debug vpdn event** command on the tunnel server when an L2TP tunnel is brought up successfully:

```
Router# debug vpdn event

20:47:33: %LINK-3-UPDOWN: Interface Async7, changed state to up
20:47:35: As7 VPDN: Looking for tunnel -- cisco.com --
20:47:35: As7 VPDN: Get tunnel info for cisco.com with NAS nas1, IP 172.21.9.13
20:47:35: As7 VPDN: Forward to address 172.21.9.13
20:47:35: As7 VPDN: Forwarding...
20:47:35: As7 VPDN: Bind interface direction=1
20:47:35: Tnl/Cl 8/1 L2TP: Session FS enabled
20:47:35: Tnl/Cl 8/1 L2TP: Session state change from idle to wait-for-tunnel
20:47:35: As7 8/1 L2TP: Create session
20:47:35: Tnl 8 L2TP: SM State idle
20:47:35: Tnl 8 L2TP: Tunnel state change from idle to wait-ctl-reply
20:47:35: Tnl 8 L2TP: SM State wait-ctl-reply
20:47:35: As7 VPDN: bum1@cisco.com is forwarded
20:47:35: Tnl 8 L2TP: Got a challenge from remote peer, nas1
20:47:35: Tnl 8 L2TP: Got a response from remote peer, nas1
20:47:35: Tnl 8 L2TP: Tunnel Authentication success
20:47:35: Tnl 8 L2TP: Tunnel state change from wait-ctl-reply to established
20:47:35: Tnl 8 L2TP: SM State established
20:47:35: As7 8/1 L2TP: Session state change from wait-for-tunnel to wait-reply
20:47:35: As7 8/1 L2TP: Session state change from wait-reply to established
20:47:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async7, changed state to up
```

Debugging Protocol-Specific Events on the NAS—Normal L2F Operations

The following is sample output from the **debug vpdn l2x-events** command on the NAS when an L2F tunnel is brought up successfully:

```
Router# debug vpdn l2x-events

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:41:17.365: L2F Open UDP socket to 172.21.9.26
*Mar 2 00:41:17.385: L2F_CONF received
*Mar 2 00:41:17.389: L2F Removing resend packet (type 1)
*Mar 2 00:41:17.477: L2F_OPEN received
*Mar 2 00:41:17.489: L2F Removing resend packet (type 2)
*Mar 2 00:41:17.493: L2F building nas2gw_mid0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:41:18.613: L2F_OPEN received
*Mar 2 00:41:18.625: L2F Got a MID management packet
*Mar 2 00:41:18.625: L2F Removing resend packet (type 2)
*Mar 2 00:41:18.629: L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6
```

The following is sample output from the **debug vpdn l2x-events** command on a NAS when an L2F tunnel is brought down normally:

```
Router# debug vpdn l2x-events

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:42:29.213: L2F_CLOSE received
*Mar 2 00:42:29.217: L2F Destroying mid
*Mar 2 00:42:29.217: L2F Removing resend packet (type 3)
*Mar 2 00:42:29.221: L2F Tunnel is going down!
*Mar 2 00:42:29.221: L2F Initiating tunnel shutdown.
*Mar 2 00:42:29.225: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F Got closing for tunnel
*Mar 2 00:42:29.233: L2F Removing resend packet
*Mar 2 00:42:29.233: L2F Closed tunnel structure
%LINK-3-UPDOWN: Interface Async6, changed state to down
*Mar 2 00:42:31.793: L2F Closed tunnel structure
*Mar 2 00:42:31.793: L2F Deleted inactive tunnel
```

Table 222 describes the fields shown in the displays.

Table 222 debug vpdn l2x-events Field Descriptions—NAS

Field	Descriptions
Tunnel coming up	
%LINK-3-UPDOWN: Interface Async6, changed state to up	Asynchronous interface came up normally.
L2F Open UDP socket to 172.21.9.26	L2F opened a User Datagram Protocol (UDP) socket to the tunnel server IP address.
L2F_CONF received	L2F_CONF signal was received. When sent from the tunnel server to the NAS, an L2F_CONF indicates the tunnel server's recognition of the tunnel creation request.

Table 222 debug vpdn l2x-events Field Descriptions—NAS (continued)

Field	Descriptions
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the tunnel server accepted the NAS configuration of an L2F tunnel.
L2F building nas2gw_mid0	L2F is building a tunnel between the NAS and the tunnel server, using the Multiplex ID (MID) MID0.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol came up. Indicates whether the software processes that handle the line protocol regard the interface as usable.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the tunnel server accepted the NAS configuration of an L2F tunnel.
L2F Got a MID management packet	MID management packets are used to communicate between the NAS and the tunnel server.
L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6	L2F synchronized the Client IDs on the NAS and the tunnel server, respectively. A multiplex ID is assigned to identify this connection in the tunnel.
Tunnel coming down	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down	Line protocol came down. Indicates whether the software processes that handle the line protocol regard the interface as usable.
%LINK-5-CHANGED: Interface Async6, changed state to reset	Interface was marked as reset.
L2F_CLOSE received	NAS received a request to close the tunnel.
L2F Destroying mid	Connection identified by the MID is being taken down.
L2F Tunnel is going down!	Advisory message about impending tunnel shutdown.
L2F Initiating tunnel shutdown.	Tunnel shutdown has started.
L2F_CLOSE received	NAS received a request to close the tunnel.
L2F Got closing for tunnel	NAS began tunnel closing operations.
%LINK-3-UPDOWN: Interface Async6, changed state to down	Asynchronous interface was taken down.
L2F Closed tunnel structure	NAS closed the tunnel.
L2F Deleted inactive tunnel	Now-inactivated tunnel was deleted.

Debugging Protocol-Specific Events on the Tunnel Server—Normal L2F Operations

The following is sample output from the **debug vpdn l2x-events** command on a tunnel server when an L2F tunnel is created:

```
Router# debug vpdn l2x-events

L2F_CONF received
L2F Creating new tunnel for nas1
L2F Got a tunnel named nas1, responding
L2F Open UDP socket to 172.21.9.25
L2F_OPEN received
L2F Removing resend packet (type 1)
L2F_OPEN received
L2F Got a MID management packet
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
```

The following is sample output from the **debug vpdn l2x-events** command on a tunnel server when the L2F tunnel is brought down normally:

```
Router# debug vpdn l2x-events

L2F_CLOSE received
L2F Destroying mid
L2F Removing resend packet (type 3)
L2F Tunnel is going down!
L2F Initiating tunnel shutdown.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
L2F_CLOSE received
L2F Got closing for tunnel
L2F Removing resend packet
L2F Removing resend packet
L2F Closed tunnel structure
L2F Closed tunnel structure
L2F Deleted inactive tunnel
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down
```

Table 223 describes the significant fields shown in the displays.

Table 223 debug vpdn l2x-events Field Descriptions—Tunnel Server

Field	Description
Tunnel coming up	
L2F_CONF received	L2F configuration is received from the NAS. When sent from a NAS to a tunnel server, the L2F_CONF is the initial packet in the conversation.
L2F Creating new tunnel for nas1	Tunnel named *nas1* is being created.
L2F Got a tunnel named nas1, responding	Tunnel server is responding.
L2F Open UDP socket to 172.21.9.25	Opening a socket to the NAS IP address.
L2F_OPEN received	L2F_OPEN management message was received, indicating the NAS is opening an L2F tunnel.
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. The two resend packet types have different meanings in different states of the tunnel.

Table 223 debug vpdn l2x-events Field Descriptions—Tunnel Server (continued)

Field	Description
L2F Got a MID management packet	L2F MID management packets are used to communicate between the NAS and the tunnel server.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up	Tunnel server is bringing up virtual access interface 1 for the L2F tunnel.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up	Line protocol is up. The line can be used.
Tunnel coming down	
L2F_CLOSE received	NAS or tunnel server received a request to close the tunnel.
L2F Destroying mid	Connection identified by the MID is being taken down.
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel.
L2F Tunnel is going down! L2F Initiating tunnel shutdown.	Router is performing normal operations when a tunnel is coming down.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down	The virtual access interface is coming down.
L2F_CLOSE received L2F Got closing for tunnel L2F Removing resend packet L2F Removing resend packet L2F Closed tunnel structure L2F Closed tunnel structure L2F Deleted inactive tunnel	Router is performing normal cleanup operations when the tunnel is being brought down.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down	Line protocol is down; virtual access interface 1 cannot be used.

Debugging Errors on the NAS—L2F Error Conditions

The following is sample output from the **debug vpdn errors** command on a NAS when the L2F tunnel is not set up:

```
Router# debug vpdn errors

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down
%LINK-5-CHANGED: Interface Async1, changed state to reset
%LINK-3-UPDOWN: Interface Async1, changed state to down
%LINK-3-UPDOWN: Interface Async1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up
VPDN tunnel management packet failed to authenticate
VPDN tunnel management packet failed to authenticate
```

Table 224 describes the significant fields shown in the display.

Table 224 debug vpdn error Field Descriptions for the NAS

Field	Description
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down	Line protocol on the asynchronous interface went down.
%LINK-5-CHANGED: Interface Async1, changed state to reset	Asynchronous interface 1 was reset.
%LINK-3-UPDOWN: Interface Async1, changed state to down %LINK-3-UPDOWN: Interface Async1, changed state to up	Link from asynchronous interface 1 link went down and then came back up.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up	Line protocol on the asynchronous interface came back up.
VPDN tunnel management packet failed to authenticate	Tunnel authentication failed. This is the most common VPDN error. **Note** Verify the password for the NAS and the tunnel server name. If you store the password on an AAA server, you can use the **debug aaa authentication** command.

The following is sample output from the **debug vpdn l2x-errors** command:

```
Router# debug vpdn l2x-errors

%LINK-3-UPDOWN: Interface Async1, changed state to up
L2F Out of sequence packet 0 (expecting 0)
L2F Tunnel authentication succeeded for cisco.com
 L2F Received a close request for a non-existent mid
 L2F Out of sequence packet 0 (expecting 0)
 L2F packet has bogus1 key 1020868 D248BA0F
L2F packet has bogus1 key 1020868 D248BA0F
```

Table 225 describes the significant fields shown in the display.

Table 225 debug vpdn l2x-errors Field Descriptions

Field	Description
%LINK-3-UPDOWN: Interface Async1, changed state to up	The line protocol on the asynchronous interface came up.
L2F Out of sequence packet 0 (expecting 0)	Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received.
L2F Tunnel authentication succeeded for cisco.com	Tunnel was established from the NAS to the tunnel server, cisco.com.
L2F Received a close request for a non-existent mid	Multiplex ID was not used previously; cannot close the tunnel.
L2F Out of sequence packet 0 (expecting 0)	Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received.

Table 225 debug vpdn l2x-errors Field Descriptions (continued)

Field	Description
L2F packet has bogus1 key 1020868 D248BA0F	Value based on the authentication response given to the peer during tunnel creation. This packet, in which the key does not match the expected value, must be discarded.
L2F packet has bogus1 key 1020868 D248BA0F	Another packet was received with an invalid key value. The packet must be discarded.

Debugging L2F Control Packets for Complete Information

The following is sample output from the **debug vpdn l2x-packets** command on a NAS. This example displays a trace for a **ping** command:

```
Router# debug vpdn l2x-packets

L2F SENDING (17): D0 1 1 10 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 16 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 10 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F SENDING (17): D0 1 1 11 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 17 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 11 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F header flags: 57345 version 57345 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key 1701976070
L2F-IN Output to Async1 (16): FF 3 C0 21 9 F 0 C 0 1D 41 AD FF 11 46 87
L2F-OUT (16): FF 3 C0 21 A F 0 C 0 1A C9 BD FF 11 46 87
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key -2120949344
L2F-OUT (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 3 1 0 0 1 8 0 62 B1
0 0 C A8 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key -2120949344
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key 1701976070
L2F-IN Output to Async1 (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 1 1 0
0 3 0 0 6A B1 0 0 C A8 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
```

Table 226 describes the significant fields shown in the display.

Table 226 debug vpdn l2x-packets Field Descriptions

Field	Description
L2F SENDING (17)	Number of bytes being sent. The first set of "SENDING"..."RECEIVED" lines displays L2F keepalive traffic. The second set displays L2F management data.
L2F header flags:	Version and flags, in decimal.
version 53249	Version.
protocol 1	Protocol for negotiation of the point-to-point link between the NAS and the tunnel server is always 1, indicating L2F management.

Table 226 debug vpdn l2x-packets Field Descriptions (continued)

Field	Description
sequence 16	Sequence numbers start at 0. Each subsequent packet is sent with the next increment of the sequence number. The sequence number is thus a free running counter represented modulo 256. There is a distinct sequence counter for each distinct MID value.
mid 0	Multiplex ID, which identifies a particular connection within the tunnel. Each new connection is assigned a MID currently unused within the tunnel.
cid 4	Client ID used to assist endpoints in demultiplexing tunnels.
length 17	Size in octets of the entire packet, including header, all fields pre-sent, and payload. Length does not reflect the addition of the checksum, if pre-sent.
offset 0	Number of bytes past the L2F header at which the payload data is expected to start. If it is 0, the first byte following the last byte of the L2F header is the first byte of payload data.
key 1701976070	Value based on the authentication response given to the peer during tunnel creation. During the life of a session, the key value serves to resist attacks based on spoofing. If a packet is received in which the key does not match the expected value, the packet must be silently discarded.
L2F RECEIVED (17)	Number of bytes received.
L2F-IN Otput to Async1 (16)	Payload datagram. The data came in to the VPDN code.
L2F-OUT (16):	Payload datagram sent out from the VPDN code to the tunnel.
L2F-OUT (101)	Ping payload datagram. The value 62 in this line is the ping packet size in hexadecimal (98 in decimal). The three lines that follow this line show ping packet data.

Related Commands

Command	Description
debug aaa authentication	Displays information on AAA/TACACS+ authentication.
debug acircuit	Displays events and failures related to attachment circuits.
debug pppoe	Display debugging information for PPPoE sessions.
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established sessions to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.

debug vpdn pppoe-data

To display data packets of PPPoE sessions, use the **debug vpdn pppoe-data** command in EXEC mode. To disable the debugging output, use the **no** form of this command.

debug vpdn pppoe-data

no debug vpdn pppoe-data

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

The **debug vpdn pppoe-data** command displays a large number of debug messages and should generally be used only on a debug chassis with a single active session.

Examples

The following is an example of output from the **debug vpdn pppoe-data** command:

```
6d20h:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
6d20h:PPPoE:OUT
 contiguous pak, size 19
     FF 03 C0 21 01 01 00 0F 03 05 C2 23 05 05 06 D3
     FF 2B DA
6d20h:PPPoE:IN
 particle pak, size 1240
     C0 21 01 01 00 0A 05 06 39 53 A5 17 00 00 00 00
     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 C0 21 02 01 00 0A 05 06 39 53 A5 17
6d20h:PPPoE:OUT
 contiguous pak, size 19
     FF 03 C0 21 01 02 00 0F 03 05 C2 23 05 05 06 D3
     FF 2B DA
6d20h:PPPoE:IN
 particle pak, size 1740
     C0 21 02 02 00 0F 03 05 C2 23 05 05 06 D3 FF 2B
     DA 00 80 C2 00 07 00 00 00 10 7B 01 2C D9 00 B0
     C2 EB 10 38 88 64 11 00
6d20h:PPPoE:OUT
 contiguous pak, size 30
     FF 03 C2 23 01 06 00 1A 10 99 1E 6E 8F 8C F2 C6
     EE 91 0A B0 01 CB 89 68 13 47 61 6E 67 61
```

```
6d20h:PPPoE:IN
 particle pak, size 3840
     C2 23 02 06 00 24 10 E6 84 FF 3A A4 49 19 CE D7
     AC D7 D5 96 CC 23 B3 41 6B 61 73 68 40 63 69 73
     63 6F 2E 63 6F 6D 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 8
     FF 03 C2 23 03 06 00 04
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 80 21 01 01 00 0A 03 06 65 65 00 66
6d20h:PPPoE:IN
 particle pak, size 1240
     80 21 01 01 00 0A 03 06 00 00 00 00 49 19 CE D7
     AC D7 D5 96 CC 23 B3 41 6B 61 73 68 40 63 69 73
     63 6F 2E 63 6F 6D 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 80 21 03 01 00 0A 03 06 65 65 00 67
6d20h:PPPoE:IN
 particle pak, size 1240
     80 21 02 01 00 0A 03 06 65 65 00 66 00 04 AA AA
     03 00 80 C2 00 07 00 00 00 10 7B 01 2C D9 00 B0
     C2 EB 10 38 88 64 11 00
6d20h:PPPoE:IN
 particle pak, size 1240
     80 21 01 02 00 0A 03 06 65 65 00 67 49 19 CE D7
     AC D7 D5 96 CC 23 B3 41 6B 61 73 68 40 63 69 73
     63 6F 2E 63 6F 6D 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 80 21 02 02 00 0A 03 06 65 65 00 67
6d20h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access1,
changed state to up
6d20h:PPPoE:OUT
 contiguous pak, size 16
     FF 03 C0 21 09 01 00 0C D3 FF 2B DA 4C 4D 49 A4
6d20h:PPPoE:IN
 particle pak, size 1440
     C0 21 0A 01 00 0C 39 53 A5 17 4C 4D 49 A4 AA AA
     03 00 80 C2 00 07 00 00 00 10 7B 01 2C D9 00 B0
     C2 EB 10 38 88 64 11 00
6d20h:PPPoE:IN
 particle pak, size 1440
     C0 21 09 01 00 0C 39 53 A5 17 00 00 00 00 00 00
     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00
```

Table 227 describes the fields shown in the displays.

Table 227 debug vpdn pppoe-data Field Descriptions

Field	Descriptions
6d20h:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up	Virtual access interface 1 came up.
6d20h:PPPoE:OUT	The host delivered a PPPoE session packet to the access concentrator.
6d20h:PPPoE:IN	The access concentrator received a PPPoE session packet.

Table 227 debug vpdn pppoe-data Field Descriptions (continued)

Field	Descriptions
6d20h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access1, changed state to up	Line protocol is up; the line can be used.
contiguous pak, size 19	Size 19 contiguous packet.
particle pak, size 1240	Size 1240 particle packet.

Related Commands

Command	Description
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established session to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpdn pppoe-error

To display PPPoE protocol errors that prevent a session from being established or errors that cause an established sessions to be closed, use the **debug vpdn pppoe-error** command in EXEC mode. To disable the debugging output, use the **no** form of this command.

debug vpdn pppoe-error

no debug vpdn pppoe-error

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following is a full list of error messages displayed by the **debug vpdn pppoe-error** command:

```
PPPOE:pppoe_acsys_err cannot grow packet
PPPoE:Cannot find PPPoE info
PPPoE:Bad MAC address:00b0c2eb1038
PPPOE:PADI has no service name tag
PPPoE:pppoe_handle_padi cannot add AC name/Cookie.
PPPoE:pppoe_handle_padi cannot grow packet
PPPoE:pppoe_handle_padi encap failed
PPPoE cannot create virtual access.
PPPoE cannot allocate session structure.
PPPoE cannot store session element in tunnel.
PPPoE cannot allocate tunnel structure.
PPPoE cannot store tunnel
PPPoE:VA221:No Session, Packet Discarded
PPPOE:Tried to shutdown a null session
PPPoE:Session already open, closing
PPPoE:Bad cookie:src_addr=00b0c2eb1038
PPPoE:Max session count on mac elem exceeded:mac=00b0c2eb1038
PPPoE:Max session count on vc exceeded:vc=3/77
PPPoE:Bad MAC address - dropping packet
PPPoE:Bad version or type - dropping packet
```

Table 228 describes the fields shown in the displays.

Table 228 debug vpdn pppoe-error Field Descriptions

Field	Descriptions
PPPOE:pppoe_acsys_err cannot grow packet	Asynchronous PPPoE packet initialization error.
PPPoE:Cannot find PPPoE info	The access concentrator sends a PADO to the host.
PPPoE:Bad MAC address:00b0c2eb1038	The host was unable to identify the Ethernet MAC address.

Table 228 debug vpdn pppoe-error Field Descriptions (continued)

Field	Descriptions
PPPOE:PADI has no service name tag	PADI requires a service name tag.
PPPoE:pppoe_handle_padi cannot add AC name/Cookie.	pppoe_handle_padi could not append AC name.
PPPoE:pppoe_handle_padi cannot grow packet	pppoe_handle_padi could not append packet.
PPPoE:pppoe_handle_padi encap failed	pppoe_handle_padi could not specify PPPoE on ATM encapsulation.
PPPoE cannot create virtual access.	PPPoE session unable to verify virtual access interface.
PPPoE cannot allocate session structure.	PPPoE session unable to allocate Stage Protocol.
PPPoE cannot store session element in tunnel.	PPPoE tunnel cannot allocate session element.
PPPoE cannot allocate tunnel structure.	PPPoE tunnel unable to allocate Stage Protocol.
PPPoE cannot store tunnel	PPPoE configuration settings unable to initialize a tunnel.
PPPoE:VA221:No Session, Packet Discarded	No sessions created. All packets dropped.
PPPOE:Tried to shutdown a null session	Null session shutdown.
PPPoE:Session already open, closing	PPPoE session already open.
PPPoE:Bad cookie:src_addr=00b0c2eb1038	PPPoE session unable to append new cookie.
PPPoE:Max session count on mac elem exceeded:mac=00b0c2eb1038	The maximum number of sessions exceeded the Ethernet MAC address.
PPPoE:Max session count on vc exceeded:vc=3/77	The maximum number of sessions exceeded the PVC connection.
PPPoE:Bad MAC address - dropping packet	The host was unable to identify the MAC address. Packet dropped.
PPPoE:Bad version or type - dropping packet	The host was unable to identify the encapsulation type.

Related Commands

Command	Description
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpdn pppoe-events

To display PPPoE protocol messages about events that are part of normal session establishment or shutdown, use the **debug vpdn pppoe-events** command in EXEC mode. To disable the debugging output, use the **no** form of this command.

debug vpdn pppoe-events

no debug vpdn pppoe-events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following is an example of output from the **debug vpdn pppoe-events** command:

```
1w5d:IN PADI from PPPoE tunnel
1w5d:OUT PADO from PPPoE tunnel
1w5d:IN PADR from PPPoE tunnel
1w5d:PPPoE:VPN session created.
1w5d:%LINK-3-UPDOWN:Interface Virtual-Access2, changed state to up

1w5d:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access2, changed state to up
```

Table 229 describes the significant fields shown in the display.

Table 229 debug vpdn pppoe-events Field Descriptions

Field	Descriptions
1w5d:IN PADI from PPPoE tunnel	The access concentrator receives a PADI packet from the PPPoE Tunnel.
1w5d:OUT PADO from PPPoE tunnel	The access concentrator sends a PADO to the host.
1w5d:IN PADR from PPPoE tunnel	The host sends a single PADR to the access concentrator that it has chosen.
1w5d:PPPoE:VPN session created.	The access concentrator receives the PADR packet and creates a VPN session.
1w5d:%LINK-3-UPDOWN:Interface Virtual-Access2, changed state to up	Virtual access interface 2 came up.
1w5d:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access2, changed state to up	Line protocol is up. The line can be used.

Related Commands

Command	Description
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established session to be closed.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpdn pppoe-packet

To display each PPPoE protocol packet exchanged, use the **debug vpdn pppoe-packet** command in EXEC mode.To disable the debugging output, use the **no** form of this command.

debug vpdn pppoe-packet

no debug vpdn pppoe-packet

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

The **debug vpdn pppoe-packet** command displays a large number of debug messages and should generally only be used on a debug chassis with a single active session.

Examples

The following is an example of output from the **debug vpdn pppoe-packet** command:

```
PPPoE control packets debugging is on

1w5d:PPPoE:discovery packet
 contiguous pak, size 74
     FF FF FF FF FF FF 00 10 7B 01 2C D9 88 63 11 09
     00 00 00 04 01 01 00 00 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
1w5d:OUT PADO from PPPoE tunnel
 contiguous pak, size 74
     00 01 09 00 AA AA 03 00 80 C2 00 07 00 00 00 10
     7B 01 2C D9 00 90 AB 13 BC A8 88 63 11 07 00 00
     00 20 01 01 00 00 01 02 00 04 41 67 6E 69 01 ...
1w5d:PPPoE:discovery packet
 contiguous pak, size 74
     00 90 AB 13 BC A8 00 10 7B 01 2C D9 88 63 11 19
     00 00 00 20 01 01 00 00 01 02 00 04 41 67 6E 69
     01 04 00 10 B7 4B 86 5B 90 A5 EF 11 64 A9 BA ...
```

Table 230 describes the significant fields shown in the displays.

Table 230 debug vpdn pppoe-packet Field Descriptions

Field	Descriptions
PPPoE control packets debugging is on	PPPoE debugging of packets is enabled.
1w5d:PPPoE:discovery packet	The host performs a discovery to initiate a PPPoE session.

Table 230 debug vpdn pppoe-packet Field Descriptions (continued)

Field	Descriptions
1w5d:OUT PADO from PPPoE tunnel	The access concentrator sends a PADO to the host.
1w5d:PPPoE:discovery packet	The host performs a discovery to initiate a PPPoE session.
contiguous pak, size 74	Size 74 contiguous packet.

Related Commands

Command	Description
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established session to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpm all

To enable all voice port module (VPM) debugging, use the **debug vpm all** command. Use the **no** form of this command to disable all VPM debugging.

debug vpm all

no debug vpm all

Syntax Description

This command has no arguments or keywords.

Defaults

VPM debugging is not enabled.

Command History

Release	Modification
11.3(1)T	This command was introduced for the Cisco 3600 series.
12.0(7)XK	This command was updated for the Cisco 2600, 3600, and MC3810 series devices.
12.1(2)T	This command was integrated into Cisco IOS release 12.1(2)T.

Usage Guidelines

Use the **debug vpm all** command to enable the complete set of VPM debugging commands: **debug vpm dsp**, **debug vpm error**, **debug vpm port**, **debug vpm spi**, and **debug vpm trunk_sc**.

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

Examples

For sample outputs, refer to the individual commands in this chapter.

Related Commands

Command	Description
debug vpm port	Limits the **debug vpm all** command to a specified port.
show debug	Displays which debug commands are enabled.
debug vpm error	Enables DSP error tracing.
debug vpm voaal2 all	Enables the display of trunk conditioning supervisory component trace information.

debug vpm dsp

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

debug vpm dsp

no debug vpm dsp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

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

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

The following output shows the digits collected:

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

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

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

debug vpm error

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

debug vpm error

no debug vpm error

Syntax Description

This command has no arguments or keywords.

Defaults

VPM debugging is not enabled.

Command History

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

Usage Guidelines

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

Examples

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

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

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

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

Related Commands

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

debug vpm port

To observe the behavior of the Holst state machine, use the **debug vpm port** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

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

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

Syntax Description

slot-number	(Optional) Specifies the slot number in the Cisco router where the voice interface card is installed. Valid entries are from 0 to 3, depending on the router being used and the slot where the voice interface card has been installed.
subunit-number	(Optional) Specifies the subunit on the voice interface card where the voice port is located. Valid entries are 0 or 1.
port	(Optional) Specifies the voice port. Valid entries are 0 or 1.

Command History

Release	Modification
11.3(1)	This command was introduced.

Usage Guidelines

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

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

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

Examples

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

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

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

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

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

Related Commands

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

debug vpm signal

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

debug vpm signal

no debug vpm signal

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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

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

The following output shows that the call is connected:

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

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

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

debug vpm signaling

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

debug vpm signaling

no debug vpm signaling

Syntax Description

This command has no arguments or keywords

Defaults

Disabled

Command Modes

EXEC

Command History

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

Examples

The following example shows output from the command:

```
Router# debug vpm signaling

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

debug vpm spi

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

debug vpm spi

no debug vpm spi

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

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

Examples

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

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

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

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

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

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

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

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

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

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

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

The following output shows the voice quality statistics collected periodically:

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

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

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

debug vpm trunk_sc

To enable the display of trunk conditioning supervisory component trace information, use the **debug vpm trunk_sc** privileged EXEC command. The **no** form of this command disables the display of this information.

debug vpm trunk_sc

no debug vpm trunk_sc

Syntax Description

This command has no arguments or keywords.

Defaults

Trunk conditioning supervisory component trace information is not displayed.

Command History

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

Usage Guidelines

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

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

Examples

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

```
Router# debug vpm trunk_sc

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

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

```
Router# debug vpm trunk_sc

Router# debug vpm port 1/1
```

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

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

Related Commands

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

debug vpm voaal2 all

To display type 1 (voice) and type 3 (control) AAL2 packets sent to and received from the DSP, use the **debug vpm voaal2 all** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

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

no debug vpm voaal2 all

Syntax Description

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

Defaults

Debugging for display of AAL2 packets is not enabled.

Command History

Release	Modification
12.1(1)XA	This command was introduced on the Cisco MC3810 series.
12.1(2)T	This command was integrated into the 12.1(2)T release.

Usage Guidelines

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

Examples

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

```
Router# debug vpm voaal2 all all_dsp

Aal2 trace is on

TYPE 1, len = 43, cid = 25, uui = 14- 19 9D C5 FE FF FF FF FF FF 7E FF 7F
FE FF 7E FF FE FF FF FF FE FE FF 7F FE FF FF 7E FF FE FF FF FF 7F FF FF FF
7D FD FC FF FF FF -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
- 19 13 8 C0 4 5 F 68 -

TYPE 1, len = 43, cid = 25, uui = 4- 19 9C 82 FD FF 7E FF FF FE FD FF 7E
FF FF FF FF FF FF FE FF FF FF FF FF FE FF FF FE FF 7E FE FF FF FF FF FE FF
FF FF FF FF FF FF -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
```

```
- 19 13 8 C0 4 5 F 68 -

TYPE 1, len = 43, cid = 25, uui = 12- 19 9D 8F FF FF 7E FF FE 7E FF FF FF
FF FE FF FF FF FE FE FF FF FF FF FF FE FF FF FF 7E FF FF FF FF FF FD FF FF
FE 7E FF FF FE FF -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
- 19 13 8 C0 4 5 F 68 -

TYPE 1, len = 43, cid = 25, uui = 4- 19 9C 82 FF FF FF FF FF FF FE FF FF
FF FF FF FF 7E FF FF FF FE 7E FF FE FF FF FF FF 7E FE FC FE 7E 7E FF FF FF
FF FF 7E FF FF FF -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
- 19 13 8 C0 4 5 F 68 -

TYPE 1, len = 43, cid = 25, uui = 10- 19 9D 51 FE FF 7E FF FF FF FE 7E FF
FE FF FF FF FF 7E FF 7E 7E FF FF FF FF FF FF FF FF FF 7E FD FF FF FE FF FE
FF FE FE 7E FF FF -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
- 19 13 8 C0 4 5 F 68 -

TYPE 1, len = 43, cid = 25, uui = 2- 19 9C 5C FF FF FE FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF 7E FF FD FF 7E FF FF FE FE FF FE FF FF
7E FF FF FF FE FE -
.
```

Related Commands

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

debug vpm voaal2 type1

To display type 1 (voice) AAL2 packets sent to and received from the DSP, use the **debug vpm voaal2 type1** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

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

no debug vpm voaal2 type1

Syntax Description

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

Defaults

Debugging for display of AAL2 packets is not enabled.

Command History

Release	Modification
12.1(1)XA	This command was introduced on the Cisco MC3810 series.
12.1(2)T	This command was integrated into the 12.1(2)T release.

Usage Guidelines

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

Examples

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

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

```
Router# debug vpm voaal2 type1 to_dsp
Aal2 trace is on

TYPE 1, len = 43, cid = 25, uui = 14- 19 9D C5 FE FF FF FF FF FF 7E FF 7F
FE FF 7E FF FE FF FF FF FE FE FF 7F FE FF FF 7E FF FE FF FF FF 7F FF FF FF
7D FD FC FF FF FF -

TYPE 1, len = 43, cid = 25, uui = 4- 19 9C 82 FD FF 7E FF FF FE FD FF 7E
FF FF FF FF FF FF FE FF FF FF FF FF FE FF FF FE FF 7E FE FF FF FF FF FE FF
FF FF FF FF FF FF -

TYPE 1, len = 43, cid = 25, uui = 12- 19 9D 8F FF FF 7E FF FE 7E FF FF FF
FF FE FF FF FF FE FE FF FF FF FF FF FE FF FF FF 7E FF FF FF FF FF FD FF FF
FE 7E FF FF FE FF -

TYPE 1, len = 43, cid = 25, uui = 4- 19 9C 82 FF FF FF FF FF FF FE FF FF
FF FF FF FF 7E FF FF FF FE 7E FF FE FF FF FF FF 7E FE FC FE 7E 7E FF FF FF
```

```
FF FF 7E FF FF FF -

TYPE 1, len = 43, cid = 25, uui = 10- 19 9D 51 FE FF 7E FF FF FF FE 7E FF
FE FF FF FF FF 7E FF 7E 7E FF FF FF FF FF FF FF FF FF 7E FD FF FF FE FF FE
FF FE FE 7E FF FF -

TYPE 1, len = 43, cid = 25, uui = 2- 19 9C 5C FF FF FE FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF 7E FF FD FF 7E FF FF FE FE FF FE FF FF
7E FF FF FF FE FE -
```

Related Commands

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

debug vpm voaal2 type3

To display type 3 (control) AAL2 packets sent to and received from the DSP, use the **debug vpm voaal2 type3** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

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

no debug vpm voaal2 type3

Syntax Description

alarms	Displays type 3 alarm packets.
alltype3	Displays all type 3 packets.
cas	Displays type 3 CAS signaling packets.
dialed	Displays type 3 dialed digit packets.
faxrelay	(Not supported) Displays type 3 fax relay packets.
state	Displays type 3 user state packets.
all_dsp	Displays messages to and from the DSP.
from_dsp	Displays messages from the DSP.
to_dsp	Displays messages to the DSP.

Defaults

Debugging for display of AAL2 packets is not enabled.

Command History

Release	Modification
12.1(1)XA	This command was introduced on the Cisco MC3810 series device.
12.1(2)T	This command was integrated into the 12.1(2)T release.

Usage Guidelines

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

Examples

The following example shows sample output from the **debug vpm voaal2 type3** command, where the example selection is to display type 3 CAS packets sent from the DSP:

```
Router# debug vpm voaal2 type3 cas from_dsp

Aal2 trace is on
Router#
3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
- 19 13 8 C0 4 5 F 68 -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
```

```
- 19 13 8 C0 4 5 F 68 -

3d21h:TYPE 3, len = 8, cid = 25, uui = 24
3d21h:CAS
 redundancy = 3, timestamp = 4, signal = 5
- 19 13 8 C0 4 5 F 68 -
```

Related Commands

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

debug vsi api

To display information on events associated with the external ATM API interface to the VSI master, use the **debug vsi api** command. The **no** form of this command disables debugging output.

debug vsi api

no debug vsi api

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

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

Examples

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

```
Router# debug vsi api

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

Table 231 describes the significant fields shown in the sample command output shown above.

Table 231 debug vsi api Field Descriptions

Field	Description
vsi_exatm_conn_req	Indicates that a connect or disconnect request was submitted to the VSI master.
0x000C0200	The logical interface identifier of the primary endpoint, in hexadecimal form.
1/35	VPI and VCI of the primary endpoint.
->	Indicates that the expected traffic flow is unidirectional (from the primary endpoint to the secondary endpoint). The other value for this field is <->, which indicates bidirectional traffic flow.
0x000C0100	Logical interface identifier of the secondary endpoint.
1/50	VPI and VCI of the secondary endpoint.

Table 231 debug vsi api Field Descriptions (continued)

Field	Description
`desired state`	Up indicates a connect request; Down indicates a disconnect request.
`status (in vsi_exatm_conn_req output)`	A mnemonic indicating the success or failure of the initial processing of the request. One of following status indications appears: • OK • INVALID_ARGS • NONEXIST_INTF • TIMEOUT • NO_RESOURCES • FAIL OK means only that the request is successfully queued for transmission to the switch; it does not indicate completion of the request.

debug vsi errors

To display information about errors encountered by the VSI master, use the **debug vsi errors** command. The **no** form of this command disables debugging output.

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

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

Syntax Description

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

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

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

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

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

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

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

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

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

Examples

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

```
Router# debug vsi errors

VSI Master: parse error (unexpected param-group contents) in GEN ERROR RSP rcvd on
ATM2/0:0/51 (slave 0)
      errored section is at offset 16, for 2 bytes:
 01.01.00.a0 00.00.00.00 00.12.00.38 00.10.00.34
```

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

Table 232 describes the significant fields shown in the sample command output shown above.

Table 232 debug vsi Errors Field Descriptions

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

debug vsi events

To display information on events that affect entire sessions, and events that affect only individual connections, use the following **debug vsi events** command. The **no** form of this command disables debugging output.

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

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

Syntax Description

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

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

Use the **debug vsi events** command to display information about events associated with the per-session state machines of the VSI master, and the per-connection state machines. If the interface parameter is specified, output is restricted to events associated with the indicated VSI control interface. If the slave number is specified, output is further restricted to events associated with the session with the indicated slave.

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

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

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

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

Examples

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

```
Router# debug vsi events

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

```
VSI Master(session 0 on ATM2/0):
  event SW_GET_CNFG_RSP, state ESTABLISHED -> ESTABLISHED
debug vsi packets
```

Table 233 describes the significant fields shown in the sample command output shown above.

Table 233 Debug VSI Events Field Descriptions

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

debug vsi packets

To display a one-line summary of each VSI message sent and received by the LSC, use the following **debug vsi packets** command. The **no** form of this command disables debugging output.

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

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

Syntax Description

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

Defaults

No default behavior or values

Command History

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

Usage Guidelines

If the interface parameter is specified, output is restricted to messages sent and received on the indicated VSI control interface. If the slave number is specified, output is further restricted to messages sent and received on the session with the indicated slave.

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

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

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

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

Examples

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

```
Router# debug vsi packets

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

Table 234 describes the significant fields shown in the sample command output shown above.

Table 234 debug vsi packets Field Descriptions

Field	Description
session	Session number identifying a particular VSI slave. Numbers begin with zero. Refer to the **show controllers vsi session** command.
ATM2/0	Identifier for the control interface on which the message is sent or received.
sent	Indicates that message is sent by the VSI master.
rcvd	Indicates that message is received by the VSI master.
msg	A mnemonic for the function code from the message header.
0/51	VPI or VCI of the VC (on the control interface) on which the message is sent or received.

debug vsi param-groups

To display the first 128 bytes of each VSI message sent and received by the MPLS LSC (in hexadecimal form), use the following **debug vsi param-groups** command. The **no** form of this command disables debugging output.

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

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

Syntax Description

interface *interface*	Specifies the interface number.
slave *number*	Specifies the slave number (beginning with zero).

Defaults

No default behavior or values.

Command History

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

Usage Guidelines

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

If the interface parameter is specified, output is restricted to messages sent and received on the indicated VSI control interface.

Note If the slave parameter is specified, output is further restricted to messages sent and received on the session with the indicated slave.**param-groups** stands for parameter groups. A parameter group is a component of a VSI message.

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

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

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

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

Examples

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

```
Router# debug vsi param-groups
```

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

Table 235 describes the significant fields shown in the sample command output shown above.

Table 235 debug vsi param-groups Field Descriptions

Field	Description
Outgoing	Indicates that the message is sent by the VSI master.
Incoming	Indicates that the message is received by the VSI master.
bytes	Number of bytes in the message, starting at the VSI header, and excluding the link layer encapsulation.
01.02...	Identifies up to the first 128 bytes of the message, in hexadecimal form.

debug vtemplate

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

debug vtemplate

no debug vtemplate

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug vtemplate

VTEMPLATE Reuse vaccess8, New Recycle queue size:50

VTEMPLATE set default vaccess8 with no ip address

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

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

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

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

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

```
Router# debug vtemplate

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

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

VTEMPLATE Found a dirty vaccess8 clone with vtemplate
VTEMPLATE ************ UNCLONE VACCESS8 **************
VTEMPLATE Unclone to-be-freed vaccess8 command#7
interface Virtual-Access8
default ppp authentication chap
default cdp enable
```

```
default fair-queue 64 256 0
default ip mroute-cache
default ip unnumbered Ethernet0
default encap ppp
default ip address
end

VTEMPLATE set default vaccess8 with no ip address

VTEMPLATE remove cloneblk vtemplate from vaccess8 with vtemplate

VTEMPLATE Add vaccess8 to recycle queue, size=51
```

Table 236 describes the significant fields shown in the display.

Table 236 debug vtemplate Field Descriptions

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

Table 236 debug vtemplate Field Descriptions (continued)

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

debug vtsp all

To show debugging information for all of the **debug vtsp** commands, use the **debug vtsp all** command. Use the **no** form of this command to disable debugging output.

debug vtsp all

no debug vtsp all

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for vtsp is not enabled.

Command History

Release	Modification
12.0(3)T	This command was introduced on the Cisco AS5300 series access servers.
12.0(7)XK	This command was first supported on the Cisco 2600, 3600 and MC3810 series devices.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

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

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

Warning **Using debug vtsp all may severely impact network performance and prevent any faxes from succeeding.**

Related Commands

Command	Description
show debug	Displays which debug commands are enabled.
debug vtsp port	Limits vtsp debug output to a specific voice port.

debug vtsp dsp

To show messages from the DSP to the access server, use the **debug vtsp dsp** EXEC command. Use the **no** form of this command to disable debugging output.

debug vtsp dsp

no debug vtsp dsp

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for vtsp dsp is not enabled.

Command History

Release	Modification
12.0(3)T	This command was introduced on the Cisco AS5300 series access servers.
12.0(7)XK	This command was first supported on the Cisco 2600, 3600, and MC3810 series devices.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

On Cisco AS5300 series access servers

The **debug vtsp dsp** command shows messages from the DSP on the VFC to the router; this command can be useful if you suspect that the VFC is not functional. It is a simple way to check if the VFC is responding to off-hook indications.

On Cisco 2600, 3600, MC3810 series

The **debug vtsp dsp** command shows messages from the DSP to the router.

Examples

The following example shows the collection of DTMF digits from the DSP on a Cisco AS5300 series access server:

```
*Nov 30 00:44:34.491: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=3
*Nov 30 00:44:36.267: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=1
*Nov 30 00:44:36.571: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
*Nov 30 00:44:36.711: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
*Nov 30 00:44:37.147: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=2
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
debug vtsp port	Limits vtsp debug output to a specific voice port.
show debug	Displays which debug commands are enabled.

debug vtsp error

To display processing errors in the voice telephony service provider, use the **debug vtsp error** EXEC command. Use the **no** form of this command to disable VTSP error debugging.

debug vtsp error

no debug vtsp error

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for VTSP errors is not enabled.

Command HistoryCo

Release	Modification
12.0(7)XK	This command was first supported on the Cisco 2600, 3600 and MC3810 series.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

The **debug vtsp error** command can be used to check for mismatches in interface capabilities.

Examples

The following example shows sample output from the **debug vtsp error** command, in which a dialed number is not reachable because it is not configured.

```
Router# deb vtsp error

Voice telephony call control error debugging is on

Router#
*Mar 1 00:21:48.698:cc_api_call_setup_ind (vdbPtr=0x1575AB0,
callInfo={called=,called_oct3=0x81,calling=9999,calling_oct3=0x0,called_oct3a=0x0,
    fdest=0 peer_tag=1},callID=0x15896A4)
*Mar 1 00:21:48.698:cc_api_call_setup_ind type 3 , prot 0
*Mar 1 00:21:48.706:cc_process_call_setup_ind (event=0x16AD0E0) handed call to app
"SESSION"
*Mar 1 00:21:48.706:sess_appl:ev(23=CC_EV_CALL_SETUP_IND), cid(15), disp(0)
*Mar 1 00:21:48.706:sess_appl:ev(SSA_EV_CALL_SETUP_IND), cid(15), disp(0)
*Mar 1 00:21:48.706:ccCallSetContext (callID=0xF, context=0x1632898)
*Mar 1 00:21:48.706:ccCallSetupAck (callID=0xF)
*Mar 1 00:21:48.706:ccGenerateTone (callID=0xF tone=8)
*Mar 1 00:21:49.710:cc_api_call_digit_begin (vdbPtr=0x1575AB0, callID=0xF, digit=5,
flags=0x1, timestamp=0xB1AE6BC4, expiration=0x0)
*Mar 1 00:21:49.710:sess_appl:ev(10=CC_EV_CALL_DIGIT_BEGIN), cid(15), disp(0)
*Mar 1 00:21:49.710:cid(15)st(SSA_CS_MAPPING)ev(SSA_EV_DIGIT_BEGIN)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(0)
*Mar 1 00:21:49.714:ssaIgnore cid(15), st(SSA_CS_MAPPING),oldst(0), ev(10)
*Mar 1 00:21:49.778:cc_api_call_digit (vdbPtr=0x1575AB0, callID=0xF, digit=5,
duration=4165,tag 0, callparty 0 )
*Mar 1 00:21:49.778:sess_appl:ev(9=CC_EV_CALL_DIGIT), cid(15), disp(0)
*Mar 1 00:21:49.778:cid(15)st(SSA_CS_MAPPING)ev(SSA_EV_CALL_DIGIT)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(0)
```

```
*Mar 1 00:21:49.782:ssaDigit
*Mar 1 00:21:49.782:ssaDigit, callinfo , digit 5, tag 0,callparty 0
*Mar 1 00:21:49.782:ssaDigit, calling 9999,result 1
*Mar 1 00:21:49.915:cc_api_call_digit_begin (vdbPtr=0x1575AB0, callID=0xF, digit=5,
flags=0x1, timestamp=0xB1AF6B6C, expiration=0x0)
*Mar 1 00:21:49.915:sess_appl:ev(10=CC_EV_CALL_DIGIT_BEGIN), cid(15), disp(0)
*Mar 1 00:21:49.915:cid(15)st(SSA_CS_MAPPING)ev(SSA_EV_DIGIT_BEGIN)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(0)
*Mar 1 00:21:49.915:ssaIgnore cid(15), st(SSA_CS_MAPPING),oldst(0), ev(10)
*Mar 1 00:21:49.999:cc_api_call_digit (vdbPtr=0x1575AB0, callID=0xF, digit=5,
duration=95,tag 0, callparty 0 )
*Mar 1 00:21:49.999:sess_appl:ev(9=CC_EV_CALL_DIGIT), cid(15), disp(0)
*Mar 1 00:21:50.003:cid(15)st(SSA_CS_MAPPING)ev(SSA_EV_CALL_DIGIT)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(0)
*Mar 1 00:21:50.003:ssaDigit
*Mar 1 00:21:50.003:ssaDigit, callinfo , digit 55, tag 0,callparty 0
*Mar 1 00:21:50.003:ssaDigit, calling 9999,result -1
*Mar 1 00:21:50.003:ccCallDisconnect (callID=0xF, cause=0x1C tag=0x0)
*Mar 1 00:21:50.003:ccCallDisconnect (callID=0xF, cause=0x1C tag=0x0)
*Mar 1 00:21:50.007:vtsp_process_event():prev_state = 0.4 ,
 state = S_WAIT_RELEASE_NC, event = E_CC_DISCONNECT
 Invalid FSM Input on channel 1/1:15
*Mar 1 00:21:52.927:vtsp_process_event():prev_state = 0.7 ,
 state = S_WAIT_RELEASE_RESP, event = E_TSP_CALL_FEATURE_IND
 Invalid FSM Input on channel 1/1:15
*Mar 1 00:21:52.931:cc_api_call_disconnect_done(vdbPtr=0x1575AB0, callID=0xF, disp=0,
tag=0x0)
*Mar 1 00:21:52.931:sess_appl:ev(13=CC_EV_CALL_DISCONNECT_DONE), cid(15), disp(0)
*Mar 1 00:21:52.931:cid(15)st(SSA_CS_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(0)
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
debug vtsp port	Limits vtsp debug output to a specific voice port.
show debug	Displays which debug commands are enabled.

debug vtsp port

To observe the behavior of the VTSP state machine on a specific voice port, use the **debug vtsp port** command. Use the **no** form of the command to turn off the debug function.

For Cisco 2600 and 3600 Series with Analog Voice Ports

debug vtsp port *slot/subunit/port*

no debug vtsp port *slot/subunit/port*

For Cisco 2600 and 3600 series with digital voice ports (with T1 packet voice trunk network modules):

debug vtsp port *slot/port:ds0-group*

no debug vtsp port *slot/port:ds0-group*

For Cisco MC3810 Series with Analog Voice Ports

debug vtsp port *slot/port*

no debug vtsp port *slot/port*

For Cisco MC3810 series with digital voice ports:

debug vtsp port *slot/port*

no debug vtsp port *slot/ds0-group*

Syntax Description

slot/subunit/port	• *slot* specifies a router slot in which a voice network module (NM) is installed. Valid entries are router slot numbers for the particular platform. • *subunit* specifies a voice interface card (VIC) where the voice port is located. Valid entries are 0 and 1. (The VIC fits into the voice network module.) • *port* specifies an analog voice port number. Valid entries are 0 and 1.

For the Cisco 2600 and 3600 series with digital voice ports:

slot/port:ds0-group	Debugs the digital voice port you specify with the *slot/port:ds0-group* designation. *slot* specifies a router slot in which the packet voice trunk network module (NM) is installed. Valid entries are router slot numbers for the particular platform. *port* specifies a T1 or E1 physical port in the voice WAN interface card (VWIC). Valid entries are 0 and 1. (One VWIC fits in an NM.) *ds0-group* specifies a T1 or E1 logical port number. Valid entries are 0 to 23 for T1 and 0 to 30 for E1.

For the Cisco MC3810 Series with Analog Voice Ports

slot/port	Debugs the analog voice port you specify with the *slot/port* designation. *slot* is the physical slot in which the analog voice module (AVM) is installed. The *slot* is always 1 for analog voice ports in the Cisco MC3810 series. *port* specifies an analog voice port number. Valid entries are 1 to 6.

For the Cisco MC3810 series with digital voice ports:

slot:ds0-group	Debugs the digital voice port you specify with the *slot:ds0-group* designation. *slot* specifies the module (and controller). Valid entries are 0 for the MFT (controller 0) and 1 for the DVM (controller 1). *ds0-group* specifies a T1 or E1 logical voice port number. Valid entries are 0 to 23 for T1 and 0 to 30 for E1.

Defaults

Debug vtsp commands are not limited to a specific port.

Command History

Release	Modification
12.0(3)XG	This command was introduced on Cisco 2600 and 3600 series routers.
12.0(3)T	This command was introduced on the Cisco AS5300 series access servers.
12.0(7)XK	This command was first supported on the Cisco MC3810 series.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

Use the **debug vtsp port** command to limit the debug output to a particular voice port. The debug output can be quite voluminous for a single channel. The entire vtsp debug output form a platform with 12 voice ports might create problems. Use this debug with any or all of the other debug modes.

Execution of **no debug vtsp all** will turn off all VTSP-level debugging. It is usually a good idea to turn off all debugging and then enter the debug commands you are interested in one by one. This will help to avoid confusion about which ports you are actually debugging.

Examples

The following example shows sample output from the **debug vtsp port 1/1/0** command:

```
Router# debug vtsp port 1/1/0

*Mar 1 03:17:33.691: vtsp_tsp_call_setup_ind (sdb=0x613FD514, tdm_info=0x0,
  tsp_info=0x613FD438, calling_number= called_number= redirect_number=): peer_tag=1110
*Mar 1 03:17:33.691: vtsp_do_call_setup_ind
*Mar 1 03:17:33.691: dsp_close_voice_channel: [] packet_len=8 channel_id=1
  packet_id=75
*Mar 1 03:17:33.691: dsp_open_voice_channel: [] packet_len=12
  channel_id=1 packet_id=74 alaw_ulaw_select=0 transport_protocol=2
*Mar 1 03:17:33.695: dsp_set_playout_delay: [] packet_len=18
```

```
  channel_id=1 packet_id=76 mode=1 initial=60 min=4 max=200 fax_nom=300
*Mar 1 03:17:33.695: dsp_echo_canceller_control: [] packet_len=10 channel_id=1
  packet_id=66 flags=0x0
*Mar 1 03:17:33.695: dsp_set_gains: [] packet_len=12 channel_id=1 packet_id=91
  in_gain=0 out_gain=65506
*Mar 1 03:17:33.695: dsp_vad_enable: [] packet_len=10 channel_id=1 packet_id=78
  thresh=-38
*Mar 1 03:17:33.695: vtsp_process_event(): [, 0.S_SETUP_INDICATED, E_CC_PROCEEDING]
*Mar 1 03:17:33.699: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
  E_CC_BRIDGE]act_bridge
*Mar 1 03:17:33.699: vtsp_ring_noan_timer_start: 1185370
*Mar 1 03:17:33.699: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
  E_CC_CAPS_IND]act_caps_ind
*Mar 1 03:17:33.699: act_caps_ind: Encap 2, Vad 2, Codec 0x1000, CodecBytes 60,
       FaxRate 2, FaxBytes 30,
       Sub-channel 10, Bitmask 0x0 SignalType 2
*Mar 1 03:17:33.703: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
  E_CC_CAPS_ACK]act_caps_ack
*Mar 1 03:17:33.703: dsp_idle_mode: [] packet_len=8 channel_id=1 packet_id=68
*Mar 1 03:17:33.703: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
  E_CC_CONNECT]act_connect
*Mar 1 03:17:33.703: vtsp_ring_noan_timer_stop: 1185370
*Mar 1 03:17:33.911: vtsp_process_event(): [, 0.S_CONNECT, E_DSPRM_PEND_SUCCESS]
  act_pend_codec_success
*Mar 1 03:17:33.911: dsp_close_voice_channel: [] packet_len=8 channel_id=1
  packet_id=75
*Mar 1 03:17:33.911: dsp_open_voice_channel: [] packet_len=12 channel_id=1
  packet_id=74 alaw_ulaw_select=0 transport_protocol=2
*Mar 1 03:17:33.911: dsp_set_playout_delay: [] packet_len=18 channel_id=1 packet_id=76
  mode=1 initial=60 min=4 max=200 fax_nom=300
*Mar 1 03:17:33.911: dsp_echo_canceller_control: [] packet_len=10 channel_id=1
  packet_id=66 flags=0x0
*Mar 1 03:17:33.911: dsp_set_gains: [] packet_len=12 channel_id=1 packet_id=91
  in_gain=0 out_gain=65506
*Mar 1 03:17:33.911: dsp_vad_enable: [] packet_len=10 channel_id=1 packet_id=78
  thresh=-38
*Mar 1 03:17:33.911: dsp_encap_config: [] packet_len=24 channel_id=1 packet_id=
 92 TransportProtocol 3 SID_support=0 sequence_number=0 rotate_flag=0 header_bytes 0xA0
*Mar 1 03:17:33.915: dsp_voice_mode: [] packet_len=22 channel_id=1 packet_id=73
 coding_type=14 voice_field_size=60 VAD_flag=1 echo_length=128
 comfort_noise=1 fax_detect=1 digit_relay=0
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
show debug	Displays which debug commands are enabled.

debug vtsp send-nse

To trigger the VTSP software module to send a triple redundant NSE, use the **debug vtsp send-nse** EXEC command. Use the **no debug vtsp send-nse** to disable this action.

debug vtsp send-nse

no debug vtsp send-nse

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco MC3810 and the Cisco 3600 series routers (except the Cisco 3620) in a private release that was not generally available.

Examples

The following example shows the VTSP software module set to send a triple redundant NSE:

```
Router# debug vtsp send-nse
```

Related Commands

Command	Description
debug rtpspi all	Debugs all RTP SPI errors, sessions, and in/out functions.
debug rtpspi errors	Debugs RTP SPI errors.
debug rtpspi inout	Debugs RTP SPI in/out functions.
debug rtpspi send-nse	Triggers the RTP SPI to send a triple redundant NSE.
debug sgcp errors	Debugs SGCP errors.
debug sgcp events	Debugs SGCP events.
debug sgcp packet	Debugs SGCP packets.

debug vtsp session

To trace how the router interacts with the DSP based on the signaling indications from the signaling stack and requests from the application, use the **debug vtsp session** command. Use the **no** form of this command to turn off the debug function.

debug vtsp session

no debug vtsp session

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for vtsp session is not enabled.

Command History

Release	Modification
12.0(3)T	This command was introduced on the Cisco AS5300 series access servers.
12.0(7)XK	This command was first supported on the Cisco 2600, 3600 and MC3810 series.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

The **debug vtsp session** command traces how the router interacts with the DSP based on the signaling indications from the signaling stack and requests from the application. This debug command displays information about how each network indication and application request is handled, signaling indications, and DSP control messages.

This debug level shows the internal workings of the voice telephony call state machine.

Examples

The following example shows sample output from the **debug vtsp session** command, in which the call has been accepted and the system is checking for incoming dial-peer matches:

```
*Nov 30 00:46:19.535: vtsp_tsp_call_accept_check (sdb=0x60CD4C58,
calling_number=408 called_number=1): peer_tag=0
*Nov 30 00:46:19.535: vtsp_tsp_call_setup_ind (sdb=0x60CD4C58,
tdm_info=0x60B80044, tsp_info=0x60B09EB0, calling_number=408 called_number=1):
peer_tag=1
```

The following example shows sample output from the **debug vtsp session** command, in which a DSP has been allocated to handle the call and has indicated the call to the higher layer code:

```
*Nov 30 00:46:19.535: vtsp_do_call_setup_ind:
*Nov 30 00:46:19.535: dsp_open_voice_channel: [0:D:12] packet_len=12
channel_id=8737 packet_id=74 alaw_ulaw_select=0 transport_protocol=2
*Nov 30 00:46:19.535: dsp_set_playout_delay: [0:D:12] packet_len=18
channel_id=8737 packet_id=76 mode=1 initial=60 min=4 max=200 fax_nom=300
*Nov 30 00:46:19.535: dsp_echo_canceller_control: [0:D:12] packet_len=10
channel_id=8737 packet_id=66 flags=0x0
*Nov 30 00:46:19.539: dsp_set_gains: [0:D:12] packet_len=12 channel_id=8737
packet_id=91 in_gain=0 out_gain=0
```

```
*Nov 30 00:46:19.539: dsp_vad_enable: [0:D:12] packet_len=10 channel_id=8737
packet_id=78 thresh=-38
*Nov 30 00:46:19.559: vtsp_process_event: [0:D:12, 0.3, 13] act_setup_ind_ack
```

The following example shows sample output from the **debug vtsp session** command, in which the higher layer code has accepted the call, placed the DSP in DTMF mode, and collected digits:

```
*Nov 30 00:46:19.559: dsp_voice_mode: [0:D:12] packet_len=20 channel_id=8737
packet_id=73 coding_type=1 voice_field_size=160 VAD_flag=0 echo_length=64
comfort_noise=1 fax_detect=1
*Nov 30 00:46:19.559: dsp_dtmf_mode: [0:D:12] packet_len=10 channel_id=8737
packet_id=65 dtmf_or_mf=0
*Nov 30 00:46:19.559: dsp_cp_tone_on: [0:D:12] packet_len=30 channel_id=8737
packet_id=72 tone_id=3 n_freq=2 freq_of_first=350 freq_of_second=440
amp_of_first=4000 amp_of_second=4000 direction=1 on_time_first=65535
off_time_first=0 on_time_second=65535 off_time_second=0
*Nov 30 00:46:19.559: vtsp_timer: 278792
*Nov 30 00:46:22.059: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.059: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.059: vtsp_timer: 279042
*Nov 30 00:46:22.363: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.363: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.363: vtsp_timer: 279072
*Nov 30 00:46:22.639: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.639: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.639: vtsp_timer: 279100
*Nov 30 00:46:22.843: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.843: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.843: vtsp_timer: 279120
*Nov 30 00:46:23.663: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:23.663: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:23.663: vtsp_timer: 279202
```

The following example shows sample output from the **debug vtsp session** command, in which the call proceeded and DTMF was disabled:

```
*Nov 30 00:46:23.663: vtsp_process_event: [0:D:12, 0.4, 15] act_dcollect_proc
*Nov 30 00:46:23.663: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:23.663: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
```

The following example shows sample output from the **debug vtsp session** command, in which the telephony call leg was conferenced with the packet network call leg, and the telephony call leg has performed capabilities exchange with the network-side call leg:

```
*Nov 30 00:46:23.699: vtsp_process_event: [0:D:12, 0.5, 17] act_bridge
*Nov 30 00:46:23.699: vtsp_process_event: [0:D:12, 0.5, 22] act_caps_ind
*Nov 30 00:46:23.699: vtsp_process_event: [0:D:12, 0.5, 23] act_caps_ack
Go into voice mode with codec indicated in caps exchange.
*Nov 30 00:46:23.699: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:23.699: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
*Nov 30 00:46:23.699: dsp_voice_mode: [0:D:12] packet_len=20 channel_id=8737
packet_id=73 coding_type=6 voice_field_size=20 VAD_flag=1 echo_length=64
comfort_noise=1 fax_detect=1
```

The following example shows sample output from the **debug vtsp session** command in which the call has been connected at remote end:

```
*Nov 30 00:46:23.779: vtsp_process_event: [0:D:12, 0.5, 10] act_connect
```

The following example shows sample output from the **debug vtsp session** command in which disconnect was indicated and passed to upper layer:

```
*Nov 30 00:46:30.267: vtsp_process_event: [0:D:12, 0.11, 5] act_generate_disc
```

The following example shows sample output from the **debug vtsp session** command, in which the conference was torn down and the disconnect handshake was completed:

```
*Nov 30 00:46:30.267: vtsp_process_event: [0:D:12, 0.11, 18] act_bdrop
*Nov 30 00:46:30.267: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:30.267: vtsp_process_event: [0:D:12, 0.11, 20] act_disconnect
*Nov 30 00:46:30.267: dsp_get_error_stat: [0:D:12] packet_len=10 channel_id=0
packet_id=6 reset_flag=1
*Nov 30 00:46:30.267: vtsp_timer: 279862
```

The following example shows sample output from the **debug vtsp session** command, in which the final DSP statistics were retrieved:

```
*Nov 30 00:46:30.275: vtsp_process_event: [0:D:12, 0.17, 30] act_get_error
*Nov 30 00:46:30.275: 0:D:12: rx_dropped=0 tx_dropped=0 rx_control=353
tx_control=338 tx_control_dropped=0 dsp_mode_channel_1=2 dsp_mode_channel_2=0
c[0]=71 c[1]=71 c[2]=71 c[3]=71 c[4]=68 c[5]=71 c[6]=68 c[7]=73 c[8]=83 c[9]=84
c[10]=87 c[11]=83 c[12]=84 c[13]=87 c[14]=71 c[15]=6
*Nov 30 00:46:30.275: dsp_get_levels: [0:D:12] packet_len=8 channel_id=8737
packet_id=89
*Nov 30 00:46:30.279: vtsp_process_event: [0:D:12, 0.17, 34] act_get_levels
*Nov 30 00:46:30.279: dsp_get_tx_stats: [0:D:12] packet_len=10 channel_id=8737
packet_id=86 reset_flag=1
*Nov 30 00:46:30.287: vtsp_process_event: [0:D:12, 0.17, 31] act_stats_complete
*Nov 30 00:46:30.287: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:30.287: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
*Nov 30 00:46:30.287: vtsp_timer: 279864
```

The following example shows sample output from the **debug vtsp session** command, in which the DSP channel was closed and released:

```
*Nov 30 00:46:30.287: vtsp_process_event: [0:D:12, 0.18, 6] act_wrelease_release
*Nov 30 00:46:30.287: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:30.287: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
*Nov 30 00:46:30.287: dsp_close_voice_channel: [0:D:12] packet_len=8
channel_id=8737 packet_id=75
*Nov 30 00:46:30.287: vtsp_process_event: [0:D:12, 0.16, 42] act_terminate
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
debug vtsp port	Limits vtsp debug output to a specific voice port.
show debug	Displays which debug commands are enabled.

debug vtsp stats

To debug periodic statistical-information-request messages sent and received from the DSP during a call, use the **debug vtsp stats** command. Use the **no** form of this command to turn off the debug function.

debug vtsp stats

no debug vtsp stats

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for vtsp stats is not enabled.

Command History

Release	Modification
12.0(3)T	This command was introduced on the Cisco AS5300 series access servers.
12.0(7)XK	This command was first supported on the Cisco 2600, 3600 and MC3810 series.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

The **debug vtsp stats** command generates a collection of DSP statistics for generating RTCP packets and a collection of other statistical information.

Examples

The following example shows sample **debug vtsp stats** output:

```
*Nov 30 00:53:26.499: vtsp_process_event: [0:D:14, 0.11, 19] act_packet_stats
*Nov 30 00:53:26.499: dsp_get_voice_playout_delay_stats: [0:D:14] packet_len=10
channel_id=8753 packet_id=83 reset_flag=0
*Nov 30 00:53:26.499: dsp_get_voice_playout_error_stats: [0:D:14] packet_len=10
channel_id=8753 packet_id=84 reset_flag=0
*Nov 30 00:53:26.499: dsp_get_rx_stats: [0:D:14] packet_len=10 channel_id=8753
packet_id=87 reset_flag=0
*Nov 30 00:53:26.503: vtsp_process_dsp_message: MSG_TX_GET_VOICE_PLAYOUT_DELAY:
clock_offset=-1664482334 curr_rx_delay_estimate=69 low_water_mark_rx_delay=69
high_water_mark_rx_delay=70
*Nov 30 00:53:26.503: vtsp_process_event: [0:D:14, 0.11, 28]
act_packet_stats_res
*Nov 30 00:53:26.503: vtsp_process_dsp_message: MSG_TX_GET_VOICE_PLAYOUT_ERROR:
predective_concelement_duration=0 interpolative_concelement_duration=0
silence_concelement_duration=0 retroactive_mem_update=0
buf_overflow_discard_duration=10 num_talkspurt_detection_errors=0
*Nov 30 00:53:26.503: vtsp_process_event: [0:D:14, 0.11, 29]
act_packet_stats_res
*Nov 30 00:53:26.503: vtsp_process_dsp_message: MSG_TX_GET_RX_STAT:
num_rx_pkts=152 num_early_pkts=-2074277660 num_late_pkts=327892
num_signalling_pkts=0 num_comfort_noise_pkts=0 receive_durtation=3130
voice_receive_duration=2970 fax_receive_duration=0 num_pack_ooseq=0
num_bad_header=0
*Nov 30 00:53:26.503: vtsp_process_event: [0:D:14, 0.11, 32]
```

```
act_packet_stats_res
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
debug vtsp port	Limits vtsp debug output to a specific voice port.
show debug	Displays which debug commands are enabled.

debug vtsp vofr subframe

To display the first 10 bytes (including header) of selected VoFR subframes for the interface, use the **debug vtsp vofr subframe** command. Use the **no** form of the command to turn off the debug function.

debug vtsp vofr subframe *payload* [**from-dsp**] [**to-dsp**]

no debug vtsp vofr subframe

Syntax Description

payload	Number used to selectively display subframes of a specific payload. Payload types are: 0: Primary Payload - WARNING! This option may cause network instability **1**: Annex-A **2**: Annex-B **3**: Annex-D **4**: All other payloads **5**: All payloads - WARNING! This option may cause network instability
from-dsp	Displays only the subframes received from the DSP.
to-dsp	Displays only the subframes going to the DSP.

Defaults

Debugging for vtsp vofr subframe is not enabled.

Command History

Release	Modification
12.0(3)XG, 12.0(4)T	This command was introduced on the Cisco 2600 and 3600 series.
12.0(4)T	This command was integrated into 12.0(4)T release.
12.0(7)XK	This command was first supported on the Cisco MC3810 series.
12.1(2)T	This command was integrated into 12.1(2)T release.

Usage Guidelines

Each debug output displays the first 10 bytes of the FRF.11 subframe, including header bytes. The **from-dsp** and **to-dsp** options can be used to limit the debugs to a single direction. If not specified, debugs are displayed for subframes when they are received from the DSP and before they are sent to the DSP.

Use extreme caution in selecting payload options 0 and 6. These options may cause network instability.

Examples

The following example shows sample output from the **debug vtsp vofr subframe** command:

```
Router# debug vtsp vofr subframe 2
vtsp VoFR subframe debugging is enabled for payload 2 to and from DSP 3620_vofr#
*Mar 6 18:21:17.413:VoFR frame received from Network (24 bytes):9E 02 19 AA AA AA AA
   AA AA AA
*Mar 6 18:21:17.449:VoFR frame received from DSP (18 bytes):9E 02 19 AA AA AA AA AA AA
   AA
*Mar 6 18:21:23.969:VoFR frame received from Network (24 bytes):9E 02 19 AA AA AA AA
   AA AA AA
*Mar 6 18:21:24.005:VoFR frame received from DSP (18 bytes):9E 02 19 AA AA AA AA AA AA
   AA
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
debug vtsp port	Limits vtsp debug output to a specific voice port.
show debug	Displays which debug commands are enabled.

debug vtsp tone

To display debug messages showing the types of tones generated by the VoIP gateway, use the **debug vtsp tone** command. To disable the debug messages, use the **no** form of this command.

debug vtsp tone

no debug vtsp tone

Syntax Description

This command has no keywords or arguments.

Defaults

Tone generation messages are not enabled.

Command History

Release	Modification
12.1(3)XI	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Examples

The following example shows that a ringback tone was generated by the VoIP gateway:

```
Router# debug vtsp tone
*Jan 1 16:33:52.395:act_alert:Tone Ring Back generated in direction Network

*Jan 1 16:33:52.399:ISDN Se0:23:TX -> ALERTING pd = 8 callref = 0x9816
```

Related Commands

Command	Description
debug vtsp dsp	Shows messages from the Digital Signal Processor (DSP) on the modem to the router.
debug vtsp session	Traces how the router interacts with the Digital Signal Processor (DSP), based on the signaling indications from the signaling stack and requests from the application.

debug x25

To display information about X.25 traffic, use one of the following **debug x25** privileged EXEC commands. The commands allow you to display all information or an increasingly restrictive part of the information.

Caution This command is processor intensive and can render the router useless. Use this command only when the aggregate of all reportable X.25 traffic is fewer than five packets per second (pps). The generic forms of this command should be restricted to low-speed, low-usage links running at less than 19.2 kbps. Because the **debug x25 vc** command and the **debug x25 vc events** command display traffic for only a small subset of virtual circuits, they are safer to use under heavy traffic conditions, as long as events for that virtual circuit are fewer than 25 pps.

To display information about all X.25 traffic, including traffic for X.25, Connection Mode Network Service (CMNS), and X.25 over TCP (XOT) services, use the **debug x25** command (default **all**). Use the **no** form of this command to disable debugging output.

debug x25

no debug x25

To display information about all X.25 traffic except data and resource record packets, use the **debug x25 events** command. Use the **no** form of this command to disable debugging output.

debug x25 events

no debug x25 events

To display information about a specific X.25 service class, use the following form of the **debug x25** command. Use the **no** form of this command to disable debugging output.

debug x25 [**only** | **cmns** | **xot**] [**events** | **all**]

no debug x25 [**only** | **cmns** | **xot**] [**events** | **all**]

To display information about a specific X.25 or CMNS context, use the following form of the **debug x25** command. Use the **no** form of this command to disable debugging output.

debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} [**events** | **all**]

no debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} [**events** | **all**]

To display information about a specific X.25 or CMNS virtual circuit, use the following form of the **debug x25** command. Use the **no** form of this command to disable debugging output.

debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} **vc** *number* [**events** | **all**]

no debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} **vc** *number* [**events** | **all**]

To display information about traffic for all virtual circuits using a given number, use the following form of the **debug x25** command. The **no** form of this command removes the filter for a particular virtual circuit from the **debug x25 all** or **debug x25 events** output. Use the **no** form of this command to disable debugging output.

debug x25 vc *number* [**events** | **all**]

no debug x25 vc *number* [**events** | **all**]

To display information about traffic to or from a specific XOT host, use the following form of the **debug x25 xot** command. Use the **no** form of this command to disable debugging output.

debug x25 xot [**remote** *ip-address* [**port** *number*]] [**local** *ip-address* [**port** *number*]] [**events** | **all**]

no debug x25 xot [**remote** *ip-address* [**port** *number*]] [**local** *ip-address* [**port** *number*]] [**events** | **all**]

Use the **debug x25** command with the **aodi** keyword to display information about an interface running PPP over an X.25 session. The **no** form of this command disables debugging output. Use the **no** form of this command to disable debugging output.

debug x25 aodi

no debug x25 aodi

Syntax Description

events	(Optional) Displays all traffic except Data and Receiver Ready (RR) packets.
only | **cmns** | **xot**	(Optional) Displays information about the specified services: X.25 only, CMNS, or XOT.
all	(Optional) Displays all traffic.
serial-interface	X.25 serial interface.
cmns-interface **mac** *mac-address*	MAC address of the CMNS interface and remote host. The interface type can be Ethernet, Token Ring, or FDDI.
vc *number*	Virtual circuit number, in the range 1 to 4095.
remote *ip-address* [**port** *number*]	(Optional) Remote IP address and, optionally, a port number in the range 1 to 65535.
local *ip-address* [**port** *number*]	(Optional) Local host IP address and, optionally, a port number in the range 1 to 65535.
aodi	Causes the **debug x25** command to display Always On/Dynamic ISDN (AO/DI) events and processing information.

Defaults

The default is that all traffic is displayed.

Command History

Release	Modification
10.0	This command was introduced.
12.0(5)T	For DNS-based X.25 routing, additional functionality was added to the **debug x25 events** command to describe the events occurring while resolving the X.25 address to an IP address using a DNS server. The **debug domain** command can be used along with **debug x25 events** to observe the whole DNS-based X.25 routing data flow. (For more details, see "debug x25 events for DNS-Based X.25 Routing" in the "Examples" section.)
12.0(7)T	For the X.25 CUGs feature, functionality was added to the **debug x25 events** command to describe events occurring during CUG activity. (For more details, see "debug x25 events for X.25 CUGs" in the "Examples" section.)

Usage Guidelines

This command is particularly useful for diagnosing problems encountered when placing calls. The **debug x25 all** output includes data, control messages, and flow control packets for all virtual circuits of the router.

All **debug x25** command forms can take either the **events** or **all** keyword. The keyword **all** is the default and causes all packets meeting the other debug criteria to be reported. The keyword **events** omits reports of any Data or Receiver Ready (RR) flow control packets; the normal flow of data and RR packets is commonly large and less interesting to the user, so event reporting can significantly decrease the processor load induced by debug reporting.

The **debug x25 interface** command is useful for diagnosing problems encountered with a single X.25 or CMNS host or virtual circuit.

Because no interface is specified by the **debug x25 vc** command, traffic on any virtual circuit that has the specified number is reported.

Virtual circuit zero (**vc 0**) cannot be specified. It is used for X.25 service messages, such as RESTART packets, not virtual circuit traffic. Service messages can be monitored only when no virtual circuit filter is used.

The **debug x25 xot** output allows you to restrict the debug output reporting to XOT traffic for one or both hosts or host/port combinations. Because each XOT virtual circuit uses a unique TCP connection, an XOT debug request that specifies both host addresses and ports will report traffic only for that virtual circuit. Also, you can restrict reporting to sessions initiated by the local or remote router by specifying 1998 for the remote or local port. (XOT connections are received on port 1998.)

Use the **debug x25 aodi** command to display interface PPP events running over an X.25 session and to debug X.25 connections between a client and server configured for AO/DI.

Examples

The following is sample output from the **debug x25** command, displaying output concerning the functions X.25 restart, call setup, data exchange, and clear:

```
Router# debug x25

Serial0: X.25 I R/Inactive Restart (5) 8 lci 0
  Cause 7, Diag 0 (Network operational/No additional information)
Serial0: X.25 O R3 Restart Confirm (3) 8 lci 0
Serial0: X.25 I P1 Call (15) 8 lci 1
From(6): 170091 To(6): 170090
   Facilities: (0)
   Call User Data (4): 0xCC000000 (ip)
Serial0: X.25 O P3 Call Confirm (3) 8 lci 1
```

```
Serial0: X.25 I D1 Data (103) 8 lci 1 PS 0 PR 0
Serial0: X.25 O D1 Data (103) 8 lci 1 PS 0 PR 1
Serial0: X.25 I P4 Clear (5) 8 lci 1
  Cause 9, Diag 122 (Out of order/Maintenance action)
Serial0: X.25 O P7 Clear Confirm (3) 8 lci 1
```

Table 237 describes the fields shown in the display.

Table 237 debug x25 Field Descriptions

Field	Description
Serial0	Interface on which the X.25 event occurred.
X.25	Type of event this message describes.
I	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.
R3	State of the service or virtual circuit (VC). Possible values follow: • R/Inactive—Packet layer awaiting link layer service • R1—Packet layer ready • R2—Data terminal equipment (DTE) restart request • R3—Data circuit-terminating equipment (DCE) restart indication • P/Inactive—VC awaiting packet layer service • P1—Idle • P2—DTE waiting for DCE to connect CALL • P3—DCE waiting for DTE to accept CALL • P4—Data transfer • P5—CALL collision • P6—DTE clear request • P7—DCE clear indication • D/Inactive—VC awaiting setup • D1—Flow control ready • D2—DTE reset request • D3—DCE reset indication See Annex B of the *ITU-T Recommendation X.25* for more information on these states.

Table 237 debug x25 Field Descriptions (continued)

Field	Description
Restart	The type of X.25 packet. Possible values follow: • R Events —Restart —Restart Confirm —Diagnostic • P Events —Call —Call Confirm —Clear —Clear Confirm • D Events —Reset —Reset Confirm • D1 Events —Data —RNR (Receiver Not Ready) —RR (Receiver Ready) —Interrupt —Interrupt Confirm • XOT Overhead —PVC Setup
(5)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 or 128.
lci 0	VC number. See Annex A of the *ITU-T Recommendation X.25* for information on VC assignment.
Cause 7	Code indicating the event that triggered the packet. The Cause field can only appear in entries for Clear, Reset, and Restart packets. Possible values for the Cause field can vary, depending on the type of packet. Refer to the “X.25 Cause and Diagnostic Codes” appendix for an explanation of these codes.
Diag 0	Code providing an additional hint as to what, if anything, went wrong. The Diag field can only appear in entries for Clear, Diagnostic (as “error 0”), Reset, and Restart packets. Refer to the “X.25 Cause and Diagnostic Codes” appendix for an explanation of these codes.
(Network operational/ No additional information)	The standard explanations of the Cause and Diagnostic codes (*cause/diag*).

The following example shows a sequence of increasingly restrictive **debug x25** commands:

```
Router# debug x25
X.25 packet debugging is on

Router# debug x25 events
X.25 special event debugging is on

Router# debug x25 interface serial 0
X.25 packet debugging is on
X.25 debug output restricted to interface Serial0

Router# debug x25 vc 1024
X.25 packet debugging is on
X.25 debug output restricted to VC number 1024

Router# debug x25 interface serial 0 vc 1024
X.25 packet debugging is on
X.25 debug output restricted to interface Serial0
X.25 debug output restricted to VC number 1024

Router# debug x25 interface serial 0 vc 1024 events
X.25 special event debugging is on
X.25 debug output restricted to interface serial 0
X.25 debug output restricted to VC number 1024
```

The following examples show the normal sequence of events for both the AO/DI client and server sides:

Client Side

```
Router# debug x25 aodi
PPP-X25: Virtual-Access1: Initiating AODI call request
PPP-X25: Bringing UP X.25 AODI VC
PPP-X25: AODI Client Call Confirm Event Received
PPP-X25: Cloning interface for AODI is Di1
PPP-X25: Queuing AODI Client Map Event
PPP-X25: Event:AODI Client Map
PPP-X25: Created interface Vi2 for AODI service
PPP-X25: Attaching primary link Vi2 to Di1
PPP-X25: Cloning Vi2 for AODI service using Di1
PPP-X25: Vi2: Setting the PPP call direction as OUT
PPP-X25: Vi2: Setting vectors for RFC1598 operation on BRI3/0:0 VC 0
PPP-X25: Vi2: Setting the interface default bandwidth to 10 Kbps
PPP-X25: Virtual-Access2: Initiating AODI call request
PPP-X25: Bringing UP X.25 AODI VC
PPP-X25: AODI Client Call Confirm Event Received
```

Server Side

```
Router# debug x25 aodi
PPP-X25: AODI Call Request Event Received
PPP-X25: Event:AODI Incoming Call Request
PPP-X25: Created interface Vi1 for AODI service
PPP-X25: Attaching primary link Vi1 to Di1
PPP-X25: Cloning Vi1 for AODI service using Di1
PPP-X25: Vi1: Setting vectors for RFC1598 operation on BRI3/0:0 VC 1
PPP-X25: Vi1: Setting the interface default bandwidth to 10 Kbps
PPP-X25: Binding X.25 VC 1 on BRI3/0:0 to Vi1
```

debug x25 events for X.25 CUGs

The following example of the **debug x25 events** command shows output related to the X.25 CUGs feature. It shows messages concerning a DCE rejecting a call because the selected network CUG had not been subscribed to by the caller.

```
Router# debug x25 events
00:48:33:Serial1:X.25 I R1 Call (14) 8 lci 1024
00:48:33: From (3):111 To (3):444
00:48:33: Facilities:(2)
00:48:33:  Closed User Group (basic):40
00:48:33: Call User Data (4):0x01000000 (pad)
00:48:33:X.25 Incoming Call packet, Closed User Group (CUG) protection, selected network
CUG not subscribed
00:48:33:Serial1:X.25 O R1 Clear (5) 8 lci 1024
00:48:33: Cause 11, Diag 65 (Access barred/Facility code not allowed)
```

debug x25 events for DNS-Based X.25 Routing

The following example of the **debug x25 events** command shows output related to the DNS-Based X.25 Routing feature. It shows messages concerning access of the DNS server. In the following example, nine alternate addresses for one XOT path are entered in the DNS server database. All nine addresses are returned to the host cache of the router by the DNS server. However, only six addresses will be used during the XOT switch attempt, because this is the limit that XOT allows.

```
Router# debug x25 events
00:18:25:Serial1:X.25 I R1 Call (11) 8 lci 1024
00:18:25: From (0): To (4):444
00:18:25: Facilities:(0)
00:18:25: Call User Data (4):0x01000000 (pad)
00:18:25:X.25 host name sent for DNS lookup is "444"
00:18:26:%3-TRUNCATE_ALT_XOT_DNS_DEST:Truncating excess XOT addresses (3)
returned by DNS
00:18:26:DNS got X.25 host mapping for "444" via network
00:18:32:[10.1.1.8 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:38:[10.1.1.7 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:44:[10.1.1.6 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:50:[10.1.1.5 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:56:[10.1.1.4 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:20:04:[10.1.1.3,1998/10.1.1.3,11007]:XOT O P2 Call (17) 8 lci 1
00:20:04: From (0): To (4):444
00:20:04: Facilities:(6)
00:20:04:  Packet sizes:128 128
00:20:04:  Window sizes:2 2
00:20:04: Call User Data (4):0x01000000 (pad)
00:20:04:[10.1.1.3,1998/10.1.1.3,11007]:XOT I P2 Call Confirm (11) 8 lci 1
00:20:04: From (0): To (0):
00:20:04: Facilities:(6)
00:20:04:  Packet sizes:128 128
00:20:04:  Window sizes:2 2
00:20:04:Serial1:X.25 O R1 Call Confirm (5) 8 lci 1024
00:20:04: From (0): To (0):
00:20:04: Facilities:(0)
```

Related Commands

Command	Description
debug ppp bap	Displays general BACP transactions.
debug ppp bap negotiation	Displays general BACP transactions, and successive steps in negotiations between peers.
debug ppp multilink	Displays information about important multilink events.
debug ppp multilink negotiation	Displays information about important multilink events and events affecting multilink groups controlled by BACP.

debug x25 annexg

To display information about Annex G (X.25 over Frame Relay) events, use the **debug x25 annexg** command. To disable debugging output, use the **no** form of this command.

debug x25 annexg

no debug x25 annexg

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0 T	This command was introduced.

Usage Guidelines

It is generally recommended that the **debug x25 annexg** command be used only when specifically requested by Cisco TAC to obtain information about a problem with an Annex G configuration. The messages displayed by the **debug x25 annexg** command are meant to aid in the diagnosing of internal errors.

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

Examples

The following example shows sample output for the **debug x25 annexg** command for a Frame Relay data-link connection identifier (DLCI) configured for Annex G operation:

```
Router# debug x25 annexg

Jul 31 05:23:20.316:annexg_process_events:DLCI 18 attached to interface Serial2/0:0 is
ACTIVE
Jul 31 05:23:20.316:annexg_ctxt_create:Creating X.25 context over Serial2/0:0 (DLCI:18
using X.25 profile:OMC), type 10, len 2, addr 00 12
Jul 31 05:23:20.316:annexg_create_lower_layer:Se2/0:0 DLCI 18, payload 1606, overhead 2
Jul 31 05:23:20.320:annexg_restart_tx:sending pak to Serial2/0:0
Jul 31 05:23:23.320:annexg_restart_tx:sending pak to Serial2/0:0
```

Table 238 describes significant fields shown in the display.

Table 238 debug x25 annexg Field Descriptions

Field	Description
payload	Amount of buffer space available per message before adding Frame Relay and device-specific headers.
overhead	The length of the Frame Relay header and any device-specific header that may be needed.

Related Commands

Command	Description
debug x25	Displays information about X.25 traffic.

debug x28

To monitor error information and X.28 connection activity, use the **debug x28** privileged privileged EXEC command. The **no** form of this command disables debugging output.

debug x28

no debug x28

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output while the PAD initiates an X.28 outgoing call:

```
Router# debug x28
X28 MODE debugging is on
Router# x28

*
03:30:43: X.28 mode session started
03:30:43: X28 escape is exit
03:30:43: Speed for console & vty lines :9600
*call 123456
COM
03:39:04: address ="123456", cud="[none]" 03:39:04: Setting X.3 Parameters for this
call...1:1 2:1 3:126 4:0 5:1 6:2 7:2 8:0 9:0 10:0 11:14 12:1 13:0 14:0 15:0 16:127 17:24
18:18 19:2 20:0 21:0 22:0

Router> exit
CLR CONF

*
*03:40:50: Session ended
* exit

Router#
*03:40:51: Exiting X.28 mode
```

debug xcctsp all

To debug External Call Control TSP information, use the **debug xcctsp all** privileged EXEC command. To turn off debugging, use the **no** form of this command.

debug xcctsp all

no debug xcctsp all

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off external call control debugging:

```
AS5300-TGW# debug xcctsp all
External call control all debugging is on

AS5300-TGW# no debug xcct all
External call control all debugging is off

AS5300-TGW#
```

Related Commands

Command	Description
debug xcctsp error	Enables debugging on external call control errors.
debug xcctsp session	Enables debugging on external call control sessions.

debug xcctsp error

To debug External Call Control TSP error information, use the **debug xcctsp error** privileged EXEC command. To turn off error debugging, use the **no** form of this command.

debug xcctsp error

no debug xcctsp error

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off error-level debugging:

```
AS5300-TGW# debug xcctsp error
External call control error debugging is on

AS5300-TGW# no debug xcctsp error
External call control error debugging is off
```

Related Commands

Command	Description
debug xcctsp all	Enables debugging on all external call control levels.
debug xcctsp session	Enables debugging on external call control sessions.

debug xcctsp session

To debug External Call Control TSP session information, use the **debug xcctsp session** privileged EXEC command. To turn off debugging, use the **no** form of this command.

debug xcctsp session

no debug xcctsp session

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off session-level debugging:

```
AS5300-TGW# debug xcct session
External call control session debugging is on

AS5300-TGW# no debug xcct session
External call control session debugging is off

AS5300-TGW#
```

Related Commands

Command	Description
debug xcctsp all	Enables debugging on external call control levels.
debug xcctsp error	Enables debugging on external call control errors.

debug xns packet

To display information on XNS packet traffic, including the addresses for source, destination, and next hop router of each packet, use the **debug xns packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug xns packet

no debug xns packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

To gain the fullest understanding of XNS routing activity, you should enable **debug xns routing** and **debug xns packet** together.

Examples

The following is sample output from the **debug xns packet** command:

```
Router# debug xns packet

XNS: src=5.0000.0c02.6d04, dst=5.ffff.ffff.ffff, packet sent
XNS: src=1.0000.0c00.440f, dst=1.ffff.ffff.ffff, rcvd. on Ethernet0
XNS: src=1.0000.0c00.440f, dst=1.ffff.ffff.ffff, local processing
```

Table 239 describes significant fields shown in the display.

Table 239 debug xns packet Field Descriptions

Field	Description
XNS:	Indicates that this is an XNS packet.
src = 5.0000.0c02.6d04	Indicates that the source address for this message is 0000.0c02.6d04 on network 5.
dst = 5.ffff.ffff.ffff	Indicates that the destination address for this message is the broadcast address ffff.ffff.ffff on network 5.
packet sent	Indicates that the packet to destination address 5.ffff.ffff.ffff has displayed using the **debug xns packet** command, was queued on the output interface.
rcvd. on Ethernet0	Indicates that the router just received this packet through the Ethernet0 interface.
local processing	Indicates that the router has examined the packet and determined that it must process it, rather than forwarding it.

debug xns routing

To display information on XNS routing transactions, use the **debug xns routing** privileged EXEC command. The **no** form of this command disables debugging output.

debug xns routing

no debug xns routing

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

To gain the fullest understanding of XNS routing activity, enable **debug xns routing** and **debug xns packet** together.

Examples

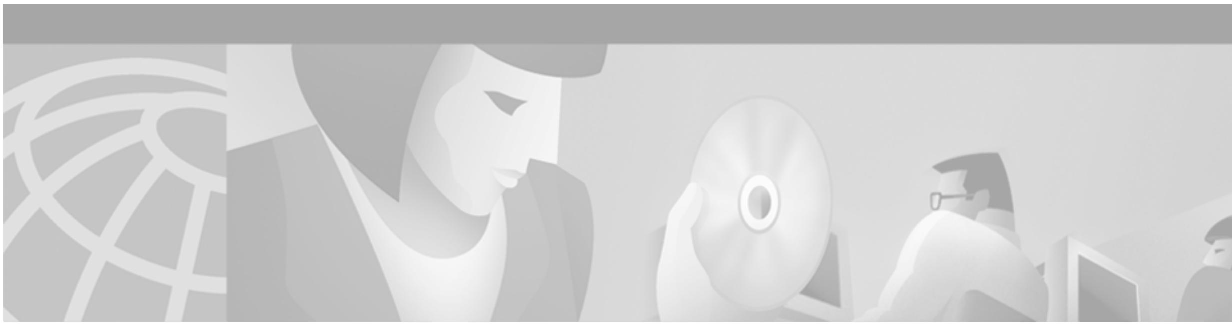
The following is sample output from the **debug xns routing** command:

```
Router# debug xns routing

XNSRIP: sending standard periodic update to 5.ffff.ffff.ffff via Ethernet2
 network 1, hop count 1
 network 2, hop count 2

XNSRIP: got standard update from 1.0000.0c00.440f socket 1 via Ethernet0
 net 2: 1 hops
```

Table 240 describes significant fields shown in the display.

Table 240 debug xns routing Field Descriptions

Field	Description
XNSRIP:	This is an XNS routing packet.
sending standard periodic update	Router indicates that this is a periodic XNS routing information update.
to 5.ffff.ffff.ffff	Destination address is ffff.ffff.ffff on network 5.
via Ethernet2	Name of the output interface.
network 1, hop count 1	Network 1 is one hop away from this router.
got standard update from 1.0000.0c00.440f	Router indicates that it has received an XNS routing information update from address 0000.0c00.440f on network 1.
socket 1	The socket number is a well-known port for XNS. Possible values include • 1—routing information • 2—echo • 3—router error



X.25 Cause and Diagnostic Codes

This appendix covers the X.25 cause and diagnostic codes that can appear in output from the **debug x25 all**, **debug x25 events**, and **debug x25 vc** command documented in the “Debug Commands” chapter. For more information on these codes, see the 1984 ITU-T X.25 Recommendation.

Note The ITU-T carries out the functions of the former Consultative Committee for International Telegraph and Telephone (CCITT).

Note The router reports the decimal value of a cause or diagnostic code, whereas other X.25 equipment may report these codes in hexadecimal notation. For this reason, this appendix lists both the decimal and hexadecimal values of the cause and diagnostic codes.

Table 241 describes the differences between our implementation of certain X.25 network-generated, “international problem” diagnostic fields and the definitions provided in Annex E of ITU-T Recommendation X.25. The Annex E Table E-1/X.25 includes the complete diagnostic field listing.

Table 241 Annex E International Problem Diagnostic Code Differences

Decimal Value	Annex E, Rec. X.25 Diagnostic Description	Cisco Proprietary Definition of Diagnostic Codes
112	International problem	Not used.
113	Remote network problem	Not used.
114	International protocol problem	Not used.
115	International link out of order	Indicates one of the following failures: failed when initializing a switched PVC; in TCP tunneling, failed when initiating or resetting a PVC; or, failed when PAD PVC circuit was initiated or reset.
116	International link busy	Not used.
117	Transit network facility problem	Not used.
118	Remote network facility problem	Not used.

Table 241 Annex E International Problem Diagnostic Code Differences (continued)

Decimal Value	Annex E, Rec. X.25 Diagnostic Description	Cisco Proprietary Definition of Diagnostic Codes
119	International routing problem	Indicates the following failure: in TCP tunneling of X.25 when session is closed by network. In addition to its standard meaning, Cisco routers use this code to signal an abnormal X.25-over-TCP (XOT) condition. This code is used when an X.25 Virtual Circuit connection is initiated using XOT, but the remote XOT peer closed the TCP connection. This commonly occurs when the remote XOT peer could not route the received call.
120	Temporary routing problem	Indicates the following failure: when tunneling X.25 through TCP/IP and the remote network is identified as unreachable. In addition to its standard meaning, Cisco routers use this code to signal an abnormal X.25-over-TCP (XOT) condition. This code is used when an X.25 Virtual Circuit connection cannot be initiated using XOT because the TCP connection fails due to an unreachable remote XOT peer.
121	Unknown called DNIC	Not used.
122	Maintenance action (may apply to maintenance action within a national network	For CMNS, indicates the following: router fails to route the call due to setup or unreachability of destination; when VC is cleared using the **clear x25-vc** EXEC command; when router CLEARs a VC when its idle timer expires.

X.25 Cause Codes

A cause code indicates an event that triggered an X.25 packet. The cause code can only appear in entries for CLEAR REQUEST, REGISTRATION CONFIRMATION, RESET REQUEST, and RESTART packets. Possible values for the cause code can vary, depending on the type of packet. Because the REGISTRATION exchange is not supported, those cause codes are not documented in this section.

Table 242 describes the meanings of cause codes for CLEAR REQUEST packets.

Table 242 Cause Code Descriptions for CLEAR REQUEST Packets

Code (Hex)	Code (Dec)	Description
00	0 (or 128 to 255)	DTE originated
01	1	Number busy
03	3	Invalid facility request
05	5	Network congestion
09	9	Out of order
0B	11	Access barred
0D	13	Not obtainable
11	17	Remote procedure error
13	19	Local procedure error
15	21	RPOA out of order
19	25	Reverse charging not accepted
21	33	Incompatible destination
29	41	Fast select not accepted
39	57	Ship absent

Table 243 describes the meanings of cause codes for RESET REQUEST packets.

Table 243 Cause Code Descriptions for RESET REQUEST Packets

Code (Hex)	Code (Dec)	Description
00	0 (or 128 to 255)	DTE originated
01	1	Out of order
03	3	Remote procedure error
05	5	Local procedure error
07	7	Network congestion
09	9	Remote DTE operational
0F	15	Network operational
11	17	Incompatible destination
1D	29	Network out of order

Table 244 describes the meanings of cause codes for RESTART packets.

Table 244 Cause Code Descriptions for RESTART Packets

Code (Hex)	Code (Dec)	Description
00	0 (or 128 to 255)	DTE restarting
01	1	Local procedure error
03	3	Network congestion
07	7	Network operational
7F	127	Registration/cancellation confirmed

X.25 Diagnostic Codes

The X.25 diag (diagnostic) code provides an additional hint as to what, if anything, went wrong. This code can only appear in entries for CLEAR REQUEST, DIAGNOSTIC, RESET REQUEST, and RESTART packets. Unlike the cause codes, the diag codes do not vary depending upon the type of packet.

Note These diagnostic codes can be produced by any equipment handling a given virtual circuit, and are then propagated through all equipment handling that virtual circuit. Thus, receipt of a diagnostic code may not indicate a problem with the router.

Table 245 describes the meanings of possible diagnostic codes.

Table 245 X.25 Diagnostic Field Code Descriptions

Code (Hex)	Code (Dec)	Description
00	**00**	No additional information
01	01	Invalid P(S)
02	02	Invalid P(R)
10	**16**	**Packet type invalid**
11	17	Packet type invalid for state R1
12	18	Packet type invalid for state R2
13	19	Packet type invalid for state R3
14	20	Packet type invalid for state P1
15	21	Packet type invalid for state P2
16	22	Packet type invalid for state P3
17	23	Packet type invalid for state P4
18	24	Packet type invalid for state P5
19	25	Packet type invalid for state P6
1A	26	Packet type invalid for state P7
1B	27	Packet type invalid for state D1

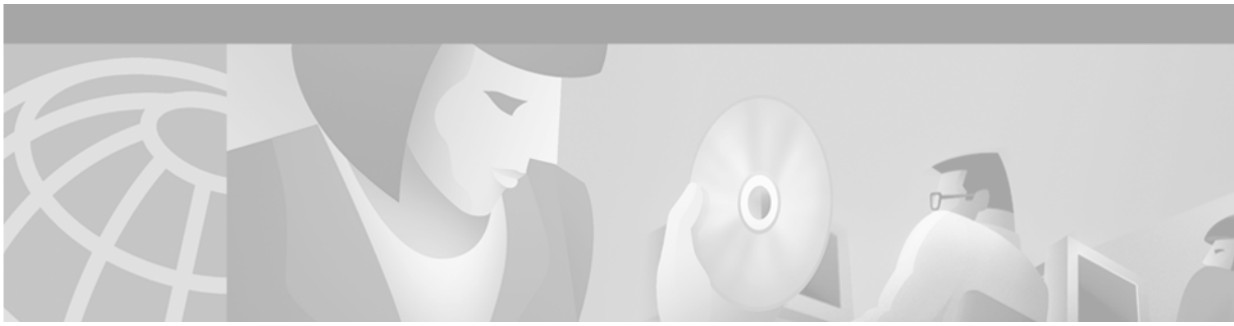
Table 245 X.25 Diagnostic Field Code Descriptions (continued)

Code (Hex)	Code (Dec)	Description
1C	28	Packet type invalid for state D2
1D	29	Packet type invalid for state D3
20	**32**	**Packet not allowed**
21	33	Unidentifiable packet
22	34	Call on one-way logical channel
23	35	Invalid packet type on a permanent virtual circuit
24	36	Packet on unassigned LCN
25	37	Reject not subscribed to
26	38	Packet too short
27	39	Packet too long
28	40	Invalid GFI (General Format Identifier)
29	41	Restart or registration packet with nonzero LCI
2A	42	Packet type not compatible with facility
2B	43	Unauthorized interrupt confirmation
2C	44	Unauthorized interrupt
2D	45	Unauthorized reject
30	**48**	**Timer expired**
31	49	Timer expired for incoming call
32	50	Timer expired for clear indication
33	51	Timer expired for reset indication
34	52	Timer expired for restart indication
35	53	Timer expired for call deflection
40	**64**	**Call setup, clearing, or registration problem**
41	65	Facility code not allowed
42	66	Facility parameter not allowed
43	67	Invalid called address
44	68	Invalid calling address
45	69	Invalid facility length
46	70	Incoming call barred
47	71	No logical channel available
48	72	Call collision
49	73	Duplicate facility requested
4A	74	Nonzero address length
4B	75	Nonzero facility length
4C	76	Facility not provided when expected
4D	77	Invalid ITU-T-specified DTE facility

Table 245 X.25 Diagnostic Field Code Descriptions (continued)

Code (Hex)	Code (Dec)	Description
4E	78	Maximum number of call redirections or deflections exceeded
50	80	Miscellaneous
51	81	Improper cause code for DTE
52	82	Octet not aligned
53	83	Inconsistent Q bit setting
54	84	NUI (Network User Identification) problem
70	**112**	**International problem**
71	113	Remote network problem
72	114	International protocol problem
73	115	International link out of order
74	116	International link busy
75	117	Transit network facility problem
76	118	Remote network facility problem
77	119	International routing problem
78	120	Temporary routing problem
79	121	Unknown called DNIC
7A	122	Maintenance action (**clear x25 vc** command issued)

Diagnostic codes with values of 80 or greater in hexadecimal, or with values of 128 or greater in decimal, are specific to a particular network. To learn the meanings of these codes, contact the administrator for that network



ISDN Switch Types, Codes, and Values

This appendix contains a list of the supported switch types. It also contains the ISDN cause codes, cause values, bearer capability values, and progress description field values that are valid within the debug commands for ISDN.

Note The ITU-T carries out the functions of the former Consultative Committee for International Telegraph and Telephone (CCITT).

Switch Types

Table 246 lists the ISDN switch types supported by the ISDN interface.

Table 246 Supported ISDN Switch Types

Identifier	Description
basic-1tr6	German 1TR6 ISDN switches
basic-5ess	AT&T basic rate switches
basic-dms100	NT DMS-100 basic rate switches
basic-net3	NET3 ISDN and Euro-ISDN switches (UK and others), also called E-DSS1 or DSS1
basic-ni1	National ISDN-1 switches
basic-nwnet3	Norway Net3 switches
basic-nznet3	New Zealand Net3 switches
basic-ts013	Australian TS013 switches
none	No switch defined
ntt	Japanese NTT ISDN switches (ISDN BRI only)
primary-4ess	AT&T 4ESS switch type for the U.S. (ISDN PRI only)
primary-5ess	AT&T 5ESS switch type for the U.S. (ISDN PRI only)
primary-dms100	NT DMS-100 switch type for the U.S. (ISDN PRI only)
primary-net5	NET5 ISDN PRI switches (Europe)
primary-ntt	INS-Net 1500 for Japan (ISDN PRI only)

Table 246 Supported ISDN Switch Types (continued)

Identifier	Description
primary-ts014	Australian TS014 switches (ISDN PRI only)
vn2	French VN2 ISDN switches (ISDN BRI only)
vn3	French VN3 ISDN switches (ISDN BRI only)
vn4	French VN4 ISDN switches (ISDN BRI only)

Cause Code Fields

Table 247 lists the ISDN cause code fields that display in the following format within the debug commands:

```
i=0x y1 y2 z1 z2 [a1 a2]
```

Table 247 ISDN Cause Code Fields

Field	Value—Description
0x	The values that follow are in hexadecimal.
y1	8—ITU-T standard coding.
y2	0—User 1—Private network serving local user 2—Public network serving local user 3—Transit network 4—Public network serving remote user 5—Private network serving remote user 7—International network A—Network beyond internetworking point
z1	Class (the more significant hexadecimal number) of cause value. Refer to Table 248 for detailed information about possible values.
z2	Value (the less significant hexadecimal number) of cause value. Refer to Table 248 for detailed information about possible values.
a1	(Optional) Diagnostic field that is always 8.
a2	(Optional) Diagnostic field that is one of the following values: 0—Unknown 1—Permanent 2—Transient

The following is sample output of this form of the **debug isdn q931** command:

```
Cause i = 0x8790
```

Cause Values

Table 248 lists descriptions of the cause value field of the cause information element. The notes referred to in the Diagnostics column follow the table. For the **debug isdn q931** command output, drop the highest bit of the cause value before using this table. For example, a cause value of 0x90 becomes 0x10.

Table 248 ISDN Cause Values

Decimal Value	Hex Value	Cause	Diagnostics	Explanation
1	01	Unallocated (unassigned) number	Note 10	ISDN number was sent to the switch in the correct format; however, the number is not assigned to any destination equipment.
2	02	No route to specified transit network	Transit network identity (Note 9)	ISDN exchange is asked to route the call through an unrecognized intermediate network.
3	03	No route to destination	Note 10	Call was routed through an intermediate network that does not serve the destination address.
6	06	Channel unacceptable		Service quality of the specified channel is insufficient to accept the connection.
7	07	Call awarded and being delivered in an established channel		User is assigned an incoming call that is being connected to an already-established call channel.
16	10	Normal call clearing	Note 10	Normal call clearing has occurred.
17	11	User busy		Called system acknowledges the connection request but is unable to accept the call because all B channels are in use.
18	12	No user responding		Connection cannot be completed because the destination does not respond to the call.
19	13	No answer from user (user alerted)		Destination responds to the connection request but fails to complete the connection within the prescribed time. The problem is at the remote end of the connection.
21	15	Call rejected	Note 10—User supplied diagnostic (Note 4)	Destination is capable of accepting the call but rejected the call for an unknown reason.

Table 248 ISDN Cause Values (continued)

Decimal Value	Hex Value	Cause	Diagnostics	Explanation
22	16	Number changed		ISDN number used to set up the call is not assigned to any system.
26	1A	Non-selected user clearing		Destination is capable of accepting the call but rejected the call because it was not assigned to the user.
27	1B	Designation out of order		Destination cannot be reached because the interface is not functioning correctly, and a signaling message cannot be delivered. This might be a temporary condition, but it could last for an extended period of time. For example, the remote equipment might be turned off.
28	1C	Invalid number format		Connection could be established because the destination address was presented in an unrecognizable format or because the destination address was incomplete.
29	1D	Facility rejected	Facility identification (Note 1)	Facility requested by the user cannot be provided by the network.
30	1E	Response to STATUS ENQUIRY		Status message was generated in direct response to the prior receipt of a status enquiry message.
31	1F	Normal, unspecified		Reports the occurrence of a normal event when no standard cause applies. No action required.
34	22	No circuit/channel available		Connection cannot be established because no appropriate channel is available to take the call.
38	26	Network out of order		Destination cannot be reached because the network is not functioning correctly, and the condition might last for an extended period of time. An immediate reconnect attempt will probably be unsuccessful.
41	29	Temporary failure		Error occurred because the network is not functioning correctly. The problem will be resolved shortly.
42	2A	Switching equipment congestion		Destination cannot be reached because the network switching equipment is temporarily overloaded.

Table 248 ISDN Cause Values (continued)

Decimal Value	Hex Value	Cause	Diagnostics	Explanation
43	2B	Access information discarded	Discarded information element identifier(s) (Note 5)	Network cannot provide the requested access information.
44	2C	Requested circuit/channel not available		Remote equipment cannot provide the requested channel for an unknown reason. This might be a temporary problem.
47	2F	Resources unavailable, unspecified		Requested channel or service is unavailable for an unknown reason. This might be a temporary problem.
49	31	Quality of service unavailable	Table 247	Requested quality of service cannot be provided by the network. This might be a subscription problem.
50	32	Requested facility not subscribed	Facility identification (Note 1)	Remote equipment supports the requested supplementary service by subscription only.
57	39	Bearer capability not authorized	Note 3	User requested a bearer capability that the network provides, but the user is not authorized to use it. This might be a subscription problem.
58	3A	Bearer capability not presently available	Note 3	Network normally provides the requested bearer capability, but it is unavailable at the present time. This might be due to a temporary network problem or to a subscription problem.
63	3F	Service or option not available, unspecified		Network or remote equipment was unable to provide the requested service option for an unspecified reason. This might be a subscription problem.
65	41	Bearer capability not implemented	Note 3	Network cannot provide the bearer capability requested by the user.
66	42	Channel type not implemented	Channel Type (Note 6)	Network or the destination equipment does not support the requested channel type.
69	45	Requested facility not implemented	Facility Identification (Note 1)	Remote equipment does not support the requested supplementary service.

Table 248 ISDN Cause Values (continued)

Decimal Value	Hex Value	Cause	Diagnostics	Explanation
70	46	Only restricted digital information bearer capability is available		Network is unable to provide unrestricted digital information bearer capability.
79	4F	Service or option not implemented, unspecified		Network or remote equipment is unable to provide the requested service option for an unspecified reason. This might be a subscription problem.
81	51	Invalid call reference value		Remote equipment received a call with a call reference that is not currently in use on the user-network interface.
82	52	Identified channel does not exist	Channel identity	Receiving equipment is requested to use a channel that is not activated on the interface for calls.
83	53	A suspended call exists, but this call identity does not		Network received a call resume request. The call resume request contained a Call Identify information element that indicates that the call identity is being used for a suspended call.
84	54	Call identity in use		Network received a call resume request. The call resume request contained a Call Identify information element that indicates that it is in use for a suspended call.
85	55	No call suspended		Network received a call resume request when there was not a suspended call pending. This might be a transient error that will be resolved by successive call retries.
86	56	Call having the requested call identity has been cleared	Clearing cause	Network received a call resume request. The call resume request contained a Call Identity information element, which once indicated a suspended call. However, the suspended call was cleared either by timeout or by the remote user.
88	58	Incompatible destination	Incompatible parameter (Note 2)	Indicates that an attempt was made to connect to non-ISDN equipment. For example, to an analog line.

Table 248 ISDN Cause Values (continued)

Decimal Value	Hex Value	Cause	Diagnostics	Explanation
91	5B	Invalid transit network selection		ISDN exchange was asked to route the call through an unrecognized intermediate network.
95	5F	Invalid message, unspecified		Invalid message was received, and no standard cause applies. This is usually due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
96	60	Mandatory information element is missing	Information element identifier(s) (Note 5)	Receiving equipment received a message that did not include one of the mandatory information elements. This is usually due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
97	61	Message type non-existent or not implemented	Message type	Receiving equipment received an unrecognized message, either because the message type was invalid or because the message type was valid but not supported. The cause is due to either a problem with the remote configuration or a problem with the local D channel.
98	62	Message not compatible with call state or message type non-existent or not implemented	Message type	Remote equipment received an invalid message, and no standard cause applies. This cause is due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
99	63	Information element non-existent or not implemented	Information element identifier(s) (Notes 5, 7)	Remote equipment received a message that includes information elements, which were not recognized. This is usually due to a D-channel error. If this error occurs systematically, report it to your ISDN service provider.
100	64	Invalid information element contents	Information element identifier(s) (Note 5)	Remote equipment received a message that includes invalid information in the information element. This is usually due to a D-channel error.

Table 248 ISDN Cause Values (continued)

Decimal Value	Hex Value	Cause	Diagnostics	Explanation
101	65	Message not compatible with call state	Message type	Remote equipment received an unexpected message that does not correspond to the current state of the connection. This is usually due to a D-channel error.
102	66	Recovery on timer expires	Timer number (Note 8)	Error-handling (recovery) procedure was initiated by a timer expiry. This is usually a temporary problem.
111	6F	Protocol error, unspecified		Unspecified D-channel error when no other standard cause applies.
127	7F	Internetworking, unspecified		Event occurred, but the network does not provide causes for the action that it takes. The precise problem is unknown.

Note 1: The coding of facility identification is network dependent.

Note 2: Incompatible parameter is composed of incompatible information element identifier.

Note 3: The format of the diagnostic field for causes 39, 3A, and 41 is shown in the ITU-T Q.850 specification, Table 3b/Q.850.

Note 4: User-supplied diagnostic field is encoded according to the user specification, subject to the maximum length of the cause information element. The coding of user-supplied diagnostics should be made in such a way that it does not conflict with the coding described in Table 247.

Note 5: Locking and non-locking shift procedures described in the ITU-T Q.931 specification apply. In principle, information element identifiers are in the same order as the information elements in the received message.

Note 6: The following coding is used:

- Bit 8—extension bit
- Bit 7 through 5—spare
- Bit 4 through 1—according to Table 4-15/Q.931 octet 3.2, channel type in ITU-T Q.931 specification

Note 7: When only locking shift information element is included and no variable length information element identifier follows, it means that the codeset in the locking shift itself is not implemented.

Note 8: The timer number is coded in IA5 characters. The following coding is used in each octet:

- Bit 8—Spare “0”
- Bit 7 through 1—IA5 character

Note 9: The diagnostic field contains the entire transit network selection or network-specific facilities information element, as applicable.

Note 10: See Table 247 for the coding that is used.

Bearer Capability Values

Table 249 lists the ISDN bearer capability values that display in the following format within the debug commands:

- 0x8890 for 64 kbps or
- 0x8890218F for 56 kbps
- 0x8090A2 for Voice call (mu-law)
- 0x9090A2 for Voice call (mu-law)
- 0x8090A3 for Voice call (a-law)
- 0x9090A3 for Voice call (a-law)

Table 249 ISDN Bearer Capability Values

Field	Value—Description
0x	Indication that the values that follow are in hexadecimal
88	ITU-T coding standard; unrestricted digital information
90	Circuit mode, 64 kbps
21	Layer 1, V.110/X.30
8F	Synchronous, no in-band negotiation, 56 kbps
0x8090A2	Voice call (mu-law)
0x9090A2	Voice call (mu-law), 3.1 kHz Audio
0x8090A3	Voice call (a-law)
0x9090A3	Voice call (a-law), 3.1 kHz Audio

Progress Field Values

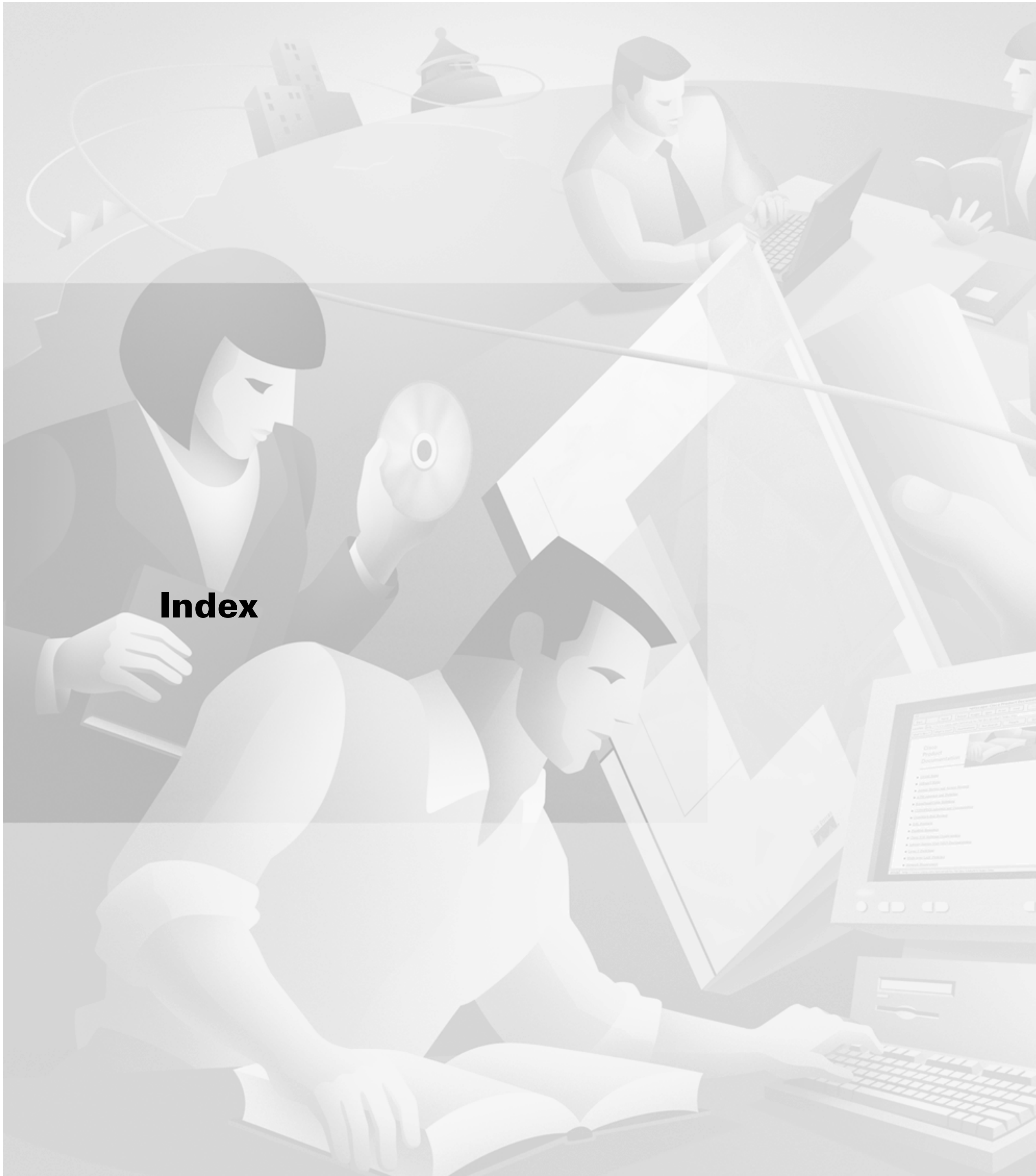
Table 250 lists the values of the Progress description field contained in the ISDN Progress indicator information element.

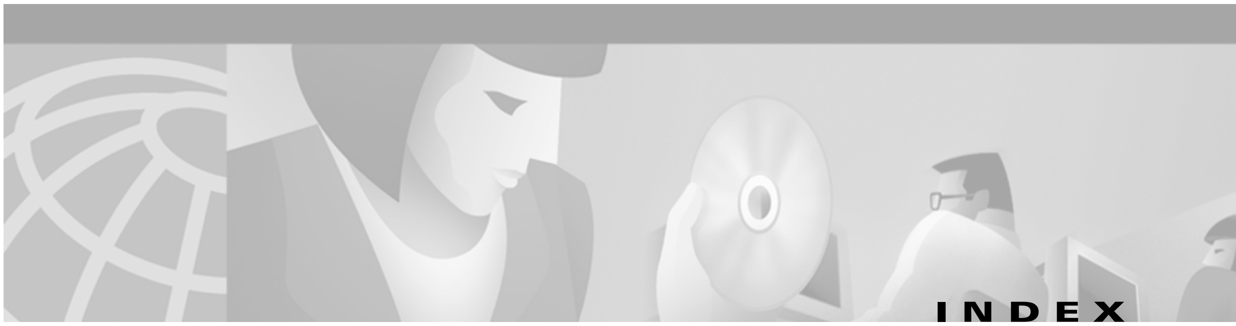
Table 250 ISDN Progress Description Field Values

Bits	Decimal Number	Description
0000001	1	Call is not end-to-end ISDN; further call progress information may be available in-band
0000010	2	Destination address is non-ISDN
0000011	3	Origination address is non-ISDN
0000100	4	Call has returned to the ISDN
0001000	8	In-band information or appropriate pattern now available
0001010	10	Delay in response at destination interface

All other values for the progress description field are reserved.

Index





Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

S

T

U

V

X

Z