Cisco IOS Voice Troubleshooting and Monitoring Guide

Release 12.4

Americas 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: 408 527-0883

Cisco IOS Voice Troubleshooting and Monitoring Guide

About Cisco IOS and Cisco IOS XE Software Documentation

Last updated: August 6, 2008

This document describes the objectives, audience, conventions, and organization used in Cisco IOS and Cisco IOS XE software documentation, collectively referred to in this document as Cisco IOS documentation. Also included are resources for obtaining technical assistance, additional documentation, and other information from Cisco. This document is organized into the following sections:

Documentation Objectives

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

Audience

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

Documentation Conventions

In Cisco IOS documentation, the term *router* may be used to refer to various Cisco products; for example, routers, access servers, and switches. These and other networking devices that support Cisco IOS software are shown interchangeably in examples and are used only for illustrative purposes. An example that shows one product does not necessarily mean that other products are not supported.

This section includes the following topics:

Typographic Conventions

Cisco IOS documentation uses the following typographic conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you 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 a Simple Network Management Protocol (SNMP) community string to *public*, do not use quotation marks around the string; otherwise, the string will include the quotation marks.

Command Syntax Conventions

Cisco IOS documentation uses the following command syntax conventions:

Convention	Description
bold	Bold text indicates commands and keywords that you enter as shown.
italic	Italic text indicates arguments for which you supply values.
[x]	Square brackets enclose an optional keyword or argument.
|	A vertical line, called a pipe, indicates a choice within a set of keywords or arguments.
[x | y]	Square brackets enclosing keywords or arguments separated by a pipe indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a pipe indicate a required choice.
[x {y | z}]	Braces and a pipe within square brackets indicate a required choice within an optional element.

Software Conventions

Cisco IOS uses the following program code conventions:

Convention	Description
`Courier font`	Courier font is used for information that is displayed on a PC or terminal screen.
`Bold Courier font`	Bold Courier font indicates text that the user must enter.
< >	Angle brackets enclose text that is not displayed, such as a password. Angle brackets also are used in contexts in which the italic font style is not supported; for example, ASCII text.
!	An exclamation point at the beginning of a line indicates that the text that follows is a comment, not a line of code. An exclamation point is also displayed by Cisco IOS software for certain processes.
[ ]	Square brackets enclose default responses to system prompts.

Reader Alert Conventions

The Cisco IOS documentation set uses the following conventions for reader alerts:

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 material not covered in the manual.

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

Documentation Organization

This section describes the Cisco IOS documentation set, how it is organized, and how to access it on Cisco.com. Included are lists of configuration guides, command references, and supplementary references and resources that make up the documentation set. The following topics are included:

Cisco IOS Documentation Set

Cisco IOS documentation consists of the following:

- Release notes and caveats provide information about platform, technology, and feature support for a release and describe severity 1 (catastrophic), severity 2 (severe), and severity 3 (moderate) defects in released Cisco IOS code. Review release notes before other documents to learn whether or not updates have been made to a feature.
- Sets of configuration guides and command references organized by technology and published for each standard Cisco IOS release.
  - Configuration guides—Compilations of documents that provide informational and task-oriented descriptions of Cisco IOS features.
  - Command references—Compilations of command pages that provide detailed information about the commands used in the Cisco IOS features and processes that make up the related configuration guides. For each technology, there is a single command reference that covers all Cisco IOS releases and that is updated at each standard release.
- Lists of all the commands in a specific release and all commands that are new, modified, removed, or replaced in the release.
- Command reference book for **debug** commands. Command pages are listed in alphabetical order.
- Reference book for system messages for all Cisco IOS releases.

Cisco IOS Documentation on Cisco.com

The following sections describe the documentation organization and how to access various document types.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

New Features List

The New Features List for each release provides a list of all features in the release with hyperlinks to the feature guides in which they are documented.

Feature Guides

Cisco IOS features are documented in feature guides. Feature guides describe one feature or a group of related features that are supported on many different software releases and platforms. Your Cisco IOS software release or platform may not support all the features documented in a feature guide. See the Feature Information table at the end of the feature guide for information about which features in that guide are supported in your software release.

Configuration Guides

Configuration guides are provided by technology and release and comprise a set of individual feature guides relevant to the release and technology.

Command References

Command reference books describe Cisco IOS commands that are supported in many different software releases and on many different platforms. The books are provided by technology. For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases*, at http://www.cisco.com/en/US/docs/ios/mcl/all_release/all_mcl.html.

Cisco IOS Supplementary Documents and Resources

Supplementary documents and resources are listed in Table 2 on page xi.

Configuration Guides, Command References, and Supplementary Resources

Table 1 lists, in alphabetical order, Cisco IOS and Cisco IOS XE software configuration guides and command references, including brief descriptions of the contents of the documents. The Cisco IOS command references are comprehensive, meaning that they include commands for both Cisco IOS software and Cisco IOS XE software, for all releases. The configuration guides and command references support many different software releases and platforms. Your Cisco IOS software release or platform may not support all these technologies.

For additional information about configuring and operating specific networking devices, go to the Product Support area of Cisco.com at http://www.cisco.com/web/psa/products/index.html.

Table 2 lists documents and resources that supplement the Cisco IOS software configuration guides and command references. These supplementary resources include release notes and caveats; master command lists; new, modified, removed, and replaced command lists; system messages; and the debug command reference.

Table 1 Cisco IOS and Cisco IOS XE Configuration Guides and Command References

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
Cisco IOS AppleTalk Configuration Guide *Cisco IOS XE AppleTalk Configuration Guide* *Cisco IOS AppleTalk Command Reference*	AppleTalk protocol.
Cisco IOS Asynchronous Transfer Mode Configuration Guide *Cisco IOS Asynchronous Transfer Mode Command Reference*	LAN ATM, multiprotocol over ATM (MPoA), and WAN ATM.

Table 1 Cisco IOS and Cisco IOS XE Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
Cisco IOS Bridging and IBM Networking Configuration Guide *Cisco IOS Bridging Command Reference* *Cisco IOS IBM Networking Command Reference*	• Transparent and source-route transparent (SRT) bridging, source-route bridging (SRB), Token Ring Inter-Switch Link (TRISL), and token ring route switch module (TRRSM). • Data-link switching plus (DLSw+), serial tunnel (STUN), block serial tunnel (BSTUN); logical link control, type 2 (LLC2), synchronous data link control (SDLC); IBM Network Media Translation, including Synchronous Data Logical Link Control (SDLLC) and qualified LLC (QLLC); downstream physical unit (DSPU), Systems Network Architecture (SNA) service point, SNA frame relay access, advanced peer-to-peer networking (APPN), native client interface architecture (NCIA) client/server topologies, and IBM Channel Attach.
Cisco IOS Broadband and DSL Configuration Guide *Cisco IOS XE Broadband and DSL Configuration Guide* *Cisco IOS Broadband and DSL Command Reference*	Point-to-Point Protocol (PPP) over ATM (PPPoA) and PPP over Ethernet (PPPoE).
Cisco IOS Carrier Ethernet Configuration Guide *Cisco IOS Carrier Ethernet Command Reference*	Connectivity fault management (CFM), Ethernet Local Management Interface (ELMI), IEEE 802.3ad link bundling, Link Layer Discovery Protocol (LLDP), media endpoint discovery (MED), and operations, administration, and maintenance (OAM).
Cisco IOS Configuration Fundamentals Configuration Guide *Cisco IOS XE Configuration Fundamentals Configuration Guide* *Cisco IOS Configuration Fundamentals Command Reference*	Autoinstall, Setup, Cisco IOS command-line interface (CLI), Cisco IOS file system (IFS), Cisco IOS web browser user interface (UI), basic file transfer services, and file management.
Cisco IOS DECnet Configuration Guide *Cisco IOS XE DECnet Configuration Guide* *Cisco IOS DECnet Command Reference*	DECnet protocol.
Cisco IOS Dial Technologies Configuration Guide *Cisco IOS XE Dial Technologies Configuration Guide* *Cisco IOS Dial Technologies Command Reference*	Asynchronous communications, dial backup, dialer technology, dial-in terminal services and AppleTalk remote access (ARA), large scale dialout, dial-on-demand routing, dialout, modem and resource pooling, ISDN, multilink PPP (MLP), PPP, virtual private dialup network (VPDN).
Cisco IOS Flexible NetFlow Configuration Guide *Cisco IOS Flexible NetFlow Command Reference*	Flexible NetFlow.

Table 1 Cisco IOS and Cisco IOS XE Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
Cisco IOS H.323 Configuration Guide	Gatekeeper enhancements for managed voice services, Gatekeeper Transaction Message Protocol, gateway codec order preservation and shutdown control, H.323 dual tone multifrequency relay, H.323 version 2 enhancements, Network Address Translation (NAT) support of H.323 v2 Registration, Admission, and Status (RAS) protocol, tokenless call authorization, and VoIP gateway trunk and carrier-based routing.
Cisco IOS High Availability Configuration Guide *Cisco IOS XE High Availability Configuration Guide* *Cisco IOS High Availability Command Reference*	A variety of High Availability (HA) features and technologies that are available for different network segments (from enterprise access to service provider core) to facilitate creation of end-to-end highly available networks. Cisco IOS HA features and technologies can be categorized in three key areas: system-level resiliency, network-level resiliency, and embedded management for resiliency.
Cisco IOS Integrated Session Border Controller Command Reference	A VoIP-enabled device that is deployed at the edge of networks. An SBC is a toolkit of functions, such as signaling interworking, network hiding, security, and quality of service (QoS).
Cisco IOS Intelligent Service Gateway Configuration Guide *Cisco IOS Intelligent Service Gateway Command Reference*	Subscriber identification, service and policy determination, session creation, session policy enforcement, session life-cycle management, accounting for access and service usage, session state monitoring.
Cisco IOS Interface and Hardware Component Configuration Guide *Cisco IOS XE Interface and Hardware Component Configuration Guide* *Cisco IOS Interface and Hardware Component Command Reference*	LAN interfaces, logical interfaces, serial interfaces, virtual interfaces, and interface configuration.
Cisco IOS IP Addressing Services Configuration Guide *Cisco IOS XE Addressing Services Configuration Guide* *Cisco IOS IP Addressing Services Command Reference*	Address Resolution Protocol (ARP), Network Address Translation (NAT), Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and Next Hop Address Resolution Protocol (NHRP).
Cisco IOS IP Application Services Configuration Guide *Cisco IOS XE IP Application Services Configuration Guide* *Cisco IOS IP Application Services Command Reference*	Enhanced Object Tracking (EOT), Gateway Load Balancing Protocol (GLBP), Hot Standby Router Protocol (HSRP), IP Services, Server Load Balancing (SLB), Stream Control Transmission Protocol (SCTP), TCP, Web Cache Communication Protocol (WCCP), User Datagram Protocol (UDP), and Virtual Router Redundancy Protocol (VRRP).
Cisco IOS IP Mobility Configuration Guide *Cisco IOS IP Mobility Command Reference*	Mobile ad hoc networks (MANet) and Cisco mobile networks.
Cisco IOS IP Multicast Configuration Guide *Cisco IOS XE IP Multicast Configuration Guide* *Cisco IOS IP Multicast Command Reference*	Protocol Independent Multicast (PIM) sparse mode (PIM-SM), bidirectional PIM (bidir-PIM), Source Specific Multicast (SSM), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol (IGMP), and Multicast VPN (MVPN).

Table 1 Cisco IOS and Cisco IOS XE Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
Cisco IOS IP Routing Protocols Configuration Guide *Cisco IOS XE IP Routing Protocols Configuration Guide* *Cisco IOS IP Routing Protocols Command Reference*	Border Gateway Protocol (BGP), multiprotocol BGP, multiprotocol BGP extensions for IP multicast, bidirectional forwarding detection (BFD), Enhanced Interior Gateway Routing Protocol (EIGRP), Interior Gateway Routing Protocol (IGRP), Intermediate System-to-Intermediate System (IS-IS), on-demand routing (ODR), Open Shortest Path First (OSPF), and Routing Information Protocol (RIP).
Cisco IOS IP SLAs Configuration Guide *Cisco IOS XE IP SLAs Configuration Guide* *Cisco IOS IP SLAs Command Reference*	Cisco IOS IP Service Level Agreements (IP SLAs).
Cisco IOS IP Switching Configuration Guide *Cisco IOS XE IP Switching Configuration Guide* *Cisco IOS IP Switching Command Reference*	Cisco Express Forwarding, fast switching, and Multicast Distributed Switching (MDS).
Cisco IOS IPv6 Configuration Guide *Cisco IOS XE IPv6 Configuration Guide* *Cisco IOS IPv6 Command Reference*	For IPv6 features, protocols, and technologies, go to the IPv6 "Start Here" document at the following URL: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-roadmap.html
Cisco IOS ISO CLNS Configuration Guide *Cisco IOS XE ISO CLNS Configuration Guide* *Cisco IOS ISO CLNS Command Reference*	ISO connectionless network service (CLNS).
Cisco IOS LAN Switching Configuration Guide *Cisco IOS XE LAN Switching Configuration Guide* *Cisco IOS LAN Switching Command Reference*	VLANs, Inter-Switch Link (ISL) encapsulation, IEEE 802.10 encapsulation, IEEE 802.1Q encapsulation, and multilayer switching (MLS).
Cisco IOS Mobile Wireless Gateway GPRS Support Node Configuration Guide *Cisco IOS Mobile Wireless Gateway GPRS Support Node Command Reference*	Cisco IOS Gateway GPRS Support Node (GGSN) in a 2.5-generation general packet radio service (GPRS) and 3-generation universal mobile telecommunication system (UMTS) network.
Cisco IOS Mobile Wireless Home Agent Configuration Guide *Cisco IOS Mobile Wireless Home Agent Command Reference*	Cisco Mobile Wireless Home Agent, an anchor point for mobile terminals for which mobile IP or proxy mobile IP services are provided.
Cisco IOS Mobile Wireless Packet Data Serving Node Configuration Guide *Cisco IOS Mobile Wireless Packet Data Serving Node Command Reference*	Cisco Packet Data Serving Node (PDSN), a wireless gateway that is between the mobile infrastructure and standard IP networks and that enables packet data services in a code division multiple access (CDMA) environment.
Cisco IOS Mobile Wireless Radio Access Networking Configuration Guide *Cisco IOS Mobile Wireless Radio Access Networking Command Reference*	Cisco IOS radio access network products.

Table 1 Cisco IOS and Cisco IOS XE Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
Cisco IOS Multiprotocol Label Switching Configuration Guide *Cisco IOS XE Multiprotocol Label Switching Configuration Guide* *Cisco IOS Multiprotocol Label Switching Command Reference*	MPLS Label Distribution Protocol (LDP), MPLS Layer 2 VPNs, MPLS Layer 3 VPNs, MPLS Traffic Engineering (TE), and MPLS Embedded Management (EM) and MIBs.
Cisco IOS Multi-Topology Routing Configuration Guide *Cisco IOS Multi-Topology Routing Command Reference*	Unicast and multicast topology configurations, traffic classification, routing protocol support, and network management support.
Cisco IOS NetFlow Configuration Guide *Cisco IOS XE NetFlow Configuration Guide* *Cisco IOS NetFlow Command Reference*	Network traffic data analysis, aggregation caches, export features.
Cisco IOS Network Management Configuration Guide *Cisco IOS XE Network Management Configuration Guide* *Cisco IOS Network Management Command Reference*	Basic system management; system monitoring and logging; troubleshooting, logging, and fault management; Cisco Discovery Protocol; Cisco IOS Scripting with Tool Control Language (Tcl); Cisco networking services (CNS); DistributedDirector; Embedded Event Manager (EEM); Embedded Resource Manager (ERM); Embedded Syslog Manager (ESM); HTTP; Remote Monitoring (RMON); SNMP; and VPN Device Manager Client for Cisco IOS Software (XSM Configuration).
Cisco IOS Novell IPX Configuration Guide *Cisco IOS XE Novell IPX Configuration Guide* *Cisco IOS Novell IPX Command Reference*	Novell Internetwork Packet Exchange (IPX) protocol.
Cisco IOS Optimized Edge Routing Configuration Guide *Cisco IOS Optimized Edge Routing Command Reference*	Optimized edge routing (OER) monitoring, policy configuration, routing control, logging and reporting, and VPN IPsec/generic routing encapsulation (GRE) tunnel interface optimization.
Cisco IOS Quality of Service Solutions Configuration Guide *Cisco IOS XE Quality of Service Solutions Configuration Guide* *Cisco IOS Quality of Service Solutions Command Reference*	Class-based weighted fair queuing (CBWFQ), custom queuing, distributed traffic shaping (DTS), generic traffic shaping (GTS), IP- to-ATM class of service (CoS), low latency queuing (LLQ), modular QoS CLI (MQC), Network-Based Application Recognition (NBAR), priority queuing, Security Device Manager (SDM), Multilink PPP (MLPPP) for QoS, header compression, AutoQoS, QoS features for voice, Resource Reservation Protocol (RSVP), weighted fair queuing (WFQ), and weighted random early detection (WRED).
Cisco IOS Security Configuration Guide *Cisco IOS XE Security Configuration Guide* *Cisco IOS Security Command Reference*	Access control lists (ACLs), authentication, authorization, and accounting (AAA), firewalls, IP security and encryption, neighbor router authentication, network access security, network data encryption with router authentication, public key infrastructure (PKI), RADIUS, TACACS+, terminal access security, and traffic filters.

Table 1 Cisco IOS and Cisco IOS XE Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
Cisco IOS Service Selection Gateway Configuration Guide *Cisco IOS Service Selection Gateway Command Reference*	Subscriber authentication, service access, and accounting.
Cisco IOS Software Activation Configuration Guide *Cisco IOS Software Activation Command Reference*	An orchestrated collection of processes and components to activate Cisco IOS software feature sets by obtaining and validating Cisco software licenses.
Cisco IOS Software Modularity Installation and Configuration Guide *Cisco IOS Software Modularity Command Reference*	Installation and basic configuration of software modularity images, including installations on single and dual route processors, installation rollbacks, software modularity binding, software modularity processes and patches.
Cisco IOS Terminal Services Configuration Guide *Cisco IOS Terminal Services Command Reference* *Cisco IOS XE Terminal Services Command Reference*	DEC, local-area transport (LAT), and X.25 packet assembler/disassembler (PAD).
Cisco IOS Virtual Switch Command Reference	Virtual switch redundancy, high availability, and packet handling; converting between standalone and virtual switch modes; virtual switch link (VSL); Virtual Switch Link Protocol (VSLP). **Note** For information about virtual switch configuration, refer to the product-specific software configuration information for the Cisco Catalyst 6500 series switch or for the Metro Ethernet 6500 series switch.
Cisco IOS Voice Configuration Library *Cisco IOS Voice Command Reference*	Cisco IOS support for voice call control protocols, interoperability, physical and virtual interface management, and troubleshooting. The library includes documentation for IP telephony applications.
Cisco IOS VPDN Configuration Guide *Cisco IOS XE VPDN Configuration Guide* *Cisco IOS VPDN Command Reference*	Layer 2 Tunneling Protocol (L2TP) dial-out load balancing and redundancy, L2TP extended failover, L2TP security VPDN, multihop by Dialed Number Identification Service (DNIS), timer and retry enhancements for L2TP and Layer 2 Forwarding (L2F), RADIUS Attribute 82: tunnel assignment ID, shell-based authentication of VPDN users, tunnel authentication via RADIUS on tunnel terminator.
Cisco IOS Wide-Area Networking Configuration Guide *Cisco IOS XE Wide-Area Networking Configuration Guide* *Cisco IOS Wide-Area Networking Command Reference*	Frame Relay, Layer 2 Tunneling Protocol Version 3 (L2TPv3), Link Access Procedure, Balanced (LAPB), Switched Multimegabit Data Service (SMDS), and X.25.
Cisco IOS Wireless LAN Configuration Guide *Cisco IOS Wireless LAN Command Reference*	Broadcast key rotation, IEEE 802.11x support, IEEE 802.1x authenticator, IEEE 802.1x local authentication service for Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling (EAP-FAST), Multiple Basic Service Set ID (BSSID), Wi-Fi Multimedia (WMM) required elements, and Wi-Fi Protected Access (WPA).

Table 2 Cisco IOS Supplementary Documents and Resources

Document Title	Description
Cisco IOS Master Command List, All Releases	Alphabetical list of all the commands documented in all Cisco IOS releases.
Cisco IOS New, Modified, Removed, and Replaced Commands	List of all the new, modified, removed, and replaced commands for a Cisco IOS release.
Cisco IOS Software System Messages	List of Cisco IOS system messages and descriptions. System messages may indicate problems with your system; be informational only; or may help diagnose problems with communications lines, internal hardware, or the system software.
Cisco IOS Debug Command Reference	Alphabetical list of **debug** commands including brief descriptions of use, command syntax, and usage guidelines.
Release Notes and Caveats	Information about new and changed features, system requirements, and other useful information about specific software releases; information about defects in specific Cisco IOS software releases.
MIBs	Files used for network monitoring. To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator at the following URL: http://www.cisco.com/go/mibs
RFCs	Standards documents maintained by the Internet Engineering Task Force (IETF) that Cisco IOS documentation references where applicable. The full text of referenced RFCs may be obtained at the following URL: http://www.rfc-editor.org/

Additional Resources and Documentation Feedback

What's New in Cisco Product Documentation is published monthly and describes all new and revised Cisco technical documentation. The *What's New in Cisco Product Documentation* publication also provides information about obtaining the following resources:

- Technical documentation
- Cisco product security overview
- Product alerts and field notices
- Technical assistance

Cisco IOS technical documentation includes embedded feedback forms where you can rate documents and provide suggestions for improvement. Your feedback helps us improve our documentation.

Using the Command-Line Interface in Cisco IOS and Cisco IOS XE Software

Last updated: August 6, 2008

This document provides basic information about the command-line interface (CLI) in Cisco IOS and Cisco IOS XE software and how you can use some of the CLI features. This document contains the following sections:

For more information about using the CLI, see the "Using the Cisco IOS Command-Line Interface" section of the *Cisco IOS Configuration Fundamentals Configuration Guide*.

For information about the software documentation set, see the "About Cisco IOS and Cisco IOS XE Software Documentation" document.

Initially Configuring a Device

Initially configuring a device varies by platform. For information about performing an initial configuration, see the hardware installation documentation that is provided with the original packaging of the product or go to the Product Support area of Cisco.com at http://www.cisco.com/web/psa/products/index.html.

After you have performed the initial configuration and connected the device to your network, you can configure the device by using the console port or a remote access method, such as Telnet or Secure Shell (SSH), to access the CLI or by using the configuration method provided on the device, such as Security Device Manager.

Changing the Default Settings for a Console or AUX Port

There are only two changes that you can make to a console port and an AUX port:

- Change the port speed with the **config-register 0x** command. Changing the port speed is not recommended. The well-known default speed is 9600.
- Change the behavior of the port; for example, by adding a password or changing the timeout value.

Note The AUX port on the Route Processor (RP) installed in a Cisco ASR1000 series router does not serve any useful customer purpose and should be accessed only under the advisement of a customer support representative.

Using the CLI

This section describes the following topics:

Understanding Command Modes

The CLI command mode structure is hierarchical, and each mode supports a set of specific commands. This section describes the most common of the many modes that exist.

Table 1 lists common command modes with associated CLI prompts, access and exit methods, and a brief description of how each mode is used.

Table 1 CLI Command Modes

Command Mode	Access Method	Prompt	Exit Method	Mode Usage
User EXEC	Log in.	`Router>`	Issue the **logout** or **exit** command.	• Change terminal settings. • Perform basic tests. • Display device status.
Privileged EXEC	From user EXEC mode, issue the **enable** command.	`Router#`	Issue the **disable** command or the **exit** command to return to user EXEC mode.	• Issue **show** and **debug** commands. • Copy images to the device. • Reload the device. • Manage device configuration files. • Manage device file systems.
Global configuration	From privileged EXEC mode, issue the **configure terminal** command.	`Router(config)#`	Issue the **exit** command or the **end** command to return to privileged EXEC mode.	Configure the device.
Interface configuration	From global configuration mode, issue the **interface** command.	`Router(config-if)#`	Issue the **exit** command to return to global configuration mode or the **end** command to return to privileged EXEC mode.	Configure individual interfaces.
Line configuration	From global configuration mode, issue the **line vty** or **line console** command.	`Router(config-line)#`	Issue the **exit** command to return to global configuration mode or the **end** command to return to privileged EXEC mode.	Configure individual terminal lines.

Table 1 CLI Command Modes (continued)

Command Mode	Access Method	Prompt	Exit Method	Mode Usage
ROM monitor	From privileged EXEC mode, issue the **reload** command. Press the **Break** key during the first 60 seconds while the system is booting.	`rommon # >` The # symbol represents the line number and increments at each prompt.	Issue the **continue** command.	• Run as the default operating mode when a valid image cannot be loaded. • Access the fall-back procedure for loading an image when the device lacks a valid image and cannot be booted. • Perform password recovery when a CTRL-Break sequence is issued within 60 seconds of a power-on or reload event.
Diagnostic (available only on the Cisco ASR1000 series router)	The router boots or enters diagnostic mode in the following scenarios. When a Cisco IOS process or processes fail, in most scenarios the router will reload. • A user-configured access policy was configured using the **transport-map** command, which directed the user into diagnostic mode. • The router was accessed using an RP auxiliary port. • A break signal (**Ctrl-C**, **Ctrl-Shift-6**, or the **send break** command) was entered, and the router was configured to enter diagnostic mode when the break signal was received.	`Router(diag)#`	If a Cisco IOS process failure is the reason for entering diagnostic mode, the failure must be resolved and the router must be rebooted to exit diagnostic mode. If the router is in diagnostic mode because of a transport-map configuration, access the router through another port or using a method that is configured to connect to the Cisco IOS CLI. If the RP auxiliary port was used to access the router, use another port for access. Accessing the router through the auxiliary port is not useful for customer purposes.	• Inspect various states on the router, including the Cisco IOS state. • Replace or roll back the configuration. • Provide methods of restarting the Cisco IOS software or other processes. • Reboot hardware, such as the entire router, an RP, an ESP, a SIP, a SPA, or possibly other hardware components. • Transfer files into or off of the router using remote access methods such as FTP, TFTP, and SCP.

EXEC commands are not saved when the software reboots. Commands that you issue in a configuration mode can be saved to the startup configuration. If you save the running configuration to the startup configuration, these commands will execute when the software is rebooted. Global configuration mode is the highest level of configuration mode. From global configuration mode, you can enter a variety of other configuration modes, including protocol-specific modes.

ROM monitor mode is a separate mode that is used when the 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. Use the question symbol (?) to view the commands that you can use while the device is in ROM monitor mode.

```
rommon 1 > ?
alias               set and display aliases command
boot                boot up an external process
confreg             configuration register utility
cont                continue executing a downloaded image
context             display the context of a loaded image
cookie              display contents of cookie PROM in hex
.
.
.
rommon 2 >
```

The following example shows how the command prompt changes to indicate a different command mode:

```
Router> enable
Router# configure terminal
Router(config)# interface ethernet 1/1
Router(config-if)# ethernet
Router(config-line)# exit
Router(config)# end
Router#
```

Note A keyboard alternative to the **end** command is Ctrl-Z.

Using the Interactive Help Feature

The CLI includes an interactive Help feature. Table 2 describes how to use the Help feature.

Table 2 CLI Interactive Help Commands

Command	Purpose
help	Provides a brief description of the help feature in any command mode.
?	Lists all commands available for a particular command mode.
*partial command***?**	Provides a list of commands that begin with the character string (no space between the command and the question mark).
partial command<**Tab**>	Completes a partial command name (no space between the command and <Tab>).
command **?**	Lists the keywords, arguments, or both associated with the command (space between the command and the question mark).
command keyword **?**	Lists the arguments that are associated with the keyword (space between the keyword and the question mark).

The following examples show how to use the help commands:

help

```
Router> help

Help may be requested at any point in a command by entering a question mark '?'.  If
nothing matches, the help list will be empty and you must backup until entering a '?'
shows the available options.

Two styles of help are provided:

1. Full help is available when you are ready to enter a command argument (e.g. 'show ?')
and describes each possible argument.

2. Partial help is provided when an abbreviated argument is entered and you want to know
what arguments match the input (e.g. 'show pr?'.)
```

?

```
Router# ?
Exec commands:
  access-enable        Create a temporary access-List entry
  access-profile       Apply user-profile to interface
  access-template      Create a temporary access-List entry
  alps                 ALPS exec commands
  archive              manage archive files
<snip>
```

***partial command*?**

```
Router(config)# zo?
zone  zone-pair
```

***partial command*<Tab>**

```
Router(config)# we<Tab> webvpn
```

***command* ?**

```
Router(config-if)# pppoe ?
  enable        Enable pppoe
  max-sessions  Maximum PPPOE sessions
```

***command keyword* ?**

```
Router(config-if)# pppoe enable ?
  group  attach a BBA group
  <cr>
```

Understanding Command Syntax

Command syntax is the format in which a command should be entered in the CLI. Commands include the name of the command, keywords, and arguments. Keywords are alphanumeric strings that are used literally. Arguments are placeholders for values that a user must supply. Keywords and arguments may be required or optional.

Specific conventions convey information about syntax and command elements. Table 3 describes these conventions.

Table 3 CLI Syntax Conventions

Symbol/Text	Function	Notes
< > (angle brackets)	Indicate that the option is an argument.	Sometimes arguments are displayed without angle brackets.
A.B.C.D.	Indicates that you must enter a dotted decimal IP address.	Angle brackets (< >) are not always used to indicate that an IP address is an argument.
WORD (all capital letters)	Indicates that you must enter one word.	Angle brackets (< >) are not always used to indicate that a WORD is an argument.
LINE (all capital letters)	Indicates that you must enter more than one word.	Angle brackets (< >) are not always used to indicate that a LINE is an argument.
<cr> (carriage return)	Indicates the end of the list of available keywords and arguments, and also indicates when keywords and arguments are optional. When <cr> is the only option, you have reached the end of the branch or the end of the command if the command has only one branch.	—

The following examples show syntax conventions:

```
Router(config)# ethernet cfm domain ?
  WORD  domain name
Router(config)# ethernet cfm domain dname ?
  level
Router(config)# ethernet cfm domain dname level ?
  <0-7>  maintenance level number
Router(config)# ethernet cfm domain dname level 7 ?
  <cr>
Router(config)# snmp-server file-transfer access-group 10 ?
  protocol  protocol options
  <cr>
Router(config)# logging host ?
  Hostname or A.B.C.D  IP address of the syslog server
  ipv6                 Configure IPv6 syslog server
Router(config)# snmp-server file-transfer access-group 10 ?
  protocol  protocol options
  <cr>
```

Understanding Enable and Enable Secret Passwords

Some privileged EXEC commands are used for actions that impact the system, and it is recommended that you set a password for these commands to prevent unauthorized use. Two types of passwords, enable (not encrypted) and enable secret (encrypted), can be set. The following commands set these passwords and are issued in global configuration mode:

- **enable** *password*
- **enable secret** *password*

Using an enable secret password is recommended because it is encrypted and more secure than the enable password. When you use an enable secret password, text is encrypted (unreadable) before it is written to the config.text file. When you use an enable password, the text is written as entered (readable) to the config.text file.

Each type of password is case sensitive, can contain from 1 to 25 uppercase and lowercase alphanumeric characters, and can start with a number. Spaces are also valid password characters; for example, "two words" is a valid password. Leading spaces are ignored, but trailing spaces are recognized.

Note Both password commands have numeric keywords that are single integer values. If you choose a number for the first character of your password followed by a space, the system will read the number as if it were the numeric keyword and not as part of your password.

When both passwords are set, the enable secret password takes precedence over the enable password.

To remove a password, use the **no** form of the commands: **no enable** *password* or **no enable secret** *password*.

For more information about password recovery procedures for Cisco products, see http://www.cisco.com/en/US/products/sw/iosswrel/ps1831/products_tech_note09186a00801746e6.shtml.

Using the Command History Feature

The CLI command history feature saves the commands you enter during a session in a command history buffer. The default number of commands saved is 10, but the number is configurable within the range of 0 to 256. This command history feature is particularly useful for recalling long or complex commands.

To change the number of commands saved in the history buffer for a terminal session, issue the **terminal history size** command:

```
Router# terminal history size num
```

A command history buffer is also available in line configuration mode with the same default and configuration options. To set the command history buffer size for a terminal session in line configuration mode, issue the **history** command:

```
Router(config-line)# history [size num]
```

To recall commands from the history buffer, use the following methods:

- Press Ctrl-P or the up arrow key—Recalls commands beginning with the most recent command. Repeat the key sequence to recall successively older commands.

- Press Ctrl-N or the down arrow key—Recalls the most recent commands in the history buffer after they have been recalled using Ctrl-P or the up arrow key. Repeat the key sequence to recall successively more recent commands.

Note The arrow keys function only on ANSI-compatible terminals such as the VT100.

- Issue the **show history** command in user EXEC or privileged EXEC mode—Lists the most recent commands that you entered. The number of commands that are displayed is determined by the setting of the **terminal history size** and **history** commands.

  The CLI command history feature is enabled by default. To disable this feature for a terminal session, issue the **terminal no history** command in user EXEC or privileged EXEC mode or the **no history** command in line configuration mode.

Abbreviating Commands

Typing a complete command name is not always required for the command to execute. The CLI recognizes an abbreviated command when the abbreviation contains enough characters to uniquely identify the command. For example, the **show version** command can be abbreviated as **sh ver**. It cannot be abbreviated as **s ver** because **s** could mean **show**, **set**, or **systat**. The **sh v** abbreviation also is not valid because the **show** command has **vrrp** as a keyword in addition to **version**. (Command and keyword examples from Cisco IOS Release 12.4(13)T.)

Using Aliases for CLI Commands

To save time and the repetition of entering the same command multiple times, you can use a command alias. An alias can be configured to do anything that can be done at the command line, but an alias cannot move between modes, type in passwords, or perform any interactive functions.

Table 4 shows the default command aliases.

Table 4 Default Command Aliases

Command Alias	Original Command
h	help
lo	logout
p	ping
s	show
u or **un**	undebug
w	where

To create a command alias, issue the **alias** command in global configuration mode. The syntax of the command is **alias** *mode command-alias original-command*. Following are some examples:

- Router(config)# **alias exec prt partition**—privileged EXEC mode
- Router(config)# **alias configure sb source-bridge**—global configuration mode
- Router(config)# **alias interface rl rate-limit**—interface configuration mode

To view both default and user-created aliases, issue the **show alias** command.

For more information about the **alias** command, see http://www.cisco.com/en/US/docs/ios/fundamentals/command/reference/cf_book.html.

Using the no and default Forms of Commands

Most configuration commands have a **no** form that is used to reset a command to its default value or disable a feature or function. For example, the **ip routing** command is enabled by default. To disable this command, you would issue the **no ip routing** command. To re-enable IP routing, you would issue the **ip routing** command.

Configuration commands may also have a **default** form, which returns the command settings to their default values. For commands that are disabled by default, using the **default** form has the same effect as using the **no** form of the command. For commands that are enabled by default and have default settings, the **default** form enables the command and returns the settings to their default values.

The **no** and **default** forms of commands are described in the command pages of command references.

Using the debug Command

A **debug** command produces extensive output that helps you troubleshoot problems in your network. These commands are available for many features and functions within Cisco IOS and Cisco IOS XE software. Some **debug** commands are **debug all**, **debug aaa accounting**, and **debug mpls packets**. To use **debug** commands during a Telnet session with a device, you must first enter the **terminal monitor** command. To turn off debugging completely, you must enter the **undebug all** command.

For more information about **debug** commands, see the *Cisco IOS Debug Command Reference* at http://www.cisco.com/en/US/docs/ios/debug/command/reference/db_book.html.

Caution Debugging is a high priority and high CPU utilization process that can render your device unusable. Use **debug** commands only to troubleshoot specific problems. The best times to run debugging are during periods of low network traffic and when few users are interacting with the network. Debugging during these periods decreases the likelihood that the **debug** command processing overhead will affect network performance or user access or response times.

Filtering Output Using Output Modifiers

Many commands produce lengthy output that may use several screens to display. Using output modifiers, you can filter this output to show only the information that you want to see.

Three output modifiers are available and are described as follows:

- **begin** *regular expression*—Displays the first line in which a match of the regular expression is found and all lines that follow.
- **include** *regular expression*—Displays all lines in which a match of the regular expression is found.
- **exclude** *regular expression*—Displays all lines except those in which a match of the regular expression is found.

To use one of these output modifiers, type the command followed by the pipe symbol (|), the modifier, and the regular expression that you want to search for or filter. A regular expression is a case-sensitive alphanumeric pattern. It can be a single character or number, a phrase, or a more complex string.

The following example illustrates how to filter output of the **show interface** command to display only lines that include the expression "protocol."

```
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
```

Understanding CLI Error Messages

You may encounter some error messages while using the CLI. Table 5 shows the common CLI error messages.

Table 5 Common CLI Error Messages

Error Message	Meaning	How to Get Help
% Ambiguous command: "show con"	You did not enter enough characters for the command to be recognized.	Reenter the command followed by a space and a question mark (?). The keywords that you are allowed to enter for the command appear.
% Incomplete command.	You did not enter all the keywords or values required by the command.	Reenter the command followed by a space and a question mark (?). The keywords that you are allowed to enter for the command appear.
% Invalid input detected at "^" marker.	You entered the command incorrectly. The caret (^) marks the point of the error.	Enter a question mark (?) to display all the commands that are available in this command mode. The keywords that you are allowed to enter for the command appear.

For more system error messages, see the following documents:

- *Cisco IOS Release 12.2SR System Message Guide*
- *Cisco IOS System Messages, Volume 1 of 2* (Cisco IOS Release 12.4)
- *Cisco IOS System Messages, Volume 2 of 2* (Cisco IOS Release 12.4)

Saving Changes to a Configuration

To save changes that you made to the configuration of a device, you must issue the **copy running-config startup-config** command or the **copy system:running-config nvram:startup-config** command. When you issue these commands, the configuration changes that you made are saved to the startup configuration and saved when the software reloads or power to the device is turned off or interrupted. The following example shows the syntax of the **copy running-config startup-config** command:

```
Router# copy running-config startup-config
Destination filename [startup-config]?
```

You press Enter to accept the startup-config filename (the default), or type a new filename and then press Enter to accept that name. The following output is displayed indicating that the configuration was saved:

```
Building configuration...
[OK]
Router#
```

On most platforms, the configuration is saved to NVRAM. On platforms with a Class A flash file system, the configuration is saved to the location specified by the CONFIG_FILE environment variable. The CONFIG_FILE variable defaults to NVRAM.

Additional Information

- "Using the Cisco IOS Command-Line Interface" section of the *Cisco IOS Configuration Fundamentals Configuration Guide*:

  http://www.cisco.com/en/US/docs/ios/fundamentals/configuration/guide/cf_cli-basics.html

  or

  "Using Cisco IOS XE Software" chapter of the *Cisco ASR1000 Series Aggregation Services Routers Software Configuration Guide*:

  http://www.cisco.com/en/US/docs/routers/asr1000/configuration/guide/chassis/using_cli.html
- Cisco Product Support Resources

  http://www.cisco.com/web/psa/products/index.html
- Support area on Cisco.com (also search for documentation by task or product)

  http://www.cisco.com/en/US/support/index.html
- *White Paper: Cisco IOS Reference Guide*

  http://www.cisco.com/en/US/products/sw/iosswrel/ps1828/products_white_paper09186a008018305e.shtml
- Software Download Center (downloads; tools; licensing, registration, advisory, and general information) (requires Cisco.com User ID and password)

  http://www.cisco.com/kobayashi/sw-center/
- Error Message Decoder, a tool to help you research and resolve error messages for Cisco IOS software

  http://www.cisco.com/pcgi-bin/Support/Errordecoder/index.cgi

- Command Lookup Tool, a tool to help you find detailed descriptions of Cisco IOS commands (requires Cisco.com user ID and password)

  http://tools.cisco.com/Support/CLILookup

- Output Interpreter, a troubleshooting tool that analyzes command output of supported **show** commands

  https://www.cisco.com/pcgi-bin/Support/OutputInterpreter/home.pl\

Guide to Cisco IOS Voice Troubleshooting and Monitoring Features

This chapter contains information about Cisco IOS voice troubleshooting and monitoring features and includes the location of the information in this document or other documents. Use Cisco Feature Navigator to see the features a Cisco IOS software release supports by platform, and to find the image that is best for you. Access Cisco Feature Navigator at http://www.cisco.com/go/fn.

Note To use Cisco Feature Navigator, you must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

New Cisco IOS voice troubleshooting and monitoring features are introduced in the following Cisco IOS software releases:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Cisco IOS Release 12.4(4)T

Feature	Description	Location
MGCP Call Centric Debug	Enables the filtering of MGCP debug output based on selected criteria and standardizes the format of the MGCP debug header. All MGCP debug output for a single call can be identified and correlated across the various layers in IOS software. Filtering debug output reduces extraneous information, making it easier to locate the correct information and reducing the impact to platform performance.	"MGCP Call Centric Debug" section on page 96
Test Call	Provides the ability for a remote station or gateway to establish a call to any destination address from a Test Call station located at a network operations center and to audibly verify the voice path.	"Using the Test Call Feature to Verify Voice Path" section on page 342
Voice Call Debug Filtering on H.323 Gatekeepers	Enables selected debugging traces for voice calls. This feature allows you to filter and trace voice call debug messages based on selected filtering criteria, reducing the volume of output for more efficient troubleshooting.	"Voice Call Debug Filtering on H.323 Gatekeepers" section on page 69

Cisco IOS Release 12.3(8)T

Feature	Description	Location
Voice Troubleshooting Enhancements	Adds standardized voice debug header and voice call debug filtering capabilities to additional and replacement debug commands, enables command profile debugging, and provides data dump for internal data structures.	"Debug Command Output on Cisco IOS Voice Gateways" section on page 17

Cisco IOS Release 12.3(4)T

Feature	Description	Location
Accounting Server Connectivity Failure and Recovery Detection	Provides the option to reject new calls entering the VoIP network and tear down all existing calls upon detecting connectivity failure to the method list that is associated with RADIUS-based accounting servers.	"Accounting Server Connectivity Failure and Recovery Detection" section on page 483
Inactive Call Detection	Enhances Cisco IOS behavior for disconnecting a call when an inactive condition is detected. This feature provides more control for managing these calls.	"Media Inactive Call Detection" section on page 436
SIP Debug Output Filtering Support	Provides the capability for SIP-related debug output to be filtered based on a set of user-defined matching conditions.	"SIP Debug Output Filtering Support" section on page 83
Voice DSP Crash Dump Analysis	Allows Cisco IOS voice platforms using TI DSPs the ability to capture the contents of the DSP memory into a dump file if there is a DSP crash.	"Voice DSP Crash Dump File Analysis" section on page 239
Voice Performance Statistics on Cisco Gateways	Enables the collection of voice call statistics based on user-configured time ranges. The statistics that can be collected are from the following functional areas: • RADIUS accounting • Cisco IOS generated internal error codes (IECs) • Gateway port (interface) statistics	"Voice Performance Statistics on Cisco Gateways" section on page 533
VoIP Debug Filtering on Cisco Voice Gateways	Allows you to filter and trace voice call debug messages based on selected filtering criteria, reducing the volume of output for more efficient troubleshooting.	"Voice Call Debug Filtering on Cisco Voice Gateways" section on page 50
VoIP Internal Error Codes	Generates IECs for gateway-detected errors that cause the gateway to release or refuse a call. IECs enhance troubleshooting for VoIP networks by helping to determine the source and reason for call termination.	"Cisco VoIP Internal Error Codes" section on page 109

Cisco IOS Release 12.2(11)T

Feature	Description	Location
Enhanced Debug Capabilities for Cisco Voice Gateways	Enables you to identify and track a specific call in a multiple-call environment.	"Debug Command Output on Cisco IOS Voice Gateways" section on page 17

Troubleshooting Voice Overview

Voice Call Flow Overview

To troubleshoot problems with voice networks, you must follow the call both inside the router and outside on the network in order to isolate the problem. You must understand the relationship of dial peers and call legs to follow the calls. For detailed information on dial peers and call legs, refer to *Dial Peer Configuration on Voice Gateway Routers*.

The following sections contain information about call flow:

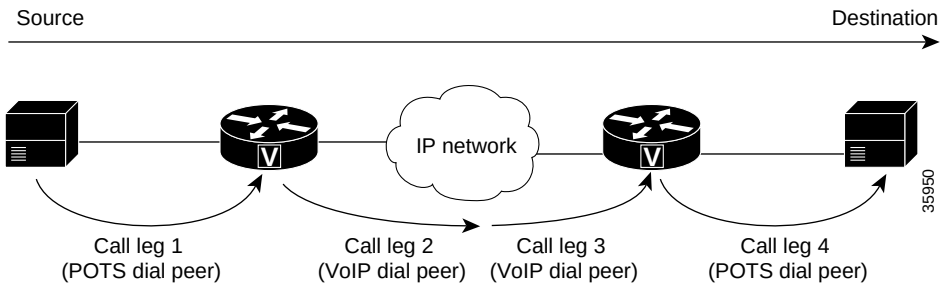
For information about voice network design, refer to *Troubleshooting and Debugging VoIP Call Basics*, document ID 14081.

Call Setup

A voice call over a packet network is segmented into discrete call legs. Each call leg is associated with a dial peer. A call leg is a logical connection between two voice gateways or between a gateway and an IP telephony device (for example, Cisco CallManager or a session initiation protocol (SIP) server).

An example of POTS and VoIP call legs is shown in Figure 1.

Figure 1 Dial Peer Call Legs



Details for the call setup are described in the following sections:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Troubleshooting and debugging should focus first on each leg independently and then on the VoIP call as a whole.

Call Setup Elements

You can isolate where a problem is occurring by determining which dial peer or call leg is having the problem, as described in the following sections:

Source and Destination POTS Dial Peers

POTS dial peers define the characteristics of a traditional telephony network connection. The POTS dial peer maps a dial string to a specific voice port on the local gateway. Normally the voice port connects the gateway to the local public switched telephone network (PSTN), PBX, or analog telephone.

Voice Network Dial Peers

Voice network dial peers define the attributes of a packet voice network connection. Voice network dial peers map a dial string to a remote network device. Some examples of these remote network devices are as follows:

- Destination gateway
- Cisco CallManager
- SIP server
- Open Settlements Protocol (OSP) server (for VoIP using settlement)
- H.323 gatekeeper
- Mail Transfer Agent (MTA) server (for Multimedia Mail over IP scenarios)

The specific type of voice-network dial peer used depends on the packet network technology. Different technologies supported on voice dial peers are as follows:

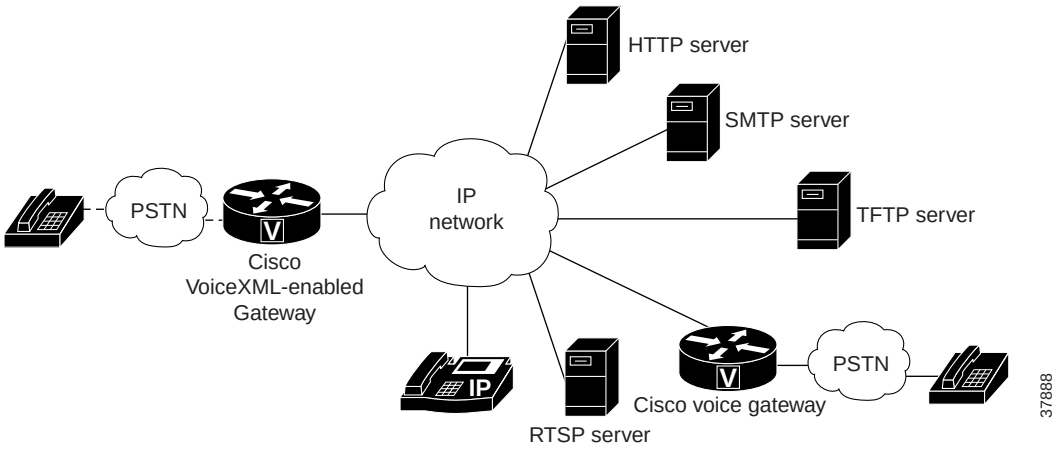
- Voice over IP (VoIP)—The dial peer is mapped to the IP address, Domain Name System (DNS) name, or server-type of the destination VoIP device that terminates the call. This mapping applies to all VoIP protocols such as H.323, SIP, and Media Gateway Control Protocol (MGCP).
- Voice over Frame Relay (VoFR)—The dial peer is mapped to the data-link connection identifier (DLCI) of the interface from which the call exits the router.
- Voice over ATM (VoATM)—The dial peer is mapped to the ATM virtual circuit for the interface from which the call exits the router.
- Multimedia Mail over IP (MMoIP)—The dial peer is mapped to the e-mail address of the Simple Mail Transfer Protocol (SMTP) server. This type of dial peer is used for store-and-forward fax (on-ramp and off-ramp faxing).

The voice-network dial peer also sets the attributes of the network connection, such as which codec to use, the capability to do voice activity detection (VAD), and dual-tone multifrequency (DTMF) relay configuration. A voice network dial peer can point to any device that is compatible with skinny protocol (SCCP), H.323, or SIP. MGCP configuration on a Cisco IOS gateway does not use voice network dial peers.

Voice and Billing Application Call Leg

An IP call leg can go to an external application server. An example of a Cisco IOS VoiceXML application is shown in Figure 2. This call leg is used for voice and billing applications, such as interactive voice response (IVR).

Figure 2 Cisco IOS VoiceXML Application Components



Inbound and Outbound Call Legs

Each call leg can be either an *inbound* or *outbound* call leg, depending on the call's direction. For a call from an IP phone to the PSTN, the VoIP call leg is the inbound call leg, and the POTS leg is the outbound call leg. The opposite is true for a call from the PSTN to the IP phone. Every call has an associated inbound and outbound call leg.

Call Setup Process

A call is segmented into call legs with a dial peer associated with each call leg. See Figure 1.

The process has eight main steps:

1. The POTS call arrives at the originating gateway, and an inbound POTS dial peer is matched.

Note At this stage, if configured on the inbound POTS dial-peer, nondefault inbound POTS services or Tool Command Language (Tcl) applications are used. When you use such services or applications, be certain that the correct inbound POTS dial peer is matched. Examples of services or applications include:

- Direct inward dialing (DID)

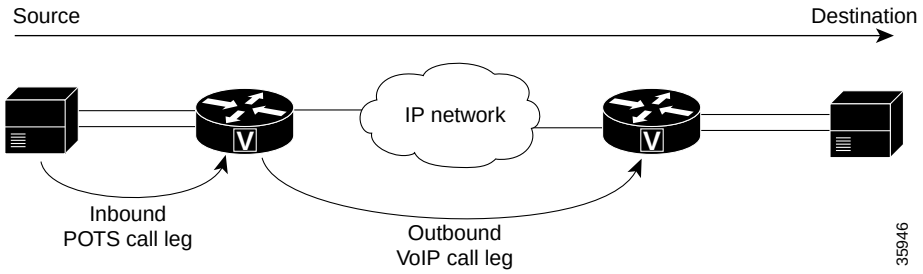
- Tcl-based applications such as IVR, VoIP SIP transfer, or on-ramp faxing for store-and-forward fax.

---

2. After the incoming call is associated with an inbound POTS dial peer, the originating gateway creates an inbound POTS call leg and assigns it a CallEntry ID and global unique identifier (GUID). For more information about these identifiers, see the "Debug Header Format" section on page 18.
3. The originating gateway uses the dialed string to match an outbound voice-network dial peer.
4. After the dialed string are associated with an outbound voice network dial peer, the originating gateway creates an outbound voice-network call leg and assigns it a call ID for the second call leg.

   The preceding process is illustrated in Figure 3.

Figure 3 Call Legs from the Perspective of the Originating Router



5. The voice network call request arrives at the terminating gateway and an inbound voice network dial peer is matched.
6. After the terminating gateway associates the incoming call to an inbound voice-network dial peer, the terminating gateway creates the inbound voice-network call leg and assigns it a call ID for the third call leg.

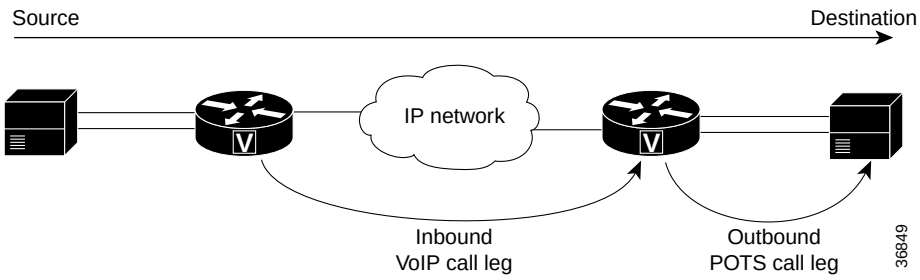
   At this point, both gateways negotiate voice-network capabilities and applications (if required). Default capabilities are not displayed in the gateway Cisco IOS configuration output. Use the **show dial-peer voice** command to display the configured capabilities, services, and applications on POTS and voice-network dial peers:

   - Default capabilities include **codec g729r8**, **vad enable**, **dtmf-relay disable**, **fax-relay disable**, **req-qos best-effort**, **acc-qos best-effort**, and **session protocol cisco** (for H.323).
   - Examples of Tcl applications include remote IP authentication and off-ramp faxing.

   When nondefault capabilities or applications are requested by the originating gateway, the terminating gateways needs to match an inbound voice network dial peer that is configured for these capabilities or applications.
7. The terminating gateway uses the dialed string to match an outbound POTS dial peer.
8. After associating the incoming call setup to an outbound POTS dial peer, the terminating gatewaycreates an outbound POTS call leg, assigns it a call ID, and terminates the call.

   The preceding steps are illustrated in Figure 4.

Figure 4 Call Legs from the Perspective of the Terminating Router



In scenarios where Cisco CallManager is present with a Cisco IOS gateway, the following assumptions apply:

- For inbound calls to Cisco CallManager through a Cisco IOS gateway, the gateway behaves as an originating device. (See Steps 1 to 4.)
- For outbound calls from Cisco CallManager through a Cisco IOS gateway, the gateway behaves as a terminating device. (See Steps 5 to 8.)

Dial Peer Matching

In order for a call to be routed, the attributes set in the dial peers must be matched. Each dial peer can contain a variety of parameters. Some parameters are valid only for voice network or POTS dial peers, and others apply to all dial peers. This section covers the following topics:

- Inbound Dial Peer Matching, page 6
- Outbound Dial Peer Matching, page 9
- Voice Network Dial Peer Matching, page 11
- POTS Dial Peer Matching, page 11

The **destination-pattern** parameter on the **dial peer** command controls call routing. For inbound dial peers, the **destination-pattern** parameter is matched against the calling number, or automatic number identifier (ANI) string. For outbound dial peers, the **destination-pattern** parameter is matched against the called number, or dialed number identification service (DNIS) string. A dial peer with the **destination-pattern** parameter works for both outbound and inbound matching.

Multiple dial peers can match a specific digit string. The gateway usually attempts to perform longest-match routing. For example, if you have two patterns, 555.... and 55501.., and the called party number is 5550123, the gateway matches the peer with 55501..; however, if the call to that peer fails, the gateway attempts to use the other matching peer. If more than one peer has the same destination pattern configured, the **preference** parameter on the dial peer can resolve the priority. The peer with the lowest preference number has the highest priority.

The operational status for a dial peer must be administratively up and valid for it to be matched. To be considered operational, dial peers must meet one of the following conditions:

- **destination-pattern** is configured *and* a **voice-port** or **session target** is also configured.
- **incoming called-number** is configured.
- **answer-address** is configured.

Matching the called party number with the **destination-pattern** parameter is always used to match an outbound dial peer. The inbound dial peer does not affect where the call is routed, but it does determine all of the call properties for the voice network side of the call, regardless of where the outbound peer

terminates.The **incoming called-number** and **answer-address** commands are used only to match inbound dial peers. They are not used for call routing or choosing an outbound dial peer. The **incoming called-number** command matches based on the called party number but does not play a role in where the call is routed. It is used only to select the inbound dial peer. If this match is unsuccessful, the **answer-address** command tries for a match using the calling party number information rather than the called party number. If both of these matches fail, the calling party information is matched against the **destination-pattern** command configured on the dial peers. On POTS ports, the inbound dial peer can be matched based on port configuration. If no match can be made, the dial peer 0 attribute is used. See the "Dial Peer 0" section on page 7 for more information about dial peer 0.

For more information about the operational status of dial peers, refer to *Understanding the Operational Status of Dial Peers on Cisco IOS Platforms*, document ID 12426.

For more detailed information on dial peer matching, configuration steps, and information about parameters, refer to "Dial Peer Features and Configuration" in the *Dial Peer Configuration on Voice Gateway Routers* document.

For additional information about dial peer matching, refer to *Understanding Inbound and Outbound Dial Peers Matching on IOS Platforms*, document ID 14074.

Inbound Dial Peer Matching

The inbound dial peer determines all of the call properties for the voice network side of the call. Inbound POTS dial peers are associated to incoming POTS call legs on the originating gateways. Inbound voice-network dial peers are associated to incoming voice-network call legs of the terminating gateway.

To match inbound call legs to dial peers, the router uses three information elements in the call setup message and five configurable dial peer attributes. The three call setup elements are:

- Called number or DNIS—A set of numbers representing the destination, which is derived from the ISDN setup message or channel associated-signaling (CAS) DNIS.
- Calling number or ANI—A set of numbers representing the origin, which is derived from the ISDN setup message or CAS ANI.
- Voice port—The voice port carrying the call.

The five configurable dial peer attributes are:

- Incoming called number—A string representing the called number or DNIS. It is configured by using the **incoming called-number** dial-peer voice configuration command in POTS or MMoIP dial peers.
- Answer address—A string representing the calling number or ANI. It is configured by using the **answer-address** dial-peer voice configuration command in POTS or VoIP dial peers and is used only for inbound calls from the IP network.
- Destination pattern—A string representing the calling number or ANI. It is configured by using the **destination-pattern** dial-peer voice configuration command in POTS or voice-network dial peers.
- Application—A string representing the predefined application that you enable on the dial peer. It is configured by using the **application** dial-peer voice configuration command on inbound POTS dial peers.
- Port—The voice port through which calls to this dial peer are placed.

When the gateway receives a call setup request, a dial-peer match is made for the incoming call to facilitate routing the call to different session applications. The gateway does not perform digit-by-digit matching. Instead, the full digit string received in the setup request is used for matching against configured dial peers.

The gateway selects an inbound dial peer by matching the information elements in the setup message with the dial peer attributes. The gateway matches these items in the following order:

1. Called number or DNIS with **incoming called-number**

   First, the gateway attempts to match the called number of the call setup request with the configured **incoming called-number** parameter of each dial peer. Because call setups always include DNIS information, Cisco recommends using the **incoming called-number** command for inbound dial peer matching. This attribute has matching priority over the **answer-address** and **destination-pattern** parameter.

2. Calling number or ANI with **answer-address**

   If no match is found in Step 1, the gateway attempts to match the calling number of the call setup request with the **answer-address** of each dial peer. This attribute may be useful in situations where you want to match calls based on the calling number (originating).

3. Calling number (ANI) with **destination-pattern**

   If no match is found in step 2, the gateway attempts to match the calling number of the call setup request to the **destination-pattern** of each dial peer.

4. Voice port (associated with the incoming call setup request) with the configured dial peer **port** parameter (applicable for inbound POTS call legs).

   If no match is found in Step 3, the gateway attempts to match the configured dial-peer **port** parameter to the voice port associated with the incoming call. If multiple dial peers have the same port configured, the dial peer first added in the configuration is matched.

If no match is found using these steps, then dial peer 0 is used. See the "Dial Peer 0" section on page 7 for more information about dial peer 0.

Note Step 4 is not applicable to voice or dial platforms such as the Cisco AS5300 access server, Cisco AS5350 universal gateway, Cisco AS5400 universal gateway, Cisco AS5800 universal gateway, and Cisco AS5850 universal gateway. If any one of first three steps are not used, then dial-peer 0 is matched and the call is treated as a dial modem call. This call treatment can result in getting modem tones as opposed to dial tone for inbound calls.

Only one condition must be met for the gateway to select a dial peer. The gateway stops searching as soon as one dial peer is matched. It is not necessary for all the attributes to be configured in the dial peer or that every attribute match the call setup information.

The longest prefix matching criteria applies while each step is performed. At each step, if multiple matches are found, the one with the longest explicit match is chosen.

For more information on dial-peer matching, configuration steps, and information about parameters, refer to "Dial Peer Features and Configuration" in the *Dial Peer Configuration on Voice Gateway Routers* document.

Dial Peer 0

If no inbound peer can be matched by the defined criteria, the inbound peer is set to dial peer 0, which sometimes appears as pid:0. The characteristics of dial peer 0 cannot be changed.

Note Cisco universal gateways, such as the Cisco AS5350, Cisco AS5400, Cisco AS5800, and Cisco AS5850, require configured inbound dial peers to match incoming POTS calls in order to be accepted as a voice call. If there is no inbound dial peer match, the call is treated and processed as a dialup (modem) call.

For an inbound voice network call, dial peer 0 has the following characteristics:

- Supports any codec
- No DTMF relay
- IP precedence 0
- VAD-enabled
- No RSVP support
- Fax-rate voice

Dial peer 0 fails to negotiate nondefault capabilities, services, and applications, such as DTMF relay or disabled VAD.

For an incoming POTS call, dial peer 0 has the following characteristics:

- No applications
- No DID

Note Avoid using dial peer 0. Having the **incoming called-number** parameter configured correctly ensures that the dial peer is always matched with the parameters you want when placing outbound calls through a gateway. Many problems with calling out through a Cisco IOS gateway are due to codec, VAD, and DTMF-relay misconfigurations when dial peer 0 is being matched. To display which dial peers are being matched for an active call, use the **show call active voice brief** command.

Inbound Dial Peer Configuration Tips

For inbound dial peers, the following configuration tips apply to certain configurations:

Session Target Command

For inbound dial peers, the **session target** command is ignored.

ISDN Overlap Configuration

When the timer (**T**) character is included at the end of the destination pattern, the router collects dialed digits until the interdigit timer expires (10 seconds, by default) or until you dial the termination character (the default is #). The timer character must be an uppercase **T**. In most cases, you must configure the **T** indicator only when the router uses two-stage dialing. If DID is configured in the inbound POTS dial peer, the router uses one-stage dialing, which means that the full dialed string is used to match outbound dial peers. The only exception is when the **isdn overlap-receiving** command is configured; the ISDN overlap-receiving feature requires the **T** indicator.

When the **isdn overlap-receiving** command is configured on ISDN interfaces, dial peers are checked for matches after every digit is received at the ISDN layer. If a full match is made, the call is routed immediately without waiting for additional digits. The **T** indicator can be used to suspend this digit-by-digit matching and force the gateway to wait until all digits are received. The **T** refers to the T302 interdigit timer at the ISDN level, configurable under the serial interface associated with the ISDN interface. ISDN also provides other mechanisms to indicate the end of digits, such as setting the Sending Complete Information Element (IE) in a Q.931 information message.

Empty Calling Field with Variable-Length Dial Plans

In some voice network configurations, variable-length dial plans are required, especially if the network connects two or more countries where telephone number strings could be different lengths. However, this configuration can result in the calling number field being replaced with an empty number.

If you enter the **T** character in the **destination-pattern** parameter for your dial peer, the router can be configured to accept a fixed-length dial string, and then wait for additional dialed digits. For example, the following dial peer configuration shows how the T character can be set to allow variable-length dial strings:

```
dial-peer voice 1 pots
   destination-pattern 9T
   port 1/0:1
```

If an incoming call arrives with no calling number information and is matched with the POTS dial peer shown based on the **destination-pattern 9T**, the gateway uses the "9" digit as the calling number and forwards the call to the corresponding device. To eliminate this behavior of replacing the empty calling number field, create a dummy POTS dial peer with just the **incoming called-number** command configured. Because the **incoming called-number** statement has higher priority than the **destination-pattern** parameter for inbound POTS matching, dial-peer voice 2 is the POTS dial peer that is used:

```
dial-peer voice 1 pots
    destination-pattern 9T
    port 1/0:1
!
dial-peer voice 2 pots
    incoming called-number .
```

Outbound Dial Peer Matching

The method a router uses to select an outbound dial peer depends on whether DID is configured in the inbound POTS dial peer. If DID is not configured in the inbound POTS dial peer, the router collects the incoming dialed string digit by digit. When one dial peer is matched, the router immediately places the call using the configured attributes in the matching dial peer.

If DID is configured in the inbound POTS dial peer, the router uses the full incoming dial string to match the destination pattern in the outbound dial peer. With DID, the setup message contains all the digits necessary to route the call; no additional digit collection is required. If more than one dial peer matches the dial string, all of the matching dial peers are used to form a rotary group. The router attempts to place the outbound call leg using all of the dial peers in the rotary group until one is successful.

For matching outbound dial peers, the gateway uses the **destination-pattern** dial peer command.

- On POTS dial peers, the **port** command is used to forward the call.
- On voice network dial peers, the **session target** command is used to forward the call.

DID Configuration

An example of incoming dial-peer configured with DID follows:

```
dial-peer voice 1 pots
  incoming called-number 81690
  voice-port 0:D
  direct-inward-dial
```

On DID calls, also referred to as one-stage dialing, the setup message contains all digits necessary to route the call, and the gateway should not do subsequent digit collection. When the gateway searches for an outbound dial peer, it uses the entire incoming dial string. This matching is by default variable-length. It is not done digit by digit because by DID definition all digits have been received. The following example helps clarify this concept:

If the DID dial-string is “81690,” the router matches dial peer 4 and forwards the complete dial-string “81690.”

```
dial-peer voice 3 voip
 destination-pattern 816
 session target ipv4:172.22.10.1
!
dial-peer voice 4 voip
 destination-pattern 81690
 session target ipv4:172.22.10.1
```

Two-Stage Dialing Configuration

If DID is not configured on the matched incoming dial peer, the gateway enters digit collection mode, called two-stage dialing. Digits are collected inband. Outbound dial peer matching is done on a digit-by-digit basis. The gateway checks for dial peer matches after receiving each digit and then routes the call when a full match is made.

In this example, the dial string is “81690.” Immediately after the router receives the digit “6,” it matches dial peer 3 and routes the call, forwarding only the digits “816.”

```
dial-peer voice 3 voip
 destination-pattern 816
 session target ipv4:172.22.10.1
!
dial-peer voice 4 voip
 destination-pattern 81690
 session target ipv4:172.22.10.1
```

In this example, dial-peer 3 is configured for wild card matching:

```
dial-peer voice 3 voip
 destination-pattern 816..
 session target ipv4:172.22.10.1
!
dial-peer voice 4 voip
 destination-pattern 81690
 session target ipv4:172.22.10.1
```

In this case, the longest-prefix rule applies and dial peer 4 is matched for the outbound call leg.

Variable-Length Dial Plans

There are situations where expected dial strings do not have a set number of digits. In such cases, it is usually best to use variable-length dial peers by configuring the **T** terminator on the dial-peer **destination-pattern** command.

The **T** terminator forces the gateway to wait until the full dial string is received by doing the following:

- Waiting for a specified interdigit timeout before routing the call.
- Routing the call once it receives the **#** termination character in the dial string. For example, if you dialed **5550112#**, the **#** would indicate to the router that you dialed all the digits and that all digits before the **#** should be used to match a dial peer.

In the following example, the router receives a call setup from the network with dial string “95550112.” Dial peer 2 then forwards the digits “5550112” to the PSTN.

```
dial-peer voice 2 pots
 destination-pattern 9T
 port 2/0:23
```

In the following example, the dial string from an inbound POTS interface is “81690”:

```
dial-peer voice 3 voip
 destination-pattern 8T
 session target ipv4:172.22.10.1
!
dial-peer voice 4 voip
 destination-pattern 81690T
 session target ipv4:172.22.10.1
```

In this case, the longest-prefix rule applies, and dial peer 4 is matched for the outbound call leg.

Note The default interdigit timeout is set for 10 seconds. To modify this value, use the **timeouts interdigit** voice-port command.

Anytime the **T** is used with **destination-pattern** parameter, it must be preceded by a **.** or digits (**.T** or **555T**, for example). Using **T** on its own causes the dial peers to act improperly and affects how calls are handled by the router.

Voice Network Dial Peer Matching

When a call comes in through VoIP, the voice gateway searches for an inbound peer by looking through all the configured voice network dial peers. Using regular digit matching rules, it tries to match the peer in the following order:

1. Match the called number with the incoming called number.
2. Match the calling number with the **answer-address** parameter.
3. Match the calling number with the destination pattern.
4. Otherwise, use dial peer 0. See the “Dial Peer 0” section on page 7 for more information.

POTS Dial Peer Matching

When a call comes in through POTS, the voice gateway searches for an inbound peer by looking through all the configured POTS dial peers. Using regular digit-matching rules, it tries to match the peer in the following order:

1. Match the called number with the incoming called number.
2. Match the calling number with the **answer-address** parameter.
3. Match the calling number with the destination pattern.

If the inbound interface is not PRI or BRI, or if a PRI or BRI interface does not match a dial peer using the preceding three rules, the voice gateway matches a POTS dial peer that has the inbound port configured if any of the following parameters are configured:

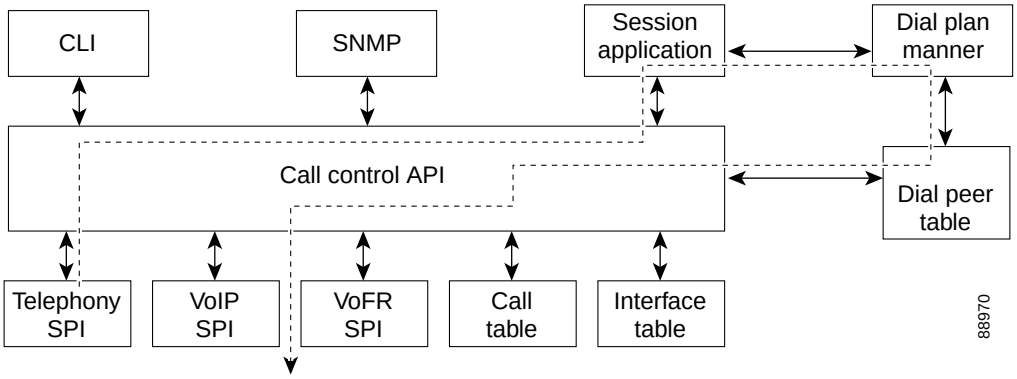
- **destination-pattern**
- **answer-address**
- **incoming called-number**

The **answer-address** parameter overwrites the call’s calling-party number or ANI.

Call Flow Through Router Components

You need to understand the call path through the router in order to determine where a problem is occurring. The call path through a router is shown in Figure 5.

Figure 5 Router Call Flow



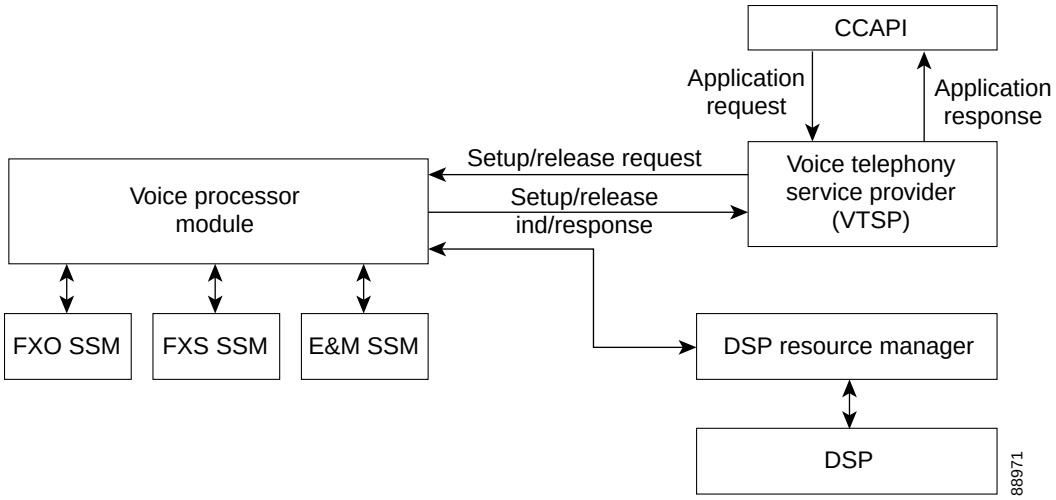
The following definitions explain the function of the main components displayed in the router call flow diagram:

- **Call control application programming interface (CCAPI)**—Three clients make use of the CCAPI: command-line interface (CLI), Simple Network Management Protocol (SNMP) agent and Session Application. The CCAPI main functions are:
  - Identify the call legs (Which dial peer is it? Where did it come from?).
  - Decide which session application takes the call (Who handles it?).
  - Invoke the packet handler.
  - Conference the call legs together.
  - Start recording call statistics.
- **Session application and dial plan mapper**—The session application uses the dial plan mapper to map a number to a dial peer (local POTS or remote VoIP). The dial plan mapper uses the dial peer table to find active dial peers.
- **Telephony and VoIP service provider interface (SPI)**—The telephony SPI communicates with the POTS dial peers. The VoIP SPI is the specific interface to the VoIP peers. Telephony or digital signal processor (DSP) drivers deliver services to the Telephony SPI; the VoIP SPI relies on session protocols.

Telephony Interface Architecture

Figure 6 shows the architecture of Cisco router telephony building blocks and how they interact with each other.

Figure 6 Telephony Interface Call Flow



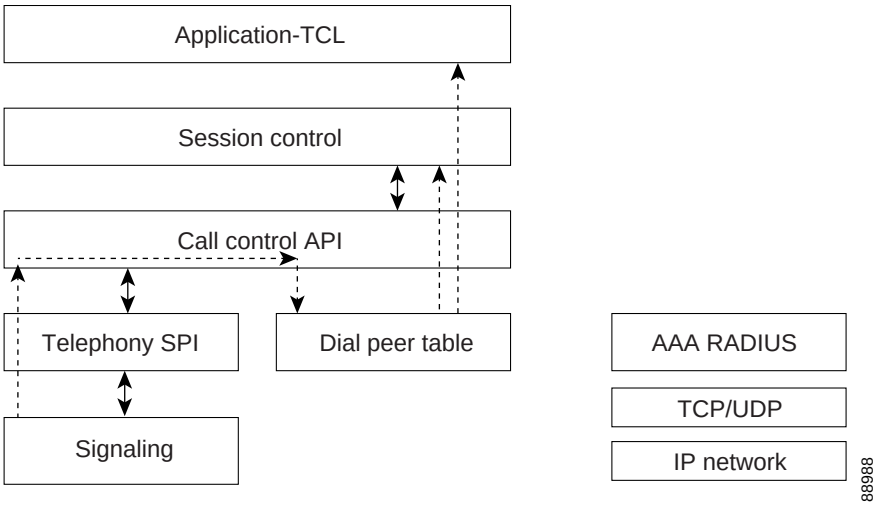
The elements in Figure 6 have the following functions and definitions:

- **CCAPI**—Software entity that establishes, terminates, and bridges call legs.
- **Voice telephony service provider (VTSP)**—Cisco IOS process that services requests from the CCAPI and formulates the appropriate requests to the DSPs or the voice processor module (VPM).
- **Voice processor module (VPM)**— The VPM is in charge of bridging and coordinating signaling processes between the telephony port signaling state machine (SSM), the DSP resource manager, and the VTSP.
- **DSP resource manager (DSPRM)**—The DSPRM provides interfaces by which the VTSP can send and receive messages to and from the DSPs.
- **Packet handler**—The packet handler forwards packets between the DSPs and the peer call legs.
- **Call peer**—The call peer can be another telephony voice connection (POTS), VoFR, VoATM, or a VoIP connection.

Voice Application Interface

For voice applications, the router interacts with an application server. In the example in Figure 7, the call from the router is routed through the application module to the AAA server.

Figure 7 Voice Application Interface



Debug Command Output on Cisco IOS Voice Gateways

The debugging capability for Cisco voice gateways enables you to identify and track a specific call in a multiple-call environment. This capability allows you to correlate call information between gateways or to identify specific debug messages associated with a single call when multiple voice calls were simultaneously active.

Voice debug output contains a standardized header to the debug outputs of multiple voice modules, such as voice telephony service provider (VTSP), call control application program interface (CCAPI), session application (SSAPP), and interactive voice response (IVR).

The following information can be found in the *Cisco IOS Debug Command Reference*:

- Using debug commands is contained in the "Using Debug Commands" chapter.
- Conditionally triggered debugging is contained in the "Conditionally-Triggered Debugging" chapter.
- Details about individual **debug** commands are listed.

This chapter contains the following information:

Voice Debug Concepts

The following concepts provide background for the enhanced debug capabilities on Cisco voice gateways:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Debug Header Format

You can control the contents of the standardized header with the following display options:

- Short 6-byte global unique identifier (GUID)
- Full 16-byte GUID
- Short header that contains only the CallEntry ID

The format of the GUID headers is as follows:
//CallEntryID/GUID/Module-dependent-list/Function-name:

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

The parameters in the header are as follows:

- CallEntryID—Numerically identifies a specific call leg such as an incoming ISDN call or an outgoing H.323 call. The CallentryID ranges from 1 to 32767. When the CallEntryID is not available to the function producing the debug message, the header displays the CallEntryID field as "-1."
- GUID—Each call is assigned a GUID, which is generally used for billing purposes. The GUID is a 16-byte quantity that uniquely identifies a call throughout the entire network and over time. Gateway information and time stamp are embedded in the GUID. The GUID is sometimes called a conference point. See the "CallEntry ID and GUID Call Legs" section on page 4 for an example of the GUID and CallEntry ID call legs.

  By default, the debug header displays a more compact 6-byte form of the GUID that is generally unique enough for debugging purposes. If you need to correlate GUID information to third-party devices or to conduct debugging sessions lasting more than a month, you can display the full 16-byte header with the **voice call debug** command using the **full-guid** keyword.

  When the GUID information is unavailable to the module producing the **debug** message, the header displays the GUID field as blank or filled with "x" characters (/xxxxxxx/). For non-IVR debugs, if the GUID field is unavailable, the header is //CallentryID/xxxxxxxx/. For IVR debugs, the header is //CallentryID//. IVR rarely has GUID information but reserves the space in order to have field-by-field consistency.
- Module-dependent-list—These parameters are dependent on which module is used. The dependent parameters for the various modules are shown in Table 1.

Table 1 Module-Dependent List Parameters

Module	Dependent Parameters
CCAPI	None.
VTSP	Port name, channel number, DSP slot, and DSP channel number.
SSAPP	Dial-peer number and the number of the connection through the dial peer. (Multiple sessions can be associated with a single dial peer.)
IVR	See the "IVR Module-Dependent List" section on page 3.

- Function-name—Identifies call progress though the internal modules of the Cisco IOS code. This is free-form debugging used by developers.

IVR Module-Dependent List

The module-dependent list for IVR information has the following form:
<Module Name> : [LP:] [<OBJ><OBJ-ID>: [<OBJ><OBJ-ID>:[...]]]

Module names are shown in Table 2. The LP parameter is the *link point*, which is the point where two or more objects can be associated . The OBJ parameter is the object name, shown in Table 3. The OBJ-ID is a numeric identifier for the object. The ranges are also shown in Table 3.

Table 2 IVR Module Names

Module Name	Module
AAPL	Application infrastructure
DCM	Digit collect
DPM	Dynamic prompt
MCM	Media content
MSM	Media stream
MSW	Media service wrapper
PCM	Place call (call setup)
RTSP	Real-time streaming protocol (RTSP) client
TCL2	Tcl IVR 2

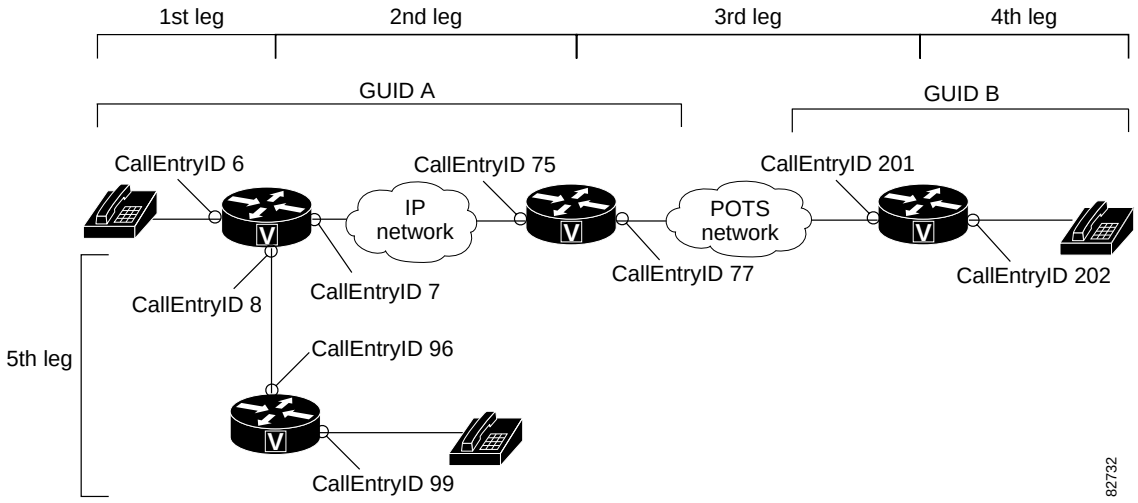
Table 3 Object Names and Identifier Values

Object Name	Description	Identifier Range
CN	Connection	–2147483647 to 2147483647 with 0 and –1 being invalid (either) or unavailable (latter)
DP	Dynamic prompt	1 to –4294967295 or 0 (invalid or unavailable)
GS	Generic stream	1 to –4294967295 or 0 (invalid or unavailable)
HN	Handler	0 to –4294967295
LG	Call leg	1 to –2147483647 or –1 (invalid or unavailable)
MC	Media content	1 to –4294967295 or 0 (invalid or unavailable)
MR	Media content reader	1 to –4294967295 or 0 (invalid or unavailable)
MS	Media stream	1 to –4294967295 or 0 (invalid or unavailable)
RS	RTSP session	1 to –4294967295 or 0 (invalid or unavailable)

CallEntry ID and GUID Call Legs

The parts of the call that are identified by CallEntry IDs and GUIDs are shown in Figure 8. The example shown in the figure is a conference call in which two of the participants are on the same IP network and the third participant is called over the POTS network. Most of the IP-based voice signaling protocols (like H.225) carry the GUID information between gateways. Traditional voice connections such as POTS and ISDN have no means to carry a GUID. A voice gateway creates a new GUID when one is not available from the call originator. In the example, call legs 1, 2, and 5 use GUID A, and leg 4 uses GUID B. Leg 3 is across the POTS network, so no GUID is associated with it. There is a CallEntry ID for each incoming or outgoing call on each gateway.

Figure 8 GUIDs and CallEntry IDs in a Conference Call



Enabling Command Profile Debugging

You can enable a set of debugs based upon a specific profile. You can enable the following profiles:

- Fax Debug Profile
- Modem Debug Profile
- Voice Debug Profile

Enabling the debugging on each of these profiles produces many types of debug information. The advantage of using profile debugging is that instead of having to enable several debug commands to get a full picture of the problem, you can enable a profile which gives you all of the debugs that are appropriate.

When profile debugging is enabled, a large number of debug messages are produced which might affect system performance. For this reason, you should only use profile debugging during low traffic periods or on a non-production system.

Caution The **debug voip profile** commands generate debug messages from many VoIP components, which generates a large number of debug messages. The number of messages can cause a performance impact on your router. This command should only be used during low traffic periods.

Fax Debug Profile

The debug commands that are activated within the fax debug profile depends on the syntax that you use when you enable the command. The syntax for the fax debug profile is as follows:

debug voip profile fax { **mail** | **relay** { **application** | **signaling** }}

For more information on command syntax, refer to the *Cisco IOS Debug Command Reference*.

If you use the **debug voip profile fax mail** command, you enable the following debug commands for an onramp or offramp fax mail call:

- **debug crm all**
- **debug csm voice**
- **debug fax fmsp all**
- **debug fax mta all**
- **debug fax mmoip aaa**
- **debug isdn q931**
- **debug tgrm all**
- **debug voip application all**
- **debug voip application vxml all**
- **debug voip ccapi all**
- **debug voip dsm all**
- **debug voip dspapi all**
- **debug voip hpi all**
- **debug voip ivr all**
- **debug voip vtsp all**

The following debug commands are enabled for access servers with MICA modem cards:

- **debug fax receive all**
- **debug fax send all**
- **debug fax fmail client**
- **debug fax fmail server**
- **debug fax text-to-fax**
- **debug fax tiff reader**
- **debug fax tiff writer**

The following debug options are enabled for access servers with universal port dial feature cards:

- **debug fax fmsp all**
- **debug fax foip all**

- **debug fax dmsp all**
- **debug fax mspi all**
- **debug voip application vxml all**
- **debug voip ivr all**

If you use the **debug voip profile fax relay** command, you enable the **debug fax relay t30 all-level-1** and the sets specified by either the **application** or **signaling** keyword.

For the **debug voip profile fax relay application** command, the following debugs are enabled for fax relay applications:

- **debug voip application all**
- **debug voip application vxml all**
- **debug voip ccapi all**
- **debug voip dialpeer all**
- **debug voip ivr all**

For the **debug voip profile fax relay signaling** command, the following debugs are enabled for fax relay signaling:

- **debug cch323 all**
- **debug ccsip error**
- **debug ccsip messages**
- **debug cdapi detail**
- **debug cdapi events**
- **debug crm all**
- **debug csm voice**
- **debug gtd error**
- **debug gtd events**
- **debug h225 asn1**
- **debug h225 events**
- **debug h225 q931**
- **debug h245 asn1**
- **debug h245 event**
- **debug isdn q931**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp media**
- **debug mgcp packets**
- **debug mgcp voipcac**
- **debug rtpspi all**
- **debug tgrm all**
- **debug voip ccapi all**

- **debug voip dsm all**
- **debug voip dspapi all**
- **debug voip hpi all**
- **debug voip rawmsg**
- **debug voip tsp all**
- **debug voip vtsp all**

For detailed information about the individual debug commands, refer to the *Cisco IOS Debug Command Reference*.

Modem Debug Profile

The debug commands that are activated within the modem debug profile depends on the syntax that you use when you enable the command. The syntax for the modem debug profile is as follows:

debug voip profile modem { **pass-through signaling** | **relay signaling** }

For more information on command syntax, refer to the *Cisco IOS Debug Command Reference*.

If you use the **debug voip profile modem pass-through signaling** command, you enable the following debug commands for modem pass-through signaling:

- **debug cch323 all**
- **debug ccsip error**
- **debug ccsip messages**
- **debug cdapi detail**
- **debug cdapi events**
- **debug csm voice**
- **debug crm all**
- **debug gtd error**
- **debug gtd events**
- **debug h225 asn1**
- **debug h225 events**
- **debug h225 q931**
- **debug isdn q931**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp media**
- **debug mgcp packets**
- **debug mgcp voipcac**
- **debug rtpspi all**
- **debug tgrm all**
- **debug voip ccapi all**
- **debug voip dsm all**

- **debug voip dspapi all**
- **debug voip hpi all**
- **debug voip rawmsg**
- **debug voip tsp all**
- **debug voip vtsp all**

If you use the **debug voip profile modem relay signaling** command, you enable the following debug commands for modem relay signaling:

- **debug cch323 all**
- **debug ccsip error**
- **debug ccsip messages**
- **debug cdapi detail**
- **debug cdapi events**
- **debug csm voice**
- **debug crm all**
- **debug h245 asn1**
- **debug h245 event**
- **debug isdn q931**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp media**
- **debug mgcp packets**
- **debug mgcp voipcac**
- **debug tgrm all**
- **debug voip ccapi all**
- **debug voip dsm all**
- **debug voip dspapi all**
- **debug voip hpi all**
- **debug voip tsp all**
- **debug voip vtsp all**

For detailed information about the individual debug commands, refer to the *Cisco IOS Debug Command Reference*.

Voice Debug Profile

The debug commands that are activated within the voice debug profile depends on the syntax that you use when you enable the command. The syntax for the voice debug profile is as follows:

debug voip profile voice { **application** | **signaling** }

For more information on command syntax, refer to the *Cisco IOS Debug Command Reference*.

If you use the **debug voip profile voice application** command, you enable the following debug commands for voice applications:

- **debug voip application all**
- **debug voip application vxml all**
- **debug voip ccapi all**
- **debug voip dialpeer all**
- **debug voip ivr all**

If you use the **debug voip profile voice signaling** command, you enable the following debug commands for voice signaling:

- **debug cch323 all**
- **debug ccsip error**
- **debug ccsip messages**
- **debug cdapi detail**
- **debug cdapi events**
- **debug crm all**
- **debug csm voice**
- **debug gtd error**
- **debug gtd events**
- **debug h225 asn1**
- **debug h225 events**
- **debug h225 q931**
- **debug h245 asn1**
- **debug h245 event**
- **debug isdn q931**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp media**
- **debug mgcp packets**
- **debug mgcp voipcac**
- **debug rtpspi all**
- **debug tgrm all**
- **debug voip ccapi all**
- **debug voip dsm all**
- **debug voip dspapi all**
- **debug voip hpi all**
- **debug voip rawmsg**
- **debug voip tsp all**
- **debug voip vtsp all**

For detailed information about the individual debug commands, refer to the *Cisco IOS Debug Command Reference*.

Enabling Individual Debug Commands for CCAPI, TSP, and VTSP Debugging

If you want to use a debug command for CCAPI, TSP, or VTSP and you want to limit the output so that the performance on the router is not impacted, you can select to use individual parts of the debug command. You can use **debug voip ccapi individual** *range*, **debug voip tsp individual** *range*, or **debug voip vtsp individual** *range* to get individual debug output for each of these processes. Table 4, Table 5, and Table 6 show the specific debugs for the individual ranges.

Table 4 CCAPI Individual Debug Values

Value	CCAPI Debug Function
1	CC_IDMSG_API_DISPLAY_IES
2	CC_IDMSG_SETUP_IND_COMM_2
3	CC_IDMSG_SETUP_IND_COMM_3
4	CC_IDMSG_SETUP_IND_COMM_4
5	CC_IDMSG_ALERT_IND_5
6	CC_IDMSG_ALERT_IND_6
7	CC_IDMSG_CONNECT_IND_7
8	CC_IDMSG_CONNECT_IND_8
9	CC_IDMSG_RECONNECT_IND_9
10	CC_IDMSG_DISCONNECTED_IND_10
11	CC_IDMSG_DISCONNECTED_IND_11
12	CC_IDMSG_DISCONNECTED_IND_12
13	CC_IDMSG_DISCONNECT_DONE_IND_13
14	CC_IDMSG_DISCONNECT_DONE_IND_14
15	CC_IDMSG_DISCONNECT_DONE_IND_15
16	CC_IDMSG_PRE_DISC_CAUSE_16
17	CC_IDMSG_PRE_DISC_CAUSE_17
18	CC_IDMSG_DIGIT_BEGIN_IND_18
19	CC_IDMSG_DIGIT_END_IND_19
20	CC_IDMSG_DIGIT_END_IND_20
21	CC_IDMSG_DIGIT_END_NO_TERM_21
22	CC_IDMSG_TONE_IND_22
23	CC_IDMSG_FEATURE_IND_23
24	CC_IDMSG_MODIFY_DONE_IND_24
25	CC_IDMSG_MODIFY_MODE_DONE_IND_25
26	CC_IDMSG_INBAND_MSG_RCVD_IND_26
27	CC_IDMSG_INBAND_MSG_DONE_IND_27

Table 4 CCAPI Individual Debug Values (continued)

Value	CCAPI Debug Function
28	CC_IDMSG_UPD_CALL_INFO_IND_28
29	CC_IDMSG_GEN_NTK_ALERT_EVENT_29
30	CC_IDMSG_VOICE_MODE_EVENT_30
31	CC_IDMSG_VOICE_MODE_EVENT_31
32	CC_IDMSG_DIALING_COMPLETE_IND_32
33	CC_IDMSG_DIGITS_DONE_IND_33
34	CC_IDMSG_DIGITS_DONE_IND_34
35	CC_IDMSG_VBD_XMIT_DONE_IND_35
36	CC_IDMSG_FWD_SETUP_IND_36
37	CC_IDMSG_RSVP_DONE_IND_37
38	CC_IDMSG_AUDIT_RSP_IND_38
39	CC_IDMSG_XFR_STATUS_IND_39
40	CC_IDMSG_XFR_STATUS_IND_40
41	CC_IDMSG_XFR_DONE_IND_41
42	CC_IDMSG_XFR_DONE_IND_42
43	CC_IDMSG_XFR_DONE_IND_43
44	CC_IDMSG_TGT_CID_ACTIVE_RCD_44
45	CC_IDMSG_MODIFY_MEDIA_IND_45
46	CC_IDMSG_MODIFY_MEDIA_ACK_IND_46
47	CC_IDMSG_MODIFY_MEDIA_REJ_IND_47
48	CC_IDMSG_MODEM_CALL_START_IND_48
49	CC_IDMSG_MODEM_CALL_DONE_IND_49
50	CC_IDMSG_ACCT_STATUS_IND_50
51	CC_IDMSG_NW_STATUS_IND_51
52	CC_IDMSG_DESTINFO_IND_52
53	CC_IDMSG_LOOPBACK_DONE_IND_53
54	CC_IDMSG_RT_PACKET_STATS_IND_54
55	CC_IDMSG_CUT_PROGRESS_IND_55
56	CC_IDMSG_CUT_PROGRESS_IND_56
57	CC_IDMSG_PROCEEDING_IND_57
58	CC_IDMSG_FACILITY_IND_58
59	CC_IDMSG_INFO_IND_59
60	CC_IDMSG_PROGRESS_IND_60
61	CC_IDMSG_USERINFO_IND_61
62	CC_IDMSG_DISC_PROG_IND_62
63	CC_IDMSG_DISC_PROG_IND_63

Table 4 CCAPI Individual Debug Values (continued)

Value	CCAPI Debug Function
64	CC_IDMSG_PING_DONE_IND_64
65	CC_IDMSG_COT_TEST_DONE_IND_65
66	CC_IDMSG_PROCESS_DONE_IND_66
67	CC_IDMSG_ASSOCIATED_IND_67
68	CC_IDMSG_SUSPEND_IND_68
69	CC_IDMSG_SUSPEND_ACK_IND_69
70	CC_IDMSG_SUSPEND_REJ_IND_70
71	CC_IDMSG_RESUME_IND_71
72	CC_IDMSG_RESUME_ACK_IND_72
73	CC_IDMSG_RESUME_REJ_IND_73
74	CC_IDMSG_IF_SETUP_REQ_PRIV_74
75	CC_IDMSG_IF_SETUP_REQ_PRIV_75
76	CC_IDMSG_IF_ALLOCATE_DSP_76
77	CC_IDMSG_CONNECT_77
78	CC_IDMSG_CONNECT_78
79	CC_IDMSG_PING_79
80	CC_IDMSG_DISCONNECT_80
81	CC_IDMSG_DISCONNECT_81
82	CC_IDMSG_DISCONNECT_82
83	CC_IDMSG_ALERT_83
84	CC_IDMSG_ALERT_84
85	CC_IDMSG_CUT_PROGRESS_85
86	CC_IDMSG_CUT_PROGRESS_86
87	CC_IDMSG_CUT_PROGRESS_87
88	CC_IDMSG_DISC_PROG_88
89	CC_IDMSG_DISC_PROG_89
90	CC_IDMSG_SET_PEER_90
91	CC_IDMSG_SET_PEER_91
92	CC_IDMSG_PROCEEDING_92
93	CC_IDMSG_SETUP_REQ_93
94	CC_IDMSG_SETUP_REQ_94
95	CC_IDMSG_SETUP_REQ_95
96	CC_IDMSG_SETUP_REQ_96
97	CC_IDMSG_SETUP_REQ_97
98	CC_IDMSG_SETUP_REQ_98
99	CC_IDMSG_SETUP_REQ_99

Table 4 CCAPI Individual Debug Values (continued)

Value	CCAPI Debug Function
100	CC_IDMSG_SETUP_REQ_100
101	CC_IDMSG_SETUP_REQ_101
102	CC_IDMSG_SETUP_ACK_102
103	CC_IDMSG_FACILITY_103
104	CC_IDMSG_TRANSFER_REQ_104
105	CC_IDMSG_GET_CONSULT_ID_105
106	CC_IDMSG_FORWARD_TO_106
107	CC_IDMSG_INFO_107
108	CC_IDMSG_NOTIFY_108
109	CC_IDMSG_PROGRESS_109
110	CC_IDMSG_PRE_DISC_110
111	CC_IDMSG_PRE_DISC_111
112	CC_IDMSG_USER_INFO_112
113	CC_IDMSG_MODIFY_113
114	CC_IDMSG_DIGIT_114
115	CC_IDMSG_DIGIT_DIAL_115
116	CC_IDMSG_DIGIT_DIAL_STOP_116
117	CC_IDMSG_FEATURE_117
118	CC_IDMSG_FEATURE_ENABLE_118
119	CC_IDMSG_ASSOCIATE_STREAM_119
120	CC_IDMSG_ASSOCIATE_STREAM_120
121	CC_IDMSG_DISASSOCIATE_STREAM_121
122	CC_IDMSG_DISASSOCIATE_STREAM_122
123	CC_IDMSG_GENERATE_TONE_INFO_123
124	CC_IDMSG_SET_DIGIT_TIMEOUTS_124
125	CC_IDMSG_SET_DIGIT_TIMEOUTS_125
126	CC_IDMSG_SUSPEND_126
127	CC_IDMSG_SUSPEND_ACK_127
128	CC_IDMSG_SUSPEND_REJ_128
129	CC_IDMSG_RESUME_129
130	CC_IDMSG_RESUME_ACK_130
131	CC_IDMSG_RESUME_REJ_131
132	CC_IDMSG_UPDATE_REDIRECT_NUM_132
133	CC_IDMSG_BABBLER_AUDIT_133
134	CC_IDMSG_CONFERENCE_CREATE_134
135	CC_IDMSG_CONFERENCE_CREATE_135

Table 4 CCAPI Individual Debug Values (continued)

Value	CCAPI Debug Function
136	CC_IDMSG_CONFERENCE_CREATE_136
137	CC_IDMSG_CONFERENCE_DESTROY_137
138	CC_IDMSG_CONFERENCE_DESTROY_138
139	CC_IDMSG_CONFERENCE_DESTROY_139
140	CC_IDMSG_LOOPBACK_140
141	CC_IDMSG_COT_TEST_141
142	CC_IDMSG_HANDOFF_142
143	CC_IDMSG_APP_RETURN_143
144	CC_IDMSG_T38_FAX_START_144
145	CC_IDMSG_T38_FAX_DONE_145

Table 5 TSP Individual Debug Values

Value	TSP Debug Function
1	INDIVIDUAL_TSP_DEBUG_TDM_HAIRPIN_CONNECT_001
2	INDIVIDUAL_TSP_DEBUG_TDM_HAIRPIN_DISCONNECT_002
3	INDIVIDUAL_TSP_DEBUG_CCRAWMSG_ENCAP_003
4	INDIVIDUAL_TSP_DEBUG_CDAPI_FORM_MSG_BASIC_SS_INFO_004
5	INDIVIDUAL_TSP_DEBUG_CDAPI_FORM_MSG_005
6	INDIVIDUAL_TSP_DEBUG_CDAPI_FORM_MSG_006
7	INDIVIDUAL_TSP_DEBUG_CDAPI_SEND_MSG_007
8	INDIVIDUAL_TSP_DEBUG_CDAPI_SEND_MSG_008
9	INDIVIDUAL_TSP_DEBUG_CDAPI_SEND_INFO_MSG_009
10	INDIVIDUAL_TSP_DEBUG_ALLOC_CDB_010
11	INDIVIDUAL_TSP_DEBUG_DEALLOC_CDB_011
12	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_012
13	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_013
14	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_014
15	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_015
16	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_016
17	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_017
18	INDIVIDUAL_TSP_DEBUG_CDAPI_SETUP_ACK_018
19	INDIVIDUAL_TSP_DEBUG_CDAPI_PROCEEDING_019
20	INDIVIDUAL_TSP_DEBUG_CDAPI_ALERT_020
21	INDIVIDUAL_TSP_DEBUG_CDAPI_CONNECT_021
22	INDIVIDUAL_TSP_DEBUG_CDAPI_INFO_022
23	INDIVIDUAL_TSP_DEBUG_CDAPI_PROGRESS_023

Table 5 TSP Individual Debug Values (continued)

Value	TSP Debug Function
24	INDIVIDUAL_TSP_DEBUG_CDAPI_FACILITY_024
25	INDIVIDUAL_TSP_DEBUG_CDAPI_FACILITY_025
26	INDIVIDUAL_TSP_DEBUG_CDAPI_PRE_CONN_DISC_REQ_026
27	INDIVIDUAL_TSP_DEBUG_CDAPI_DISC_PROG_IND_027
28	INDIVIDUAL_TSP_DEBUG_CDAPI_DISCONNECT_REQ_028
29	INDIVIDUAL_TSP_DEBUG_CDAPI_DISCONNECT_REQ_029
30	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_SS_RESP_030
31	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_INFO_IND_031
32	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_PROCEEDING_032
33	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_ALERT_033
34	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_ALERT_EXIT_034
35	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_ALERT_EXIT_035
36	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_PROGRESS_036
37	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_INFO_037
38	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_CONNECT_038
39	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_CONNECT_CONF_039
40	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_DISC_PROG_IND_040
41	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_PROG_IND_PROGRESS_041
42	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_IND_042
43	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_IND_EXIT_043
44	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_COMP_044
45	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_COMP_CLEAR_045
46	INDIVIDUAL_TSP_DEBUG_SETUP_REQ_EXIT_046
47	INDIVIDUAL_TSP_DEBUG_SETUP_REQ_EXIT_047
48	INDIVIDUAL_TSP_DEBUG_SETUP_REQ_EXIT_048
49	INDIVIDUAL_TSP_DEBUG_TSP_SET_TRANSFER_INFO_049
50	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_050
51	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_051
52	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_052
53	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_053
54	INDIVIDUAL_TSP_DEBUG_TSP_MAIN_054
55	INDIVIDUAL_TSP_DEBUG_DO_GLOBAL_END_TO_END_DISC_055
56	INDIVIDUAL_TSP_DEBUG_TSP_CDAPI_MSG_DUMP_056
57	INDIVIDUAL_TSP_DEBUG_TSP_COT_TIMER_START_057
58	INDIVIDUAL_TSP_DEBUG_TSP_COT_TIMER_STOP_058
59	INDIVIDUAL_TSP_DEBUG_TSP_COT_RESULT_059

Table 5 TSP Individual Debug Values (continued)

Value	TSP Debug Function
60	INDIVIDUAL_TSP_DEBUG_TSP_COT_DONE_060
61	INDIVIDUAL_TSP_DEBUG_TSP_COT_TIMEOUT_061
62	INDIVIDUAL_TSP_DEBUG_TSP_COT_REQ_062
63	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_COT_SETUP_ACK_063
64	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RCV_COT_MSG_064
65	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RCV_COT_MSG_065
66	INDIVIDUAL_TSP_DEBUG_TSP_CDAPI_PUT_CAUSE_IE_066
67	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_SETUP_ACK_067
68	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RCV_MSG_068

I

Table 6 VTSP Individual Debug Values

Value	VTSP Debug Function
1	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_PEND_DEFER_001
2	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_WAIT_PEND_SUCCESS_002
3	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_WAIT_PEND_FAIL_003
4	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_COMPLETE_004
5	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_COMPLETE_EXIT_005
6	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_CHECK_006
7	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_CHECK_EXIT_007
8	INDIVIDUAL_VTSP_DEBUG_GENERATE_DISC_008
9	INDIVIDUAL_VTSP_DEBUG_GENERATE_DISC_EXIT_009
10	INDIVIDUAL_VTSP_DEBUG_SETUP_IND_ACK_010
11	INDIVIDUAL_VTSP_DEBUG_SETUP_IND_ACK_EXIT_011
12	INDIVIDUAL_VTSP_DEBUG_PROCEEDING_012
13	INDIVIDUAL_VTSP_DEBUG_PRE_CON_DISCONNECT_013
14	INDIVIDUAL_VTSP_DEBUG_PRE_CON_DISCONNECT_EXIT_014
15	INDIVIDUAL_VTSP_DEBUG_SET_DIGIT_TIMEOUTS_015
16	INDIVIDUAL_VTSP_DEBUG_CONNECT_016
17	INDIVIDUAL_VTSP_DEBUG_LOOPBACK_017
18	INDIVIDUAL_VTSP_DEBUG_RING_NOAN_TIMER_018
19	INDIVIDUAL_VTSP_DEBUG_ALERT_CONNECT_019
20	INDIVIDUAL_VTSP_DEBUG_PRE_CON_DISC_REL_EXIT_020
21	INDIVIDUAL_VTSP_DEBUG_HOST_DISC_CLEANUP_021
22	INDIVIDUAL_VTSP_DEBUG_HOST_DISC_CLEANUP_EXIT_022
23	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_023
24	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_EXIT_024

Table 6 VTSP Individual Debug Values (continued)

Value	VTSP Debug Function
25	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_EXIT_025
26	INDIVIDUAL_VTSP_DEBUG_CONNECT_DIAL_026
27	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_DIAL_027
28	INDIVIDUAL_VTSP_DEBUG_PRE_DISC_CAUSE_028
29	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_CONNECT_029
30	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_PEND_FAIL_030
31	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_DISC_031
32	INDIVIDUAL_VTSP_DEBUG_RELEASE_TIMEOUT_032
33	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROCEEDING_EXIT_033
34	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROCEEDING_EXIT_034
35	INDIVIDUAL_VTSP_DEBUG_PEND_RELEASE_IND_035
36	INDIVIDUAL_VTSP_DEBUG_PEND_RELEASE_IND_EXIT_036
37	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_NO_DSP_CHAN_037
38	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_NO_DSP_CHAN_EXIT_038
39	INDIVIDUAL_VTSP_DEBUG_CALL_FEATURE_IND_039
40	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROGRESS_040
41	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_041
42	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_EXIT_042
43	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_FIRST_PROGRESS_043
44	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_FIRST_PROGRESS_EXIT_044
45	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_FIRST_PROGRESS_EXIT_045
46	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROG_PROCEEDING_046
47	INDIVIDUAL_VTSP_DEBUG_PROCEEDING_R2_PEND_DIAL_047
48	INDIVIDUAL_VTSP_DEBUG_ALERT_R2_PEND_DIAL_048
49	INDIVIDUAL_VTSP_DEBUG_CONN_R2_PEND_DIAL_049
50	INDIVIDUAL_VTSP_DEBUG_SETUP_R2_PEND_DIAL_050
51	INDIVIDUAL_VTSP_DEBUG_R2_PEND_DIAL_ALL_051
52	INDIVIDUAL_VTSP_DEBUG_INFO_IND_052
53	INDIVIDUAL_VTSP_DEBUG_ALERT_053
54	INDIVIDUAL_VTSP_DEBUG_ALERT_EXIT_054
55	INDIVIDUAL_VTSP_DEBUG_PROGRESS_055
56	INDIVIDUAL_VTSP_DEBUG_DISC_PROG_IND_056
57	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_DISC_PI_IND_057
58	INDIVIDUAL_VTSP_DEBUG_INFO_058
59	INDIVIDUAL_VTSP_DEBUG_FEATURE_059
60	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_NO_TIMEOUT_060

Table 6 VTSP Individual Debug Values (continued)

Value	VTSP Debug Function
61	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_NO_TIMEOUT_EXIT_061
62	INDIVIDUAL_VTSP_DEBUG_CALL_FEATURE_ENABLE_062
63	INDIVIDUAL_VTSP_DEBUG_XCCSM_COT_TEST_DONE_063
64	INDIVIDUAL_VTSP_DEBUG_XCCSM_COT_TEST_TIMEOUT_064
65	INDIVIDUAL_VTSP_DEBUG_XCCSM_COT_TEST_065
66	INDIVIDUAL_VTSP_DEBUG_CALL_FEATURE_066
67	INDIVIDUAL_VTSP_DEBUG_TCSM_COT_TEST_DONE_067
68	INDIVIDUAL_VTSP_DEBUG_TCSM_COT_TEST_TIMEOUT_068
69	INDIVIDUAL_VTSP_DEBUG_TCSM_ACT_COT_TEST_069
70	INDIVIDUAL_VTSP_DEBUG_PLAY_BUSY_TIMER_START_070
71	INDIVIDUAL_VTSP_DEBUG_PLAY_BUSY_TIMER_STOP_071
72	INDIVIDUAL_VTSP_DEBUG_RING_NOAN_TIMER_START_072
73	INDIVIDUAL_VTSP_DEBUG_RING_NOAN_TIMER_STOP_073
74	INDIVIDUAL_VTSP_DEBUG_VTSP_TIMER_074
75	INDIVIDUAL_VTSP_DEBUG_VTSP_TIMER_STOP_075
76	INDIVIDUAL_VTSP_DEBUG_VTSP_ALLOCATE_CDB_076
77	INDIVIDUAL_VTSP_DEBUG_VTSP_DO_CALL_SETUP_IND_077
78	INDIVIDUAL_VTSP_DEBUG_VTSP_DO_CALL_SETUP_IND_EXIT_078
79	INDIVIDUAL_VTSP_DEBUG_VTSP_REQUEST_CALL_079
80	INDIVIDUAL_VTSP_DEBUG_VTSP_REQUEST_CALL_EXIT_080
81	INDIVIDUAL_VTSP_DEBUG_VTSP_REALLOC_CDB_081
82	INDIVIDUAL_VTSP_DEBUG_VTSP_OG_CALL_REQ_EXIT_082
83	INDIVIDUAL_VTSP_DEBUG_VTSP_FREE_CDB_083
84	INDIVIDUAL_VTSP_DEBUG_TGRM_DISC_REL_084
85	INDIVIDUAL_VTSP_DEBUG_VTSP_CC_CALL_DISCONNECTED_085
86	INDIVIDUAL_VTSP_DEBUG_SIGO_BDROP_086
87	INDIVIDUAL_VTSP_DEBUG_SIGO_PRE_CON_DISCONNECT_087
88	INDIVIDUAL_VTSP_DEBUG_SIGO_PROCEEDING_088
89	INDIVIDUAL_VTSP_DEBUG_SIGO_GENERATE_DISC_089
90	INDIVIDUAL_VTSP_DEBUG_SIGO_ALERT_090
91	INDIVIDUAL_VTSP_DEBUG_SIGO_ALERT_CONNECT_091
92	INDIVIDUAL_VTSP_DEBUG_SIGO_SETUP_PEND_CONNECT_092
93	INDIVIDUAL_VTSP_DEBUG_DO_SIGO_CALL_SETUP_REQ_093
94	INDIVIDUAL_VTSP_DEBUG_DO_SIGO_CALL_SETUP_REQ_SESSION_094
95	INDIVIDUAL_VTSP_DEBUG_DSM_MEDIA_EVENT_CB_095
96	INDIVIDUAL_VTSP_DEBUG_DSM_PEER_EVENT_CB_096

Table 6 VTSP Individual Debug Values (continued)

Value	VTSP Debug Function
97	INDIVIDUAL_VTSP_DEBUG_DSM_FEATURE_NOTIFY_CB_097
98	INDIVIDUAL_VTSP_DEBUG_DSM_BRIDGE_CHECK_CB_098
99	INDIVIDUAL_VTSP_DEBUG_DSM_BRIDGE_STATUS_EXIT_099
100	INDIVIDUAL_VTSP_DEBUG_DSM_SET_FAX_FEAT_EXIT_100
101	INDIVIDUAL_VTSP_DEBUG_DS_DO_DIAL_101
102	INDIVIDUAL_VTSP_DEBUG_DS_DIALING_DEFAULT_102

Managing the Voice Call Debug Output

The debug output for calls on Cisco voice gateways is managed by using the **voice call debug** command in global configuration mode. When this command is enabled, the standardized header appears when voice debugs are used.

Supported Commands

The voice debug commands that support the standardized header include the following:

- **debug crm**
- **debug fax dmsp**
- **debug fax fmsp**
- **debug fax foip**
- **debug fax mmoip aaa**
- **debug fax mspi**
- **debug fax mta**
- **debug mgcp all**
- **debug mgcp endpoint**
- **debug mgcp endptdb**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp gcfm**
- **debug mgcp inout**
- **debug mgcp media**
- debug mgcp nas
- debug mgcp packets
- debug mgcp parser
- **debug mgcp src**
- **debug mgcp state**

- **debug mgcp voipcac**
- **debug rtsp all**
- **debug rtsp api**
- **debug rtsp client**
- **debug rtsp error**
- **debug rtsp pmh**
- **debug rtsp session**
- **debug rtsp socket**
- **debug tgrm**
- **debug voip application vxml**
- **debug voip avlist**
- **debug voip ccapi**
- **debug voip dialpeer**
- **debug voip dsm**
- **debug voip dspapi**
- **debug voip hpi**
- **debug voip ivr all**
- **debug voip ivr applib**
- **debug voip ivr callsetup**
- **debug voip ivr digitcollect**
- **debug voip ivr dynamic**
- **debug voip ivr error**
- **debug voip ivr script**
- **debug voip ivr settlement**
- **debug voip ivr states**
- **debug voip ivr tclcommands**
- **debug voip profile fax**
- **debug voip profile help**
- debug voip profile modem
- debug voip profile voice
- debug voip rawmsg
- debug voip tsp
- debug voip vtsp
- debug vtsp all
- debug vtsp dsp
- debug vtsp error
- debug vtsp event
- debug vtsp port

- debug vtsp rtp
- debug vtsp send-nse
- debug vtsp session
- debug vtsp stats
- debug vtsp vofr subframe
- debug vtsp tone

For detailed examples of these debug commands, see the *Cisco IOS Debug Command Reference*.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice call debug** {full-guid | short-header}
4. **exit**
5. Run desired debug command.

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

Step 3	`voice call debug {full-guid | short-header}`
	
	Example:
	`Router(config)# voice call debug full-guid`
		Specifies the full GUID or short header for debugging a voice call in a multiple-call environment.
		• **full-guid**—Displays the GUID in a 16-byte header. When the **no** version of this command is input with the **full-guid** keyword, the short 6-byte version displays. This is the default.
		• **short-header**—Displays the CallEntry ID in the header without displaying the GUID or module-specific parameters.

	Command or Action	Purpose

Step 4	`exit`
	
	Example:
	`Router(config)# exit`
		Exits to privileged EXEC mode.

Step 5	Run desired voice debug command. **Example:** `Router# debug voip ccapi inout`	Enter the appropriate debug command for your troubleshooting needs. • The voice debug commands that support the standardized header and detailed examples of the individual debug commands are in the *Cisco IOS Debug Command Reference*, Release 12.3.

Sample Output Examples for the Enhanced debug Commands

The following sections provide examples for the enhanced **debug** commands:

- Full GUID Header: Example, page 22
- Short Header: Example, page 23
- Setup and Teardown: Example, page 23

Full GUID Header: Example

The following is sample output from the **voice call debug** command when the **full-guid** keyword is specified:

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

In the first line, the following parameters are described:

- 1 denotes the CallEntry ID.
- 0E2C8A90-BC00-11D5-8002-DACCFDCEF87D is the GUID.
- VTSP:(0:D):0:0:4385 is the module and its dependent parameters. In this case, the module is VTSP, the port number is 0:D, the channel number is 0, the DSP slot is 0, and the DSP channel number is 4385.
- vtsp_insert_cdb: is the function name.

In the second line, the following parameters are described:

- -1 indicates that the CallEntry ID for the CCAPI module is unavailable.
- xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx indicates that the GUID is unavailable.
- CCAPI is the module.
- cc_incr_if_call_volume: is the function name.

Short Header: Example

The following is sample output from the **voice call debug** command when the **short-header** keyword is specified:

```
Router(config)# voice call debug short-header
!
00:10:12: //-1/cc_incr_if_call_volume:
00:10:12: //9/vtsp_open_voice_and_set_params:
```

In the first line, the CallEntry ID is not available, and in the following line the CallEntry ID is 9. The remainder of the line displays the function name.

Setup and Teardown: Example

In the following example, the **debug voip ccapi inout** command used with the **voice call debug full-guid** 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. This output shows how calls are being handled by the voice gateway. Using this debug level, you can see the call setup and teardown operations performed on both the telephony and network call legs.

```
Router# debug voip ccapi inout

*Mar  1 15:35:53.588: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructTDUsrContainer:
usrContainer[0x638C1BF0], magic[FACE0FFF]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDUtilAddDataToUsrContainer:
container=0x638C1BF0, tagID=6, dataSize=16, instID=-1,modifier=1
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructInstanceTDObject:
tdObject[0x638BC1AC], nxtElem[0x0], magic[0xFACE0FFF] tagID[6], dataLen[16], modif[1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer: Adding
tdObject[0x638BC1AC] instID[-1] into container[0x638C1BF0]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDUtilAddDataToUsrContainer:
container=0x638C1BF0, tagID=5, dataSize=276, instID=-1,modifier=1
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructInstanceTDObject:
tdObject[0x63401148], nxtElem[0x0], magic[0xFACE0FFF] tagID[5], dataLen[276], modif[1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer: Adding
tdObject[0x63401148] instID[-1] into container[0x638C1BF0]
```

In the following lines, the CCAPI receives the call setup. The called number is 34999, and the calling number is 55555. The calling number matches dial peer 10002.

```
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/cc_api_display_ie_subfields:
*Mar  1 15:35:53.592: cc_api_call_setup_ind:
*Mar  1 15:35:53.592:  cisco-username=
*Mar  1 15:35:53.596: ----- ccCallInfo IE subfields -----
*Mar  1 15:35:53.596:  cisco-ani=55555
*Mar  1 15:35:53.596:  cisco-anitype=0
*Mar  1 15:35:53.596:  cisco-aniplan=0
*Mar  1 15:35:53.596:  cisco-anipi=0
*Mar  1 15:35:53.596:  cisco-anisi=0
*Mar  1 15:35:53.596:  dest=34999
*Mar  1 15:35:53.596:  cisco-desttype=0
*Mar  1 15:35:53.596:  cisco-destplan=0
*Mar  1 15:35:53.596:  cisco-rdn=
*Mar  1 15:35:53.596:  cisco-rdntype=-1
*Mar  1 15:35:53.596:  cisco-rdnplan=-1
*Mar  1 15:35:53.596:  cisco-rdnpi=-1
*Mar  1 15:35:53.596:  cisco-rdnsi=-1
*Mar  1 15:35:53.596:  cisco-redirectreason=-1
```

```
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind: (vdbPtr=0x637EC1E0,
callInfo={called=34999,called_oct3=0x80,calling=55555,calling_oct3=0x80,calling_oct3a=0x0,
calling_xlated=false,subscriber_type_str=RegularLine,fdest=1,peer_tag=10002,
prog_ind=0,callingIE_present 1, src_route_label=, tgt_route_label=
clid_transparent=0},callID=0x637B4278)

*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind:
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind: type 13 , prot 0
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: ccCheckClipClir: calling number is: "55555", calling oct3a is: 0x0
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: Calling Party number is User Provided
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: Leaving ccCheckClipClir
  calling number is: "55555"
  calling oct3 is:  0x80
  calling oct3a is: 0x0
```

In the next line, 44 is the CallEntry ID.

```
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: Increment call volume:
0
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: current call volume: 1
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: entry's incoming TRUE.
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: is_incomingis TRUE
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructHashProfileTab:
profileTable[0x6380E11C], numBuckets[11], numEntries[0]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager: Invoking
necessary profileTable updaters...
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer:
Updating profileTable[0x6380E11C] with objects in container[0x638C1BF0]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer:
obtained key[5] for the tag[6]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket:
profileTable[0x6380E11C], tdObject[0x638BC1AC]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer:
obtained key[0] for the tag[5]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket:
profileTable[0x6380E11C], tdObject[0x63401148]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager:
*Mar  1 15:35:53.600: ccTDUtilDumpAllElemInProfileTab: profileTable[0x6380E11C],
 numBuckets[11], numEntries[2]
*Mar  1 15:35:53.600: Bucket { 0 } ------>0x63401148[0x0,t-5,l-276,d-0x63401168,
m-1,u-56153,g-FACE0FFF]
*Mar  1 15:35:53.604:
*Mar  1 15:35:53.604: Bucket { 5 }
------>0x638BC1AC[0x0,t-6,l-16,d-0x638BC1CC,m-1,u-56153,g-FACE0FFF]
*Mar  1 15:35:53.604:
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructTDUsrContainer:
Container[0x638C1BF0]
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: not the VoIP or
MMoIP
*Mar  1 15:35:53.608: //-1/xxxxxxxxxxxx/CCAPI/cc_process_call_setup_ind: (event=
0x63073AA0)
```

In the next line, 45F2AAE28044 is the GUID. The tag 10002 entry shows that the incoming dial peer matched the CallEntry ID.

```
*Mar  1 15:35:53.608: //44/45F2AAE28044/CCAPI/cc_process_call_setup_ind: >>>>CCAPI handed
cid 44 with tag 10002 to app "DEFAULT"
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(24=CC_EV_CALL_SETUP_IND), cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(SSA_EV_CALL_SETUP_IND),
cid(44), disp(0)
```

```
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/ssaCallSetupInd:
```

The next line shows CallEntry ID in hexadecimal form, 0x2C (44 in decimal). The CallEntry ID and GUID numbers have been identified. The incoming dial peer is 10002.

```
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2C,
context=0x634A430C)

*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: cid(44),
st(SSA_CS_MAPPING),oldst(0), ev(24)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: src route label=,
tgt route label= tg_label_flag 0x0
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: finalDest
cllng(55555), clled(34999) tgt_route_label()tg_label_flag 0x0
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: cid(44),
st(SSA_CS_CALL_SETTING),oldst(0), ev(24)dpMatchPeersMoreArg result= 0
```

For CallEntry ID 44, two dial-peer tags (10001 and 20002) were matched with called number 34999.

```
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaDebugPeers: ssaSetupPeer cid(44)
peer list: tag(10001) called number (34999) tag(20002) called number
(34999)
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaSetupPeer: dialpeer tags in
rotary= 10001  20002
```

The next line shows that five digits were matched for this dial peer and no prefix was added. The encapType (2) entry indicates a VoIP call.

```
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaSetupPeer: cid(44),
destPat(34999), matched(5), prefix(), peer(637B0984), peer->encapType (2)
*Mar  1 15:35:53.612: //-1/xxxxxxxxxxxx/CCAPI/cc_can_gateway: Call legs: In=6, Out=1
```

The next line shows the voice gateway sending out a call-proceeding message to the incoming call leg with a progress indicator of 0x0.

```
*Mar  1 15:35:53.612: //44/xxxxxxxxxxxx/CCAPI/ccCallProceeding: (callID=0x2C,
prog_ind=0x0)
```

The next line shows the voice gateway sending out the call-setup request to the outgoing call leg. The dial peer is 10001 with the incoming CallEntry ID being 0x2C.

```
*Mar  1 15:35:53.612: //44/xxxxxxxxxxxx/CCAPI/ccCallSetupRequest: (Inbound call= 0x2C,
outbound peer =10001, dest=,
        params=0x63085D80 mode=0, *callID=0x63086314, prog_ind = 0callingIE_present 1)

*Mar  1 15:35:53.612: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.612: ccCallSetupRequest numbering_type 0x80
*Mar  1 15:35:53.612: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.616: ccCallSetupRequest: calling number is:55555

*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: calling oct3ais:0x0

*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: ccCheckClipClir: calling number is: "55555", calling oct3a is: 0x0
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: Calling Party number is User Provided
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: Leaving ccCheckClipClir
  calling number is: "55555"
  calling oct3 is:  0x80
  calling oct3a is: 0x0
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: after ccCheckClipClir -
calling oct3a is:0x0
```

The next line shows that all digits are passed.

```
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: dest pattern 34999,
called 34999, digit_strip 0
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.616: callingNumber=55555, calledNumber=34999, redirectNumber=
display_info= calling_oct3a=0
*Mar  1 15:35:53.616: accountNumber=,
finalDestFlag=1,guid=45f2.aae2.1571.11cc.8044.95f5.fabb.6b0f
*Mar  1 15:35:53.616: peer_tag=10001
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/cc_api_display_ie_subfields:
*Mar  1 15:35:53.616: ccCallSetupRequest:
*Mar  1 15:35:53.616:  cisco-username=
*Mar  1 15:35:53.616: ----- ccCallInfo IE subfields -----
*Mar  1 15:35:53.616:  cisco-ani=55555
*Mar  1 15:35:53.616:  cisco-anitype=0
*Mar  1 15:35:53.616:  cisco-aniplan=0
*Mar  1 15:35:53.616:  cisco-anipi=0
*Mar  1 15:35:53.616:  cisco-anisi=0
*Mar  1 15:35:53.620:  dest=34999
*Mar  1 15:35:53.620:  cisco-desttype=0
*Mar  1 15:35:53.620:  cisco-destplan=0
*Mar  1 15:35:53.620:  cisco-rdn=
*Mar  1 15:35:53.620:  cisco-rdntype=-1
*Mar  1 15:35:53.620:  cisco-rdnplan=-1
*Mar  1 15:35:53.620:  cisco-rdnpi=-1
*Mar  1 15:35:53.620:  cisco-rdnsi=-1
*Mar  1 15:35:53.620:  cisco-redirectreason=-1

*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
(vdbPtr=0x62EC61A4, dest=, callParams={called=34999,called_oct3=0x80,
calling=55555,calling_oct3=0x80, calling_oct3a= 0x0, calling_xlated=false,
subscriber_type_str=RegularLine, fdest=1, voice_peer_tag=10001},mode=0x0)
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 15:35:53.620: ccIFCallSetupRequestPrivate: src route label  tgt route label
tg_label_flag 0x0
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:  vdbPtr type =
1
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
(vdbPtr=0x62EC61A4, dest=, callParams={called=34999, called_oct3 0x80,  calling=55555,
calling_oct3 0x80, calling_oct3a 0x0, calling_xlated=false,  fdest=1,
voice_peer_tag=10001}, mode=0x0, xltrc=-5)
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
```

In the next line, outgoing CallEntry ID 45 is bound to the same GUID 45F2AAE28044.

```
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: not incoming entry
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: entry's incoming
FALSE.
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: is_incomingis FALSE
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/ccSaveDialpeerTag: (callID=0x2C,
dialpeer_tag=10001)
*Mar  1 15:35:53.624: //45/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2D,
context=0x634A537C) 0x2D (decimal 45 is the second call leg ID).
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/ccCallReportDigits: (callID=0x2C,enable=0x0)
```

The next line shows that the voice gateway informs the incoming call leg that digits were forwarded.

```
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/cc_api_call_report_digits_done:
(vdbPtr=0x637EC1E0, callID=0x2C, disp=0)
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(54=CC_EV_CALL_REPORT_DIGITS_DONE), cid(44), disp(0)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SS
```

```
Router#APP:10002:-1/ssaTraceSct:
cid(44)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_REPORT_DIGITS_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaTraceSct:
-cid2(45)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaDebugPeers: ssaReportDigitsDone
cid(44) peer list: tag(20002) called number (34999)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaReportDigitsDone: callid=44
Reporting disabled.
*Mar  1 15:35:53.628: //-1/xxxxxxxxxxxx/CCAPI/cc_api_supported_data: data_mode=0x10082
*Mar  1 15:35:53.628: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_ic_leg_obtained_numbers:
callID=0x2D
```

The next two lines show the IP address of the terminating gateway and indicate that the terminating gateway is reached through Ethernet port 0/0.

```
*Mar  1 15:35:53.628: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: remote IP is
172.31.85.111
*Mar  1 15:35:53.632: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: hwidb is Ethernet0/0
*Mar  1 15:35:53.632: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: create entry in
list: 1
*Mar  1 15:35:53.636: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetInstanceCount: For tagID[1] of
callID[45]
*Mar  1 15:35:53.636: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessManager: No
profileTable set for callID[45]
*Mar  1 15:35:53.636: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetInstanceCount: For tagID[2] of
callID[45]
*Mar  1 15:35:53.636: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessManager: No
profileTable set for callID[45]
```

The next line shows that the voice gateway received a call proceeding message from the terminating gateway. The following line shows that the voice gateway received a call alert from the terminating gateway.

```
*Mar  1 15:35:53.740: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_proceeding: (vdbPtr=0x62EC61A4,
callID=0x2D,
      prog_ind=0x0)
*Mar  1 15:35:53.740: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_alert: (vdbPtr=0x62EC61A4,
callID=0x2D, prog_ind=0x0, sig_ind=0x1)
*Mar  1 15:35:53.744: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(21=CC_EV_CALL_PROCEEDING), cid(45), disp(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
cid(45)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_PROCEEDING)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
-cid2(44)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaCallProc:
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaIgnore: cid(45),
st(SSA_CS_CALL_SETTING),oldst(1), ev(21)
*Mar  1 15:35:53.744: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(7=CC_EV_CALL_ALERT),
cid(45), disp(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
cid(45)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_ALERT)oldst(SSA_CS_CALL_SETTING)cfid(-1)csize
(0)in(0)fDest(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
-cid2(44)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:35:53.744: //44/45F2AAE28044/SSAPP:10002:-1/ssaAlert:
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
Router#
```

In the following line, the voice gateway forwarded a call alert to the originating gateway.

```
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccCallAlert: (callID=0x2C, prog_ind=0x0,
sig_ind=0x1)
Router#
```

The phone is answered at the called number.

```
Router#!call answered
```

The voice gateway receives a connect message from the terminating gateway.

```
*Mar  1 15:36:05.016: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_connected: (vdbPtr=0x62EC61A4,
callID=0x2D), prog_ind = 0
*Mar  1 15:36:05.016: //45/45F2AAE28044/CCAPI/cc_api_call_connected: setting
callEntry->connected to TRUE
```

The next line shows that the call accounting starts. The leg_type=False message means this is for an outgoing call. The line that follows shows that AAA accounting is not configured.

```
*Mar  1 15:36:05.016: //45/45F2AAE28044/CCAPI/cc_api_call_connected: calling accounting
start for callID=45 leg_type=0
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/CCAPI/ccCallSetAAA_Accounting: callID=0x2D,
accounting=0
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(8=CC_EV_CALL_CONNECTED),
cid(45), disp(0)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
cid(45)st(SSA_CS_ALERT_RCVD)ev(SSA_EV_CALL_CONNECTED)
oldst(SSA_CS_CALL_SETTING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
-cid2(44)st2(SSA_CS_ALERT_RCVD)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaConnect:
*Mar  1 15:36:05.020: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
```

The next lines show a conference being set up between the two call legs 0x2C and 0x2D. Bridge complete messages are sent to both the terminating and originating gateways.

```
*Mar  1 15:36:05.020: //44/xxxxxxxxxxxx/CCAPI/ccConferenceCreate: (confID=0x63086424,
callID1=0x2C, callID2=0x2D, tag=0x0)
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done:
(confID=0x15,srcIF=0x62EC61A4, srcCallID=0x2D, dstCallID=0x2C, disposition=0, tag=0x0)
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done:
(confID=0x15,srcIF=0x637EC1E0, srcCallID=0x2C, dstCallID=0x2D, disposition=0, tag=0x0)
```

Here, the voice gateway sets up negotiating capability with the originating telephony leg.

```
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (dstVdbPtr=0x62EC61A4,
dstCallId=0x2D, srcCallId=0x2C,
     caps={codec=0x2887F, fax_rate=0xBF, vad=0x3, modem=0x2
           codec_bytes=0, signal_type=3})
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (Playout: mode 0, initial
60,min 40, max 300)
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(29=CC_EV_CONF_CREATE_DONE), cid(44), disp(0)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
cid(44)st(SSA_CS_CONFERENCING)ev(SSA_EV_CONF_CREATE_DONE)
oldst(SSA_CS_CALL_SETTING)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
-cid2(45)st2(SSA_CS_CONFERENCING)oldst2(SSA_CS_ALERT_RCVD)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaConfCreateDone:
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/ccCallConnect: (callID=0x2C), prog_ind = 0
*Mar  1 15:36:05.024: //44/45F2AAE28044/CCAPI/ccCallConnect: setting callEntry->connected
to TRUE
```

```
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaDebugPeers: ssaFlushPeerTagQueue
cid(44) peer list: tag(20002) called number (34999)
*Mar  1 15:36:05.028: //-1/xxxxxxxxxxxx/CCAPI/cc_process_notify_bridge_done:
(event=0x63067FC0)
```

The voice gateway sets up negotiating capability with the terminating VoIP leg.

```
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (dstVdbPtr=0x637EC1E0,
dstCallId=0x2C, srcCallId=0x2D,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2})
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (Playout: mode 0, initial
60,min 40, max 300)
```

The capabilities are acknowledged for both call legs.

```
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ack: (dstVdbPtr=0x637EC1E0,
dstCallId=0x2C, srcCallId=0x2D,
    caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2, seq_num_start=2944})
*Mar  1 15:36:05.028: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ack: (dstVdbPtr=0x62EC
61A4, dstCallId=0x2D, srcCallId=0x2C,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2, seq_num_start=2944})

*Mar  1 15:36:05.032: //44/xxxxxxxxxxxx/CCAPI/cc_api_voice_mode_event: callID=0x2C
*Mar  1 15:36:05.032: //44/45F2AAE28044/CCAPI/cc_api_voice_mode_event: Call Pointer
=634A430C
*Mar  1 15:36:05.032: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(52=CC_EV_VOICE_MODE_DONE), cid(44), disp(0)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
Router#
Router#cid(44)st(SSA_CS_ACTIVE)ev(SSA_EV_VOICE_MODE_DONE)oldst(SSA_CS_CONFERENCING)cfid(21
)csize(2)in(1)fDest(1)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
-cid2(45)st2(SSA_CS_ACTIVE)oldst2(SSA_CS_ALERT_RCVD)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaIgnore: cid(44),
st(SSA_CS_ACTIVE),oldst(5), ev(52)
Router#
Router#! digit punched
Router#
```

The phone at the terminating gateway enters digit 1.

```
*Mar  1 15:36:11.204: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPt
r=0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=1, digit_begin_flags=0x0, rtp_timestamp=0x0
    rtp_expiration=0x0, dest_mask=0x2)
*Mar  1 15:36:11.504: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_end:
(dstVdbPtr=0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=1,duration=300,xruleCallingTag=0,xruleCalledTag=0, dest_mask=0x2),
digit_tone_mode=0
```

The phone at the terminating gateway enters digit 2.

```
*Mar  1 15:36:11.604: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin:
(dstVdbPtr=0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=2, digit_begin_flags=0x0, rtp_timestamp=0x0
    rtp_expiration=0x0, dest_mask=0x2)
*Mar  1 15:36:11.904: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_end: (dstVdbPtr=
0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=2,duration=300,xruleCallingTag=0,xruleCalledTag=0, dest_mask=0x2),
digit_tone_mode=0
Router#
```

```
*Mar  1 15:36:14.476: //-1/xxxxxxxxxxxx/CCAPI/cc_handle_periodic_timer: Calling the
callback, ccTimerctx - 0x628B6330
*Mar  1 15:36:14.476: //-1/xxxxxxxxxxxx/CCAPI/ccTimerStart: ccTimerctx - 0x628B6330
Router#
Router#!call hung up  The user at the terminating gateway hangs up the call.
Router#
```

The voice gateway receives a disconnect message from the terminating gateway. The cause code is 0x10 which is normal call clearing.

```
*Mar  1 15:36:22.916: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnected: (vdbPtr=
0x62EC61A4, callID=0x2D, cause=0x10)
*Mar  1 15:36:22.920: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(11=CC_EV_CALL_DISCONNECTED), cid(45), disp(0)
*Mar  1 15:36:22.920: //45/45F2AAE28044/SSAPP:0:21/ssaTraceSct:
cid(45)st(SSA_CS_ACTIVE)ev(SSA_EV_CALL_DISCONNECTED)
oldst(SSA_CS_ALERT_RCVD)cfid(21)csize(2)in(0)fDest(0)
*Mar  1 15:36:22.920: //45/45F2AAE28044/SSAPP:0:21/ssaTraceSct:
-cid2(44)st2(SSA_CS_ACTIVE)oldst2(SSA_CS_ACTIVE)
*Mar  1 15:36:22.920: ssa: Disconnected cid(45) state(5) cause(0x10)
```

The voice gateway begins tearing down the conference and dropping the bridge.

```
*Mar  1 15:36:22.920: //-1/xxxxxxxxxxxx/CCAPI/ccConferenceDestroy: (confID=0x15, tag=0x0)
*Mar  1 15:36:22.920: //45/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done: (confID=0x15,
srcIF=0x62EC61A4, srcCallID=0x2D, dstCallID=0x2C, disposition=0 tag=0x0)
*Mar  1 15:36:22.920: //44/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done: (confID=0x15,
srcIF=0x637EC1E0, srcCallID=0x2C, dstCallID=0x2D, disposition=0 tag=0x0)
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(30=CC_EV_CONF_DESTROY_DONE), cid(44), disp(0)
*Mar  1 15:36:22.924: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
cid(44)st(SSA_CS_CONF_DESTROYING)ev(SSA_EV_CONF_DESTROY_DONE)
oldst(SSA_CS_ACTIVE)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:22.924: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_CONF_DESTROYING)oldst2(SSA_CS_ACTIVE)
*Mar  1 15:36:22.924: //45/45F2AAE28044/SSAPP:0:-1/ssaConfDestroyDone:
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/CCAPI/ccCallDisconnect: (callID=0x2C, cause=0x10
tag=0x0)
```

The voice gateway stops call accounting on the incoming call, indicated by the leg_type=True message. The cause code is then set for the originating leg.

```
*Mar  1 15:36:22.924: //44/45F2AAE28044/CCAPI/ccCallDisconnect: calling accounting start
for callID=44 leg_type=1
*Mar  1 15:36:22.924: //44/45F2AAE28044/CCAPI/ccCallDisconnect: existing_cause = 0x0,
new_cause = 0x10
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/CCAPI/cc_api_get_transfer_info: (callID=0x2C)
*Mar  1 15:36:22.924: //45/xxxxxxxxxxxx/CCAPI/ccCallDisconnect: (callID=0x2D, cause=0x10
tag=0x0)
```

The voice gateway stops call accounting for the outgoing call, indicated by the leg_type=False message. The cause code is verified for the terminating leg.

```
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: calling accounting start
for callID=45 leg_type=0
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: existing_cause = 0x10,
new_cause = 0x10
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: using the existing_cause
0x10
*Mar  1 15:36:22.928: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_transfer_info: (callID=0x2D)
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_api_icpif: expect factor = 0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/g113_calculate_impairment: (delay=79,
      loss=0), Io=0 Iq=0 Idte=0 Idd=0 Ie=10 Itot=10
```

```
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: the remote IP is
171.69.85.111
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: hwidb is Ethernet0/0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: reduce callnum of
entry: 0, voip: 0, mmoip: 0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: remove anentry
*Mar  1 15:36:22.932: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnect_done:
(vdbPtr=0x62EC61A4, callID=0x2D, disp=0, tag=0x0)
*Mar  1 15:36:22.932: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessManager: No
profileTable set for callID[45]
*Mar  1 15:36:22.936: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetDataByRef: No tdObjectfound in
profileTable for tagID[6] of callID[45]
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: not incoming entry
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: entry's incoming
FALSE.
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: is_incomingis FALSE
*Mar  1 15:36:22.940: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(12=CC_EV_CALL_DISCONNECT_DONE), cid(45), disp(0)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)oldst(SSA_CS_ACTIVE)cfid(-1)csize(2)in(0)fDe
st(0)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
-cid2(44)st2(SSA_CS_DISCONNECTING)oldst2(SSA_CS_CONF_DESTROYING)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaDisconnectDone:
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaAAA_CheckAccounting: accounting
generation enabled
*Mar  1 15:36:22.940: //45/xxxxxxxxxxxx/CCAPI/ccCallSetAAA_Accounting: callID=0x2D,
accounting=0
*Mar  1 15:36:22.944: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: not the VoIP or
MMoIP
*Mar  1 15:36:22.948: //44/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnect_done:
(vdbPtr=0x637EC1E0, callID=0x2C, disp=0, tag=0x0)
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry:
ccFreeRawMsgInfo(0x6307595C)
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: Decrement call volume
counter 1
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: current call volume: 0
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: entry's incoming TRUE.
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: is_incomingis TRUE
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: Deleting
profileTable[0x6380E11C]
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructTDHashProfileTab: Destructor
Profile Table (0x6380E11C)
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructInstanceTDObject:
tdObject[0x63401148] tagID[5]
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructInstanceTDObject:
tdObject[0x638BC1AC] tagID[6]
*Mar  1 15:36:22.956: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl:
ev(12=CC_EV_CALL_DISCONNECT_DONE), cid(44), disp(0)
*Mar  1 15:36:22.956: //44/45F2AAE28044/SSAPP:10002:-1/ssaTraceSct:
cid(44)st(SSA_CS_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)oldst(SSA_CS_CONF_DESTROYING
)cfid(-1)csize(1)in(1)fDest(1)
Router#
*Mar  1 15:36:22.956: //44/45F2AAE28044/SSAPP:10002:-1/ssaDisconnectDone:
```

Filtering Troubleshooting Output

Revised: October 31, 2005

This chapter describes how to filter troubleshooting output. The methods for filtering this output are as follows:

Filtering Output from 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 vertical line 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 functions, see the "Using the Command-Line Interface" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*, Release 12.3.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Voice Call Debug Filtering on Cisco Voice Gateways

Use voice call debug filtering to get selected debugging traces for voice calls. This feature allows you to filter and trace voice call debug messages based on selected filtering criteria, reducing the volume of output for more efficient troubleshooting.

This section contains the following information:

Restrictions for Voice Call Debug Filtering

- End-to-end filtering between gateways is not supported.
- Filtering for CAS, IOS-AAA, IVR Version 1.0, media, and VoiceXML is not supported.
- Matching conditions cannot be set for specific signaling protocols.
- Matching conditions based on current DSP information are not supported.

Information About Voice Call Debug Filtering

Information from using debug commands for voice calls is crucial for troubleshooting, but the volume of raw data can be very large. In order to isolate the most valuable data, use the Voice Call Debug Filtering feature. This feature allows the debug output for the voice call to be filtered according to a variety of criteria, including:

- Calling party number with prefix
- Called party number with prefix
- Carrier IDs
- Dial peers
- Local IP address
- Remote IP address
- Telephony interface or port
- Trunk groups

Note Call filtering also works on IP-to-IP gateway connections using H.323.

The selected criteria are set on the gateway, and different sets of criteria can be stored.

To better understand the voice call debug filtering on Cisco voice gateways, see the following sections:

- Exact and Partial Matching, page 6
- Media and Signaling Streams, page 7

Debug Commands that Support Voice Call Filtering

When a call filter is applied, the filtering applies to all of the debugs affected by the call filter. Debug commands that support voice call debug filtering include the following:

- **debug cch323 h225**
- **debug cch323 h245**
- **debug cch323 preauth**
- **debug cch323 session**
- **debug ccsip all**
- **debug ccsip calls**
- **debug ccsip err**
- **debug ccsip events**
- **debug ccsip messages**
- **debug ccsip preauth**
- **debug ccsip states**
- **debug mgcp all**
- **debug mgcp endpoint**
- **debug mgcp endptdb**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp gcfm**
- **debug mgcp inout**
- **debug mgcp media**
- **debug mgcp src**
- **debug mgcp state**
- **debug mgcp voipcac**
- **debug voip aaa**
- **debug voip ccapi error**
- **debug voip ccapi inout**
- **debug voip ipipgw**
- **debug voip ivr all**
- **debug voip ivr applib**
- **debug voip ivr callsetup**
- **debug voip ivr digitcollect**
- **debug voip ivr dynamic**
- **debug voip ivr error**

- **debug voip ivr script**
- **debug voip ivr settlement**
- **debug voip ivr states**
- **debug voip ivr tclcommands**
- **debug voip rawmsg**
- **debug vtsp all**
- **debug vtsp dsp**
- **debug vtsp error**
- **debug vtsp event**
- **debug vtsp port**
- **debug vtsp rtp**
- **debug vtsp send-nse**
- **debug vtsp session**
- **debug vtsp stats**
- **debug vtsp vofr subframe**
- **debug vtsp tone**
- **debug vtsp vofr**

Note See the *Cisco IOS Debug Command Reference* for detailed information about these debug commands.

Generic Call Filter Module

The debug commands described in the "Debug Commands that Support Voice Call Filtering" section on page 3 support the following voice modules within the voice gateway:

- CCAPI
- Dial peers
- H.323
- ISDN
- IVR (Version 2.0 only)
- MGCP
- SIP
- SSAPP
- TGRAM
- Voice AAA
- VTSP

The filtering for these modules is managed by the generic call filter module (GCFM). The filtering conditions are configured in the GCFM, and then the individual modules are informed when a call has to be filtered. The GCFM coordinates between multiple modules to handle filtering conditions.

All modules use the global unique identifier (GUID) to identify an individual call to GCFM. Each call is assigned a GUID and retains the same GUID throughout the entire network and over time. Gateway information and time stamp are embedded in the GUID. GUIDs identify an individual call among the multiple filtered-out calls so that the call can be isolated. For more information about GUIDs and the debug header, see the "Voice Debug Concepts" section on page 17.

Activity in the GCFM can be traced using the **debug call filter detail** and **debug call filter inout** commands. See the *Cisco IOS Debug Command Reference* for more information about these debug commands.

Calling and Called Number Strings

The string pattern for calling and called numbers can be either a complete telephone number or a partial telephone number with wildcard digits, represented by a period (.) character. Each "." represents a wildcard for an individual digit that the originating voice gateway expects to match. For example, if the calling and called number strings is defined as "555. . . .", then any dialed string beginning with 555, plus at least four additional digits, matches this calling or called number.

Table 7 shows all of the wildcard symbols that are supported in the calling and called number strings.

Table 7 Symbols Used in Calling and Called Number Strings

Symbol	Description
.	Indicates a single-digit placeholder. For example, 555. . . . matches any dialed string beginning with 555, plus at least four additional digits.
[ ]	Indicates a range of digits. A consecutive range is indicated with a hyphen (-); for example, [5-7]. A nonconsecutive range is indicated with a comma (,); for example, [5,8]. Hyphens and commas can be used in combination; for example, [5-7,9]. **Note** Only single-digit ranges are supported. For example, [98-102] is invalid.
( )	Indicates a pattern; for example, 408(555). It is used in conjunction with the symbol ?, %, or +.
?	Indicates that the preceding digit occurred zero or one time. Enter **ctrl-v** before entering ? from your keyboard.
%	Indicates that the preceding digit occurred zero or more times. This functions the same as the "*" used in regular expression.
+	Indicates that the preceding digit occurred one or more times.
T	Indicates the interdigit timeout. The voice gateway pauses to collect additional dialed digits.

Note The period (.) is the only wildcard character that is supported for dial strings that are configured using the **answer-address** or **incoming called-number** command.

Table 8 shows some examples of how these wildcard symbols are applied to the calling and called number strings and the dial string that results when dial string 4085550199 is matched to the calling or called number. The wildcard symbols follow regular expression rules.

Table 8 Number Matching Examples Using Wildcard Symbols

Destination Pattern	Dial String Translation	String After Stripping[1]
408555.+	408555, followed by one or more wildcard digits. This pattern implies that the string must contain at least seven digits starting with 408555.	0199
408555.%	408555, followed by zero or more wildcard digits. This pattern implies that the string must contain at least 408555.	0199
408555+	40855, followed by 5 repeated one or more times.	0199
408555%	40855, followed by 5 repeated zero or more times. Any explicitly matching digit before the % symbol is not stripped off.	50199
408555?	40855, followed by 5 repeated zero or one time. Any explicitly matching digit before the ? symbol is not stripped off.	50199
40855[5-7].+	40855, followed by 5, 6, or 7, plus any digit repeated one or more times.	50199
40855[5-7].%	40855, followed by 5, 6, or 7, plus any digit repeated zero or more times.	50199
40855[5-7]+0199	40855, followed by 5, 6, or 7 repeated one or more times, followed by 0199.	50199
408(555)+0199	408, followed by 555, which may repeat one or more times, followed by 0199.	5550199

1. These examples apply only to one-stage dialing, where DID is enabled on the inbound POTS dial peer. If the voice gateway is using two-stage dialing and collecting digits one at a time as dialed, then the call is routed immediately after a dial peer is matched and any subsequent dialed digits are lost.

In addition to wildcard characters, the following symbols can be used in the calling and called number strings:

- Asterisk (*) and pound sign (#)—These symbols on standard touchtone dial pads can be used anywhere in the pattern. They can be used as the leading character (for example, *650), except on the Cisco 3600 series.
- Dollar sign ($)—Disables variable-length matching. It must be used at the end of the dial string.

Exact and Partial Matching

The conditions under each set of call filters are inclusive, so if multiple conditions are specified under a filter, they are all matched. To compare different conditions, create additional filters.

Matching conditions are as follows:

- Exact match—All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise.
- Partial match—No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because of the large amount of debug output that might be generated before matches explicitly fail.

Media and Signaling Streams

Media streams carry voice, video, fax, and data. Examples of media streams are G.711 or G.723 encoded voice streams or fax data. With the voice call debug filter, the media streams are traced for the voice gateway receiving the media stream. Some traces associated with media streams can be filtered, such as SPI-level traces associated with opening and closing the media channels. However, media RTP/RTCP packet-level traces are not filtered.

Signaling streams include both address signaling and supervisory signaling. Examples of signaling streams include H.323 and SIP protocol streams. With the voice call debug filter, the signaling streams are traced for the gateway or endpoint for the signaling stream.

Configuring the Voice Call Debug Filter

To configure the voice call debug filter, perform the following tasks:

- Configuring Call-Specific Conditions, page 7 (required)
- Enabling Debug for the Set Filtering Conditions, page 11 (required)

Configuring Call-Specific Conditions

Configure call-specific conditions to set the attributes that are filtered for voice calls.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call filter match-list** *number* **voice**
4. **incoming calling-number** *string*
5. **incoming called-number** *string*
6. **incoming secondary-called-number** *string*
7. **incoming port** *string*
8. **incoming signaling** {**local** | **remote**} **ipv4** *ip_address*
9. **incoming media** {**local** | **remote**} **ipv4** *ip_address*
10. **incoming dialpeer** *tag*
11. **source carrier-id** *string*
12. **source trunk-group-label** *group-number*
13. **outgoing calling-number** *string*
14. **outgoing called-number** *string*
15. outgoing secondary-called-number *string*
16. **outgoing port** *string*
17. **outgoing signaling** {**local** | **remote**} **ipv4** *ip_address*
18. **outgoing media** {**local** | **remote**} **ipv4** *ip_address*
19. **outgoing dialpeer** *tag*

20. **target carrier-id** *string*
21. **target trunk-group-label** *group-number*
22. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**call filter match-list** *number* **voice** **Example:** `Router(config)# call filter match-list 1 voice`	Enters call filter match list configuration mode to define the filter conditions. • *number*—Numeric label that uniquely identifies the match list. Range is 1 to 16. **Note** At least one of the following optional parameters (Step 4 to Step 21) for call filtering must be configured.
Step 4	**incoming calling-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming calling-number 408555`	(Optional) Specifies the incoming calling number to be filtered. • *string*—Numeric string that identifies all or part of the incoming calling number.
Step 5	**incoming called-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming called-number 408555`	(Optional) Specifies the incoming called number to be filtered. • *string*—Numeric string that identifies all or part of the incoming called number.
Step 6	**incoming secondary-called-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming secondary-called-number 408555`	(Optional) Specifies the incoming secondary called number to be filtered. • The secondary called number is the number from the second stage in a two-stage scenario. • *string*—Numeric string that identifies all or part of the incoming secondary called number.
Step 7	**incoming port** *string* **Example:** `Router(conf-call-filter-mlist)# incoming port 1/0/0`	(Optional) Specifies the incoming port to be filtered. • The telephony interfaces are defined for calls from the PSTN. The string value varies depending on the voice gateway. • *string*—Identifies the incoming port number. This value is platform-specific and varies between platforms.

	Command or Action	Purpose
Step 8	**incoming signaling** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# incoming signaling local ipv4 192.168.10.255`	(Optional) Specifies the incoming signaling IPv4 address. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 9	**incoming media** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# incoming media local ipv4 192.168.10.255`	(Optional) Specifies the incoming media IPv4 address. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 10	**incoming dialpeer** *tag* **Example:** `Router(conf-call-filter-mlist)# incoming dialpeer 14`	(Optional) Specifies the incoming dial peer to be filtered. • *tag*—Digits that define a specific dial peer. Valid entries are 1 to 2147483647.
Step 11	**source carrier-id** *string* **Example:** `Router(conf-call-filter-mlist)# source carrier-id 4321`	(Optional) Specifies the source carrier ID to be filtered. • *string*—Alphanumeric identifier for the carrier ID.
Step 12	**source trunk-group-label** *group-number* **Example:** `Router(conf-call-filter-mlist)# source trunk-group-label 20`	(Optional) Specifies the source trunk group to be filtered. • *group-number*—A value from 0 to 23 that identifies the trunk group.
Step 13	**outgoing calling-number** *string* **Example:** `Router(conf-call-filter-mlist)# outgoing calling-number 408525`	(Optional) Specifies the outgoing calling number to be filtered. • This number goes out after number translation and expansion are complete. • *string*—Numeric string that identifies all or part of the outgoing calling number.
Step 14	**outgoing called-number** *string* **Example:** `Router(conf-call-filter-mlist)# outgoing called-number 408525`	(Optional) Specifies the outgoing called number to be filtered. • This number goes out after number translation and expansion are complete. • *string*—Numeric string that identifies all or part of the outgoing called number.
Step 15	**outgoing secondary-called-number** *string* **Example:** `Router(conf-call-filter-mlist)# outgoing secondary-called-number 408525`	(Optional) Specifies the outgoing secondary called number to be filtered. • The secondary called number is the number from the second stage in a two-stage scenario. • *string*—Numeric string that identifies all or part of the outgoing secondary called number.

	Command or Action	Purpose
Step 16	**outgoing port** *string* **Example:** `Router(conf-call-filter-mlist)# outgoing port 1/0/0`	(Optional) Specifies the outgoing port to be filtered. • The telephony interfaces are defined for calls from PSTN. The string value varies, depending on different voice gateways. • *string*—Identifies the outgoing port number. This value is voice gateway-specific and varies between voice gateways.
Step 17	**outgoing signaling** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# outgoing signaling local ipv4 192.168.10.255`	(Optional) Specifies the outgoing signaling IPv4 address for the gatekeeper managing the signaling. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 18	**outgoing media** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# outgoing media local ipv4 192.168.10.255`	(Optional) Specifies the outgoing media IPv4 address for the voice gateway receiving the media stream. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 19	**outgoing dialpeer** *tag* **Example:** `Router(conf-call-filter-mlist)# outgoing dialpeer 14`	(Optional) Specifies the outgoing dial peer to be filtered. • *tag*—Digits that define a specific dial peer. Valid entries are 1 to 2147483647.
Step 20	**target carrier-id** *string* **Example:** `Router(conf-call-filter-mlist)# target carrier-id 4321`	(Optional) Specifies the target carrier ID to be filtered. • *string*—Alphanumeric identifier for the carrier ID.
Step 21	**target trunk-group-label** *group-number* **Example:** `Router(conf-call-filter-mlist)# target trunk-group-label 20`	(Optional) Specifies the target trunk group to be filtered. • *group-number*—A value from 0 to 23 that identifies the trunk group.
Step 22	**end** **Example:** `Router(conf-call-filter-mlist)# end`	Exits to privileged EXEC mode.

Troubleshooting Tips

To verify the conditions that you have set, use the **show call filter match-list** command. This command displays the criteria set for the specified match list.

What to Do Next

After the conditions are set for the voice call debug, debug commands can be enabled. Proceed to the "Enabling Debug for the Set Filtering Conditions" section on page 11.

Enabling Debug for the Set Filtering Conditions

Use the **debug** command to enable the set conditions to get the filtered output.

Prerequisites

The conditions for the voice call debug filter must be set as described in the "Configuring Call-Specific Conditions" section on page 7.

SUMMARY STEPS

1. **enable**
2. **debug condition match-list** *tag* {**exact-match** | **partial-match**}
3. **debug cch323 {capacity | h225 | h245 | preauth | ras | rawmsg | session}**

   or

   debug ccsip {all | calls | err | events | messages | preauth | states}

   or

   debug isdn q931

   or

   debug voip aaa

   or

   debug voip ccapi {error | inout}

   or

   debug voip ipipgw

   or

   debug voip ivr {all | applib | callsetup | digitcollect | dynamic | error | script | settlement | states | tclcommands}

   or

   debug voip rawmsg

   or

   debug vtsp {all | dsp | error | event | port | rtp | send-nse | session | stats | vofr subframe | tone | vofr}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug condition match-list** *tag* **{exact-match | partial-match}** **Example:** `Router# debug condition match-list 1 exact-match`	Enables the filter match list for the set conditions. • *tag*—Numeric label that uniquely identifies the match list. Range is 1 to 16. The number for the match list is set using the **call filter match-list** command. • **exact-match**—All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise. • **partial- match**—No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because a large amount of debug output is generated before matches explicitly fail.

	Command or Action	Purpose

Step 3	**debug cch323 {capacity | h225 | h245 | preauth | ras | rawmsg | session}**
	or
	debug ccsip {all | calls | err | events | messages | preauth | states}
	or
	debug isdn q931
	or
	debug voip aaa
	or
	debug voip ccapi {error | inout}
	or
	debug voip ipipgw
	or
	debug voip ivr {all | applib | callsetup | digitcollect | dynamic | error | script | settlement | states | tclcommands}
	or
	debug voip rawmsg
	or
	debug vtsp {all | dsp | error | event | port | rtp | send-nse | session | stats | vofr subframe | tone | vofr}
	
	Example:
	`Router# debug cch323 h225`
	`Router# debug ccsip events`
	`Router# debug isdn q931`
	`Router# debug voip aaa`
	`Router# debug voip ccapi inout`
	`Router# debug voip ipipgw`
	`Router# debug voip ivr all`
	`Router# debug voip rawmsg`
	`Router# debug vtsp dsp`
		Enables the appropriate voice call debug commands.
		• *See the Cisco IOS Debug Command Reference* for detailed descriptions of these debug commands.
		• The debug output commences at this point.

Troubleshooting Tips

To verify debug conditions, use the following commands:

- **show debug**

  This command displays the debugs that are enabled.

- **show call filter components**

  This command displays the components that register internally with the filtering module. This command shows which components are registered with the GCFM, which is the internal module that controls which components are filtered.

- **show call filter match-list**

  This command displays the criteria set for the specified match list. It shows a list of all the match lists, shows which ones are enabled, and shows whether they are enabled for partial or exact matching.

Output Examples for Voice Call Debug Filtering

This section provides configuration examples to match the identified configuration tasks in the previous section:

- Exact Match Filtering: Example, page 14
- Partial Match Filtering: Example, page 17

Exact Match Filtering: Example

When the exact match condition is used for voice call debug filtering, all related debug output is filtered until all conditions in the match list are explicitly met. In the following example, the configuration, enabled debugs, and debug output for a Cisco AS5400 universal gateway are shown.

Dial-Peer Configuration for Exact Match Filtering

```
dial-peer voice 501 pots
 preference 1
 incoming called-number 50200
 destination-pattern 50201
 direct-inward-dial
 port 6/0:D
 prefix 50201
!
dial-peer voice 502 voip
 preference 1
 incoming called-number 50201
 destination-pattern 50200
 session target ipv4:172.16.101.21
 dtmf-relay h245-alphanumeric
 fax-relay ecm disable
 fax rate disable
!
```

Debug Output for Exact Match Filtering

```
Router# show debug

The following ISDN debugs are enabled on all DSLs:

debug isdn error is             ON.
debug isdn q931 is              ON.   (filter is ON)
Voice Telephony session debugging is on (filter is ON)
Voice Telephony dsp debugging is on (filter is ON)
Voice Telephony error debugging is on (filter is ON)
voip ccAPI function enter/exit debugging is on (filter is ON)
```

In the following output, the **show call filter match-list** command is used to show which conditions have been set for the specified call filter:

```
Router# show call filter match-list

****************************************
call filter match-list 9 voice
****************************************
  incoming calling-number 50200
  incoming called-number 50201
  incoming signal local ipv4 172.16.101.22
  incoming signal remote ipv4  172.16.101.21
  incoming media local ipv4  172.16.101.22
  incoming media remote ipv4  172.16.101.21
```

```
  incoming dialpeer 502
  outgoing calling-number 50200
  outgoing called-number 50201
  outgoing port 6/0:D
  outgoing dialpeer 501
 debug condition match-list is set to EXACT_MATCH
*****************************************
call filter match-list 10 voice
*****************************************
  incoming calling-number 50300
  incoming called-number 50301
  incoming signal local ipv4 172.16.101.22
  incoming signal remote ipv4  172.16.101.21
  incoming media local ipv4  172.16.101.22
  incoming media remote ipv4  172.16.101.21
  incoming dialpeer 504
  outgoing calling-number 50300
  outgoing called-number 50301
  outgoing port 6/1:D
  outgoing dialpeer 503
 debug condition match-list is set to EXACT_MATCH
```

The following debug output contains the exact match for the configured conditions. Some of the matching conditions are highlighted in bold text.

```
Feb  6 11:13:30.799: digit_strip:1, pcn:50201, poa:50201
Feb  6 11:13:30.799: pcn:, poa:
Feb  6 11:13:30.799: Final pcn:, poa:, dial_string:50201
Feb  6 11:13:30.803:
//6/CFD853DE8004/VTSP:(6/0:D):-1:0:0/vtsp_gcfm_percall_status_callback: found cdb and
update
Feb  6 11:13:30.803:
//6/CFD853DE8004/VTSP:(6/0:D):-1:0:0/vtsp_update_dsm_stream_mgr_filter_flag: update
dsp_stream_mgr_t  debug flag
Feb  6 11:13:30.803: //5/CFD853DE8004/CCAPI/ccapi_gcfm_percall_status_callback: found
callEntry and update
Feb  6 11:13:30.803: //6/CFD853DE8004/CCAPI/ccapi_gcfm_percall_status_callback: found
callEntry and update
Feb  6 11:13:30.803: //5/CFD853DE8004/SSAPP:502:-1/ssaTraceSct:
cid(5)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_REPORT_DIGITS_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1)
Feb  6 11:13:30.803: //5/CFD853DE8004/SSAPP:502:-1/ssaTraceSct:
-cid2(6)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
Feb  6 11:13:30.803: //5/CFD853DE8004/SSAPP:502:-1/ssaDebugPeers: ssaReportDigitsDone
cid(5) peer list: tag(2501) called number (50201)
Feb  6 11:13:30.803: //5/CFD853DE8004/SSAPP:502:-1/ssaReportDigitsDone: callid=5 Reporting
disabled.
Feb  6 11:13:31.007: ISDN Se6/0:23 Q931: callid 0x800B, callref 0x0003,
                guid CFD853DE8004
Feb  6 11:13:31.007: ISDN Se6/0:23 Q931: RX <- CALL_PROC pd = 8  callref = 0x8003
        Channel ID i = 0xA98397
                Exclusive, Channel 23
Feb  6 11:13:31.007: ISDN Se6/0:23 Q931: callid 0x800B, callref 0x0003,
                guid CFD853DE8004
Feb  6 11:13:31.007: ISDN Se6/0:23 Q931: RX <- ALERTING pd = 8  callref = 0x8003
Feb  6 11:13:31.011: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_process_event: vtsp:[6/0:D
(6), S_SETUP_REQUEST, E_TSP_PROCEEDING]
Feb  6 11:13:31.011: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/act_setup_pend_proceeding: .
Feb  6 11:13:31.011:
//6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsp_stream_mgr_reinit_platform_info: .
Feb  6 11:13:31.011: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsm_open_voice_and_set_params:
.
Feb  6 11:13:31.011: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/set_playout_dmgr: playout
default
```

```
Feb  6 11:13:31.011:
//6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsm_dsp_echo_canceller_control: echo_cancel: 1
Feb  6 11:13:31.011: ISDN Se6/0:23 Q931: callid 0x800B, callref 0x0003,
               guid CFD853DE8004
Feb  6 11:13:31.011: ISDN Se6/0:23 Q931: RX <- CONNECT pd = 8  callref = 0x8003
Feb  6 11:13:31.011: //6/CFD853DE8004/SSAPP:0:-1/ssaTraceSct:
cid(6)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_PROCEEDING)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(0)fDest(0)
Feb  6 11:13:31.011: //6/CFD853DE8004/SSAPP:0:-1/ssaTraceSct:
-cid2(5)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
Feb  6 11:13:31.011: //6/CFD853DE8004/SSAPP:0:-1/ssaCallProc:
Feb  6 11:13:31.011: //6/CFD853DE8004/SSAPP:0:-1/ssaIgnore: cid(6),
st(SSA_CS_CALL_SETTING),oldst(1), ev(21)
Feb  6 11:13:31.011: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_process_event: vtsp:[6/0:D
(6), S_SETUP_REQ_PROC, E_TSP_ALERT]
Feb  6 11:13:31.011: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/act_setup_pend_alert: .
Feb  6 11:13:31.011: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_ring_noan_timer_start:
371381
Feb  6 11:13:31.015: ISDN Se6/0:23 Q931: callid 0x800B, callref 0x0003,
               guid CFD853DE8004
Feb  6 11:13:31.015: ISDN Se6/0:23 Q931: TX -> CONNECT_ACK pd = 8  callref = 0x0003
Feb  6 11:13:31.015: //6/CFD853DE8004/SSAPP:0:-1/ssaTraceSct:
cid(6)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_ALERT)
oldst(SSA_CS_CALL_SETTING)cfid(-1)csize(0)in(0)fDest(0)
Feb  6 11:13:31.015: //6/CFD853DE8004/SSAPP:0:-1/ssaTraceSct:
-cid2(5)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
Feb  6 11:13:31.015: //5/CFD853DE8004/SSAPP:502:-1/ssaAlert:
Feb  6 11:13:31.015: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsm_exec: [Feat SM: S:NONE B
SM: S:S_DSM_INIT E:E_DSM_CC_BRIDGE]
Feb  6 11:13:31.015: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsm_act_bridge: .
Feb  6 11:13:31.015: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_dsm_bridge_status_cb: .
Feb  6 11:13:31.015: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_dsm_set_fax_feat_param:
Fax relay is ENABLED,  Primary Fax protocol is T38_FAX_RELAY, Fallback Fax protocol is
CISCO_FAX_RELAY
Feb  6 11:13:31.015: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_dsm_peer_event_cb:
E_DSM_CC_CAPS_IND
Feb  6 11:13:31.015: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_process_event: vtsp:[6/0:D
(6), S_SETUP_REQ_PROC, E_TSP_CONNECT]
Feb  6 11:13:31.015: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/act_setup_pend_connect: .
Feb  6 11:13:31.015: //6/CFD853DE8004/VTSP:(6/0:D):22:0:0/vtsp_ring_noan_timer_stop:
371382
Feb  6 11:13:31.015: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsp_stream_mgr_play_tone: .
Feb  6 11:13:31.015: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsm_exec: [Feat SM: S:NONE B
SM: S:S_DSM_BRIDGING E:E_DSM_CC_GEN_TONE]
Feb  6 11:13:31.015: //6/CFD853DE8004/DSM:(6/0:D):-1:0:4098/dsm_act_gen_tone: Tone is not
on, ignoring
Feb  6 11:13:31.015: //6/CFD853DE8004/CCAPI/cc_api_call_connected: setting
callEntry->connected to TRUE
```

Partial Match Filtering: Example

When the partial match condition is used for voice call debug filtering, no related debug output is filtered until there is a single explicit match failure. In the following example, the configuration, enabled debugs, and debug output for a Cisco 3745 modular access router are shown. Because partial match is set, the router displays the ISDN debug messages on all calls and displays only the **debug vtsp event** messages on the specified dial peer, dial peer 1.

Debug Output for Partial Match Filtering

```
Router# show debug

The following ISDN debugs are enabled on all DSLs:

debug isdn error is             ON.
debug isdn q931 is              ON.   (filter is ON)
Voice Telephony event debugging is on (filter is ON)
```

In the following output, the **show call filter match-list** command is used to show which conditions are set for the specified call filter:

```
Router# show call filter match-list 4
  incoming calling-number 10..
  incoming called-number 50..
  incoming dialpeer 1
 debug condition match-list is set to PARTIAL_MATCH
```

The following debug output shows ISDN debug messages on all calls, but displays only the **debug vtsp event** messages on dial peer **1. The VTSP messages are highlighted with bold text.**

```
*Mar  3 16:21:52.024: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01E6,
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:21:52.024: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01E6
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808381
                Exclusive, Interface 0, Channel 1
        Calling Party Number i = 0x00, 0x80, '1000'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5000'
                Plan:Unknown, Type:Unknown
*Mar  3 16:21:52.028:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_SETUP_INDICATED,  event: E_CC_PROCEEDING]
*Mar  3 16:21:52.032: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:21:52.036: ISDN Se2/0:23 Q931: TX -> CALL_PROC pd = 8  callref = 0x81E6
        Channel ID i = 0xA98381
                Exclusive, Channel 1
*Mar  3 16:21:52.080:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_PROCEEDING,  event: E_CC_ALERT]
*Mar  3 16:21:52.084: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:21:52.084: ISDN Se2/0:23 Q931: TX -> ALERTING pd = 8  callref = 0x81E6
        Progress Ind i = 0x8088 - In-band info or appropriate now available
*Mar  3 16:21:52.084:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_ALERTING,  event: E_CC_CONNECT]
*Mar  3 16:21:52.088: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
```

```
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:21:52.088: ISDN Se2/0:23 Q931: TX -> CONNECT pd = 8  callref = 0x81E6
*Mar  3 16:21:52.392: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:21:52.392: ISDN Se2/0:23 Q931: RX <- CONNECT_ACK pd = 8  callref = 0x01E6
*Mar  3 16:21:57.024: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01E7,
                guid 09E86AA0-170A-11CC-81E8-000B465B86B0
*Mar  3 16:21:57.024: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01E7
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808382
                Exclusive, Interface 0, Channel 2
        Calling Party Number i = 0x00, 0x80, '1001'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5001'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:02.032: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01E8,
                guid 0CE49409-170A-11CC-81E9-000B465B86B0
*Mar  3 16:22:02.032: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01E8
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808383
                Exclusive, Interface 0, Channel 3
        Calling Party Number i = 0x00, 0x80, '1002'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5002'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:07.032: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01E9,
                guid 0FDF8489-170A-11CC-81EA-000B465B86B0
*Mar  3 16:22:07.032: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01E9
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808384
                Exclusive, Interface 0, Channel 4
        Calling Party Number i = 0x00, 0x80, '1003'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5003'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:12.032: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01EA,
                guid 12DA7509-170A-11CC-81EB-000B465B86B0
*Mar  3 16:22:12.032: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01EA
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808385
                Exclusive, Interface 0, Channel 5
        Calling Party Number i = 0x00, 0x80, '1004'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5004'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:17.036: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01EB,
                guid 15D601B1-170A-11CC-81EC-000B465B86B0
*Mar  3 16:22:17.036: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01EB
```

```
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808386
                Exclusive, Interface 0, Channel 6
        Calling Party Number i = 0x00, 0x80, '1005'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5005'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:22.040: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01EC,
                guid 18D18E59-170A-11CC-81ED-000B465B86B0
*Mar  3 16:22:22.040: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01EC
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808387
                Exclusive, Interface 0, Channel 7
        Calling Party Number i = 0x00, 0x80, '1006'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5006'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:27.040: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01ED,
                guid 1BCC7ED9-170A-11CC-81EE-000B465B86B0
*Mar  3 16:22:27.040: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01ED
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808388
                Exclusive, Interface 0, Channel 8
        Calling Party Number i = 0x00, 0x80, '1007'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5007'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:32.048: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01EE,
                guid 1EC8A7A9-170A-11CC-81EF-000B465B86B0
*Mar  3 16:22:32.048: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01EE
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xE9808382
                Exclusive, Interface 0, Channel 2
        Calling Party Number i = 0x00, 0x80, '1008'
                Plan:Unknown, Type:Unknown
        Called Party Number i = 0x80, '5008'
                Plan:Unknown, Type:Unknown
*Mar  3 16:22:34.688: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:22:34.688: ISDN Se2/0:23 Q931: RX <- DISCONNECT pd = 8  callref = 0x01E6
        Cause i = 0x8290 - Normal call clearing
*Mar  3 16:22:34.688: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
                guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:22:34.688: ISDN Se2/0:23 Q931: TX -> RELEASE pd = 8  callref = 0x81E6
*Mar  3 16:22:34.688:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_CONNECT,  event: E_TSP_DISCONNECT_IND]
```

```
*Mar  3 16:22:34.688:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_CONNECT,  event: E_CC_DISCONNECT]
*Mar  3 16:22:34.692:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_WAIT_STATS,  event: E_VTSP_DSM_STATS_COMPLETE]
*Mar  3 16:22:34.692:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_WAIT_RELEASE,  event: E_TSP_DISCONNECT_CONF]
*Mar  3 16:22:34.692:
//1270/06ED7A20-170A-11CC-81E7-000B465B86B0/VTSP:(2/0:23):0:0:0/vtsp_process_event:
[state:S_CLOSE_DSPRM,  event: E_VTSP_DSM_CLOSE_COMPLETE]
*Mar  3 16:22:34.812: ISDN Se2/0:23 Q931: callid 0x01EA, callref 0x01E6,
               guid 06ED7A20-170A-11CC-81E7-000B465B86B0
*Mar  3 16:22:34.812: ISDN Se2/0:23 Q931: RX <- RELEASE_COMP pd = 8  callref = 0x01E6
*Mar  3 16:22:37.048: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01EF,
               guid 21C39829-170A-11CC-81F0-000B465B86B0
*Mar  3 16:22:37.048: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01EF
       Bearer Capability i = 0x8090A2
               Standard = CCITT
               Transer Capability = Speech
               Transfer Mode = Circuit
               Transfer Rate = 64 kbit/s
       Channel ID i = 0xE9808381
               Exclusive, Interface 0, Channel 1
       Calling Party Number i = 0x00, 0x80, '1009'
               Plan:Unknown, Type:Unknown
       Called Party Number i = 0x80, '5009'
               Plan:Unknown, Type:Unknown
*Mar  3 16:22:42.052: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01F0,
               guid 24BF24D1-170A-11CC-81F1-000B465B86B0
*Mar  3 16:22:42.052: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01F0
       Bearer Capability i = 0x8090A2
               Standard = CCITT
               Transer Capability = Speech
               Transfer Mode = Circuit
               Transfer Rate = 64 kbit/s
       Channel ID i = 0xE9808383
               Exclusive, Interface 0, Channel 3
       Calling Party Number i = 0x00, 0x80, '1010'
               Plan:Unknown, Type:Unknown
       Called Party Number i = 0x80, '5010'
               Plan:Unknown, Type:Unknown
*Mar  3 16:22:47.056: ISDN Se2/0:23 Q931: callid 0x0000, callref 0x01F1,
               guid 27BAB179-170A-11CC-81F2-000B465B86B0
*Mar  3 16:22:47.056: ISDN Se2/0:23 Q931: RX <- SETUP pd = 8  callref = 0x01F1
       Bearer Capability i = 0x8090A2
               Standard = CCITT
               Transer Capability = Speech
               Transfer Mode = Circuit
               Transfer Rate = 64 kbit/s
       Channel ID i = 0xE9808384
               Exclusive, Interface 0, Channel 4
       Calling Party Number i = 0x00, 0x80, '1011'
               Plan:Unknown, Type:Unknown
       Called Party Number i = 0x80, '5011'
               Plan:Unknown, Type:Unknown
```

Voice Call Debug Filtering on H.323 Gatekeepers

Use voice call debug filtering on Cisco voice gatekeepers to get selected debugging traces for voice calls. This feature allows you to filter and trace voice call debug messages based on selected filtering criteria, reducing the volume of output for more efficient troubleshooting.

This section describes the capability added in Cisco IOS Release 12.4(4)T to filter troubleshooting output on H.323 gatekeepers and contains the following sections:

Restrictions for Voice Call Debug Filtering on Voice Gatekeepers

The following restrictions apply to the Cisco voice gatekeepers:

- Voice call debug filtering on gatekeepers is available only if you have a Cisco IOS image that contains the gatekeeper functionality or joint functionality (for both gateways and gatekeepers).
- Filtering of debugs in the gatekeeper DNS process is not supported.
- Filtering of ASN and GUP messages is not supported.
- The following are not supported by call filtering:
  - If debug messages are printed in non-ARQ path, the path will not support call filtering.
  - H.323V1, H323 proxy, and DGK are not supported.
  - Calls transferred from primary gatekeeper to the secondary gatekeeper will not be filtered in the secondary gatekeeper, but new calls in secondary gatekeeper will apply filtering.
  - All error, incorrect, and delayed GKTMP messages are not printed in call filtering because the gatekeeper ignores the message at a lower level and the messages are not printed.
  - GKTMP messages that are not related to calls are not supported by call filtering (registration, unregistration, and resource messages).

Information About Voice Call Debug Filtering

Information from using debug commands for voice calls is crucial for troubleshooting, but the volume of raw data can be very large. In order to isolate the most valuable data, you can use the Voice Call Debug Filtering feature on gatekeepers. Debug output for the voice gatekeepers can be filtered according to the following criteria:

- Incoming called number
- Incoming calling number
- Gatekeeper resolved final destination endpoint (IPv4) address

Note In addition to working on voice gatekeepers, call filtering works on IP-to-IP gateway connections using H.323.

The selected criteria are set on the gatekeeper, and different sets of criteria can be stored.

To better understand the voice call debug filtering on Cisco voice gatekeepers, see the following sections:

Debug Commands that Support Voice Call Filtering on Cisco Voice Gatekeepers

When a call filter is applied, the filtering applies to all of the debugs affected by the call filter. Debug commands that support voice call debug filtering on gatekeepers are the following:

- **debug gatekeeper call** *level*
- **debug gatekeeper main** *level*
- **debug gatekeeper servers**

Note In the syntax, the *level* argument can be a number from 1 through 10. Refer to the *Cisco IOS Debug Command Reference* for detailed information about these debug commands.

Gatekeeper Filter Module

The filtering for these modules is managed by the gatekeeper filter module (GKFM). The filtering conditions are configured in the GKFM, and then the individual modules are informed when a call has to be filtered. The GKFM coordinates between multiple modules to handle filtering conditions.

The GKFM provides the core infrastructure for the gatekeeper module to use for providing call tracing capabilities. This module is part of the GCFM subsystem (sub_gcfm) and provides all the necessary implementations for the command-line interface (CLI) functions, conditions matching logics for the gatekeeper.

The gatekeeper debug subsystem uses the GKFM infrastructure to implement standard style header formats.

Activity in the GKFM can be traced using the **debug call filter detail** and **debug call filter inout** commands. Refer to the *Cisco IOS Debug Command Reference* for more information about these debug commands.

The GKFM functionalities are summarized as follows:

- Maintain a set of matching lists that are activated through the CLI interface for filtering out desired calls.
- Provide an interface to the gatekeeper modules for filter query for exact and partial matches.
- Implement an algorithm to execute the matching logic by examining the activated filter tags provisioned through the CLI interface. The algorithm is optimized in a way that the logic is executed only when there is a new call parameter populated in the context or there is a change in the CLI provisioned data.
- Employ a filter rules engine to prevent activation of illegal and conflicting filter tags (that is, filter tags having conflicting conditions provisioned across them).

Calling and Called Number Strings

The string pattern for calling and called numbers can be either a complete telephone number or a partial telephone number with wildcard digits, represented by a period (.) character. Each "." represents a wildcard for an individual digit that the originating voice gateway expects to match. For example, if the calling and called number strings is defined as "555. . . .", then any dialed string beginning with 555, plus at least four additional digits, matches this calling or called number.

Table 9 shows all of the wildcard symbols that are supported in the calling and called number strings.

Table 9 Wild CardSymbols Supported in Calling and Called Number Strings

Symbol	Description
.	Indicates a single-digit placeholder. For example, 555. . . . matches any dialed string beginning with 555, plus at least four additional digits.
[ ]	Indicates a range of digits. A consecutive range is indicated with a hyphen (-); for example, [5-7]. A nonconsecutive range is indicated with a comma (,); for example, [5,8]. Hyphens and commas can be used in combination; for example, [5-7,9]. **Note** Only single-digit ranges are supported. For example, [98-102] is invalid.
( )	Indicates a pattern; for example, 408(555). It is used in conjunction with the symbol ?, %, or +.
?	Indicates that the preceding digit occurred zero or one time. Press **ctrl-v** before entering ? from your keyboard.
%	Indicates that the preceding digit occurred zero or more times. This functions the same as the "*" used in regular expression.
+	Indicates that the preceding digit occurred one or more times.
T	Indicates the interdigit timeout. The voice gateway pauses to collect additional dialed digits.

Note The wildcard symbols follow regular expression rules, and whatever wildcard applies to the gateway applies to the gatekeeper as well. The period (.) is the only wildcard character that is supported for dial strings that are configured using the **answer-address** or **incoming called-number** command.

Exact and Partial Matching Conditions Rules and Information

To filter out desired calls, you must define a list of matching conditions. There must be separate filters defined for VoIP gateways and VoIP gatekeepers, because both the gateway and gatekeeper can exist on the same box.

Multiple filter tags could be provisioned for the gatekeeper, each with one or all three different conditions under them. But the GKFM provides a rules engine to validate while activating thefilter tags when one or more filter tags are contradictory or conflicting with each other. Matching conditions are as follows:

- Exact match—All related debug output is filtered and shown when all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise.

- Partial match—Related debug output is filtered and shown when even only one condition matches. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because of the large amount of debug output that might be generated before matches explicitly fail.

To filter out the desired calls, you must define a list of matching conditions. Separate filters need to be defined for VoIP gateways and VoIP gatekeepers, because both the gateway and gatekeeper can exist on the same box.

The following global CLI is used to define the conditions on the gatekeeper. The **gatekeeper** keyword appears in the CLI only if you have a Cisco IOS image that contains gatekeeper debug filter functionality or a combination of gateway and gatekeeper debug filter functionality.

call filter match-list *number* **voice gatekeeper**

Then, under the submode, a set of conditions can be defined. Only the following will be valid for filtering calls on the gatekeeper:

incoming calling-number *string*

incoming called-number *string*

One new match condition is added that is valid for gatekeeper filtering only:

gatekeeper resolved ipv4 *ipaddress*

For applying the match filters configured, the CLI for gatekeepers is the same as that used for gateways:

debug condition match-list *<1-16>* **exact-match** | **partial-match**

Table 10 summarizes the rules applied while you are executing the exact and partial match logic.

Table 10 Rules Applied for Executing Exact and Partial Matching Logic

Number	Decription
1	The exact match is an AND logic across all the provisioned conditions. For example: call filter match-list 2 gatekeeper incoming called-number 9876 incoming calling-number 2345 resolved destination ipv4 1.2.3.4 The debugs are emitted only when all the callp provided input matches. IC Called Number **&&** IC Calling Number **&&** Resolved destination IP address
2	The partial match is a logical OR across all the provisioned conditions. For example: call filter match-list 2 gatekeeper incoming called-number 9876 incoming calling-number 2345 resolved destination ipv4 1.2.3.4 The debugs are emitted only when any one of the callp-provided inputs matches. IC Called Number || IC Calling Number || Resolved destination IP address

Table 10 Rules Applied for Executing Exact and Partial Matching Logic (continued)

Number	Decription
3	When the admin provisions a set of conditions under a filter tag that may not contain all three (supported) conditions, then the rules are applied only on that subset of conditions. The non-provisioned conditions are considered as wildcard or don't care. For example: call filter match-list 2 gatekeeper incoming called-number 9876 resolved destination ipv4 1.2.3.4 Because incoming calling number was not provided, it is taken as a don't care condition and excluded when the GKFM is executing the matching logic. This tag shall be an exact match for all calls with called number 9876, resolved destination 1.2.3.4, and any calling number xxxx.
4	When the admin provisions only the resolved destination address under a filter tag, then all debugs (for which filtering is employed) shall be throwing the outputs until the callp learns the resolved destination address (that is, until the RAS-LCF message was obtained or the route server provides the resolved address during a later part of the call). This scenario should be well understood before such a filter tag is activated. For example: call filter match-list 1 gatekeeper resolved destination ipv4 1.2.3.4 Then debugs for all the calls with any IC called/calling numbers shall be thrown until the resolved destination address is known. When the resolved destination address happens to be 1.2.3.4, then the debug outputs would continue—if not, it would stop at that point.
5	If the admin provisions a filter tag with no matching conditions under it, such a tag cannot be activated. For example: ! call filter match-list 3 gatekeeper ! #deb condition match-list 3 exact Failed: Activation of tag with no filter condition is not allowed.

Table 10 Rules Applied for Executing Exact and Partial Matching Logic (continued)

Number	Decription
6	If an activated filter tag condition is modified, that tag needs to be explicitly reactivated after any changes. For example: ! call filter match-list 2 gatekeeper incoming called-number 9876 incoming calling-number 2345 resolved destination ipv4 1.2.3.4 ! # show call filter match-list ********** call filter match-list 2 gatekeeper ********** incoming called-number 9876 incoming calling-number 2345 gatekeeper resolved destination address 1.2.3.4 debug condition match-list is set to EXACT_MATCH ! # conf t (config)# call filter match-list 2 gatekeeper (conf-call-filter-mlist)# incoming calling-number 4089 (conf-call-filter-mlist)# end # # show call filter match-list ********** call filter match-list 2 gatekeeper ********** incoming called-number 9876 incoming calling-number 2345 gatekeeper resolved destination address 1.2.3.4 debug condition match-list is not ARMED
7	A call for which the GKFM decides failed match (suppress) does not result in any debugs or reexecution of matching logic during the lifespan of that call, even though the CLI admin modifies the provisioned conditions during the lifespan of that call.

GKFM Debug Filter Rules Engine

The following are the rules employed by the filter activation rules engine in order to avoid the conflicts between one or more activated filter tags.

Y- indicates configured

X- indicates not configured (don’t care)

1. New condition pattern to be activated—CDN:Y, CGN:X, RES_IPV4:X

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
Y	X	X	Compare CDN only. Reject - if same Allow - if different
X	X	Y	Reject always.
X	Y	X	Reject always.
X	Y	Y	Reject always.
Y	X	Y	Compare CDN only. Reject - if same Allow - if different
Y	Y	X	Compare CDN only. Reject - if same Allow - if different
Y	Y	Y	Compare CDN only. Reject - if same Allow - if different

2. New condition pattern to be activated—CDN:X, CGN:Y, RES_IPV4:X

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
X	Y	X	Compare CGN only. Reject - if same Allow - if different
X	Y	Y	Compare CGN only. Reject - if same Allow - if different
Y	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	Y	Y	Compare CGN only. Reject - if same Allow - if different
X	X	Y	Reject always.
Y	X	X	Reject always.
Y	X	Y	Reject always.

3. New condition pattern to be activated—CDN:Y, CGN:Y, RES_IPV4:X

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
Y	Y	X	Compare CDN and CGN. Reject - if both are same Allow - if anyone different
Y	Y	Y	Compare CDN and CGN only. Reject - if both are same Allow - if anyone different
X	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	X	X	Compare CDN only. Reject - if same Allow - if different
X	X	Y	Reject always.
X	Y	Y	Compare CGN only. Reject - if same Allow - if different
Y	X	Y	Compare CDN only. Reject - if same Allow if different

4. New condition pattern to be activated—CDN:X, CGN:X, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
X	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
Y	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
X	Y	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
Y	Y	Y	Compare Res. IPv4 only. Reject - if same Allow - if different

X	Y	X	Reject always.
Y	X	X	Reject always.
Y	Y	X	Reject always.

5. New condition pattern to be activated—CDN:Y, CGN:X, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed rules
Y	X	Y	Compare CDN and Res. IPv4. Reject - if both are same Allow - if anyone different
Y	Y	Y	Compare CDN and Res. IPv4. Reject - if both are same Allow - if anyone different.
Y	X	X	Compare CDN only. Reject - if same Allow - if different
Y	Y	X	Compare CDN only. Reject - if same Allow - if different
X	X	Y	Compare IPv4 only. Reject - if same Allow - if different
X	Y	Y	Compare IPv4 only. Reject - if same Allow - if different
X	Y	X	Reject always.

6. New condition pattern to be activated—CDN:X, CGN:Y, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
X	Y	Y	Compare CGN and Res. IPv4. Reject - if both are same Allow - if anyone different
Y	Y	Y	Compare CGN and Res. IPv4. Reject - if both are same Allow - if anyone different.
X	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different

Y	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
X	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	X	X	Reject always.

7. New condition pattern to be activated—CDN:Y, CGN:Y, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
Y	Y	Y	Compare CDN, CGN and Res. IPv4. Reject - if all are same. Allow - if any one different
X	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different.
X	Y	X	Compare CGN only. Rejectect - if same Allow - if different
X	Y	Y	Compare CGN & Res. IPv4. Rejectect - if both are same Allow - if any one different
Y	X	X	Compare CDN only. Reject - if same Allow - if different
Y	X	Y	Compare CDN and Res. IPv4 only. Reject - if both are same Allow - if any one different
Y	Y	X	Compare CDN and CGN only. Reject - if both are same Allow - if any one different.

Configuring the Voice Call Debug Filter for Cisco Gatekeepers

To configure the voice call debug filter on Cisco gatekeepers, perform the following tasks:

- Configuring Call-Specific Conditions for Gatekeepers, page 31 (required)
- Enabling Debug for the Set Filtering Conditions, page 32 (required)

Configuring Call-Specific Conditions for Gatekeepers

Configure call-specific conditions to set the attributes that are filtered for voice calls.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call filter match-list** *number* **gatekeeper**
4. **incoming calling-number** *string*
5. **incoming called-number** *string*
6. gatekeeper resolved ipv4 *ipaddress*
7. **exit**
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**call filter match-list** *number* **gatekeeper** **Example:** `Router(config)# call filter match-list 1 gatekeeper`	Enters call filter match list configuration mode to define the filter conditions on the gatekeeper. • *number*—Numeric label that uniquely identifies the match list. Range is 1 to 16. **Note** At least one of the following optional parameters (Step 4 to Step 7) for call filtering must be configured.
Step 4	**incoming calling-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming calling-number 408525`	(Optional) Specifies the incoming calling number to be filtered. • *string*—Numeric string that identifies all or part of the incoming calling number.

	Command or Action	Purpose
Step 5	**incoming called-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming called-number 408525`	(Optional) Specifies the incoming called number to be filtered. • *string*—Numeric string that identifies all or part of the incoming called number.
Step 6	**gatekeeper resolved ipv4** *ipaddress* **Example:** `Router(conf-call-filter-mlist)# gatekeeper resolved ipv4 10.1.1.1`	(Optional) Specifies the gatekeeper-resolved destination IP address to be filtered. • *ipaddress*—IP address that identifies the destination.
Step 7	**exit** **Example:** `Router(conf-call-filter-mlist)# exit`	Exits to global configuration mode.
Step 8	**exit** **Example:** `Router(config)# exit`	Exits to privileged EXEC mode.

Enabling Debug for the Set Filtering Conditions

Use the **debug** commands to enable the set conditions to get the filtered output.

Prerequisites

The conditions for the voice call debug filter must be set as described in the "Configuring Call-Specific Conditions for Gatekeepers" section on page 31.

SUMMARY STEPS

1. **enable**
2. **debug condition match-list** *tag* {**exact-match** | **partial-match**}
3. **debug gatekeeper call** *level*

   or

   debug gatekeeper main *level*

   or

   debug gatekeeper servers

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug condition match-list** *tag* **{exact-match | partial-match}** **Example:** `Router# debug condition match-list 1 exact-match`	Enables the filter match list for the set conditions. • *tag*—Numeric label that uniquely identifies the match list. Range is 1 to 16. The number for the match list is set using the **call filter match-list** command. • **exact-match**—All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise. • **partial-match**—No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because a large amount of debug output is generated before matches explicitly fail.

Step 3	**debug gatekeeper call** *level*
	or
	debug gatekeeper main *level*
	or
	debug gatekeeper servers
	
	Example:
	`Router# debug gatekeeper call 5`
	
	Example:
	`Router# debug gatekeeper main 3`
	
	Example:
	`Router# debug gatekeeper servers`
		Enables the appropriate gatekeeper call debug commands.
		• *level—Specifies a level for the debug message. Entries can be 1 through 10.*
		• *Refer to the Cisco IOS Debug Command Reference* for detailed descriptions of these debug commands.
		• The debug output commences at this point.

Troubleshooting Tips

To verify debug conditions, use the following commands:

- **show debug**

  This command displays the debugs that are enabled.

- **show call filter components**

  This command displays the components that register internally with the filtering module. This command shows which components are registered with the GKFM, which is the internal module that controls which components are filtered.

- **show call filter match-list**

  This command displays the criteria set for the specified match list. It shows a list of all the match lists, shows which ones are enabled, and shows whether they are enabled for partial or exact matching.

Examples

This section provides sample output and a typical debug listing:

- Sample Output for show call filter and show debug: Example, page 34
- Sample Output for show debug: Example, page 34

Sample Output for show call filter and show debug: Example

When the exact match condition is used for gatekeeper voice call debug filtering, all related debug output is filtered until all conditions in the match list are explicitly met:

```
Router# show call filter match-list

*****************************************
call filter match-list 1 gatekeeper
*****************************************
gatekeeper resolved destination address 10.77.154.91
incoming called-number 444
debug condition match-list is not armed
*****************************************
call filter match-list 2 gatekeeper
*****************************************
incoming called-number 204
debug condition match-list is set to EXACT_MATCH
Router#
################################
Router# show debug
Gatekeeper:
Gatekeeper Server Messages debugging is on (filter is ON)
gk call debug level = 10 (filter is ON)
gk zone debug level = 10
c2691-1-OGK#
```

Sample Output for show debug: Example

Following is a sample output resulting from the **show debug** command:

```
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_arq_select_viazone: matched
zone is GK-A, and z_invianamelen=0
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_arq_select_viazone: about to
check the destination side, dst_zonep=0x6293721C
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_arq_select_viazone: matched
zone is GK-A, and z_outvianamelen=0
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_get_addrinfo: No tech prefix
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_get_addrinfo: Alias not found
```

*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_get_addrinfo: (203) unknown address and no default technology defined.

SIP Debug Output Filtering Support

The SIP Debug Output Filtering Support feature provides the capability for SIP-related debug output to be filtered based on a set of user-defined matching conditions. This section contains the following information:

Restrictions for SIP Debug Output Filtering Support

The SIP Debug Output Filtering Support feature filters all enabled bugs, not just SIP bugs.

Information About SIP Debug Output Filtering Support

To configure the SIP Debug Output Filtering Support feature, you should understand the following concepts:

Feature Design of SIP Debug Output Filtering Support

Prior to the SIP Debug Output Filtering Support feature, debugging and troubleshooting on the VoIP gateway was made more challenging by the extensive amounts of raw data generated by debug output. This feature allows the debug output for a SIP call to be filtered according to a variety of criteria. The SIP Debug Output Filtering Support feature provides a generic call filtering mechanism that does the following:

- Allows you to define a set of matching conditions for filtering calls.
- Identifies the desired calls based on the configured matching conditions inside VoIP gateways.
- Coordinates the filtering effort on traced calls between multiple modules inside VoIP gateways.
- Displays the debugging trace for calls that match the specified conditions.

The SIP Debug Output Filtering Support feature document provides SIP-specific information on the debug filtering design in Cisco IOS gateways implemented by the voice call debug filtering feature. See the “Voice Call Debug Filtering on Cisco Voice Gateways” section on page 2 for more information.

Generic Call Filtering Module

Cisco implements the SIP Debug Output Filtering Support feature using the GCFM to identify and track calls based on configured parameters. You can use the command-line interface (CLI) to turn the GCFM on and off. When the GCFM is turned on, only the debug messages for calls that match the filtering conditions are displayed.

The GCFM manages the set of matching condition lists and coordination and tracking of calls between multiple modules within the voice gateway architecture. The SIP module uses the GCFM function to create GUIDs to track individual inbound and outbound SIP calls. Activity in the GCFM can be traced using the **debug call filter detail** and **debug call filter inout** commands. See the *Cisco IOS Debug Command Reference* for more information about these debug commands.

SIP Debug Commands that Support Output Filtering

- **debug ccsip all**
- **debug ccsip calls**
- **debug ccsip events**
- **debug ccsip messages**
- **debug ccsip preauth**
- **debug ccsip states**

See the *Cisco IOS Debug Command Reference* for more information about SIP debug commands.

Note Because of the importance of the messages associated with the **debug ccsip err** command, this debug output is not filterable.

Matching Conditions

To filter calls, define a list of matching conditions. The SIP Debug Output Filtering Support feature supports matches based on the following conditions:

- Incoming calling-number string
- Incoming called-number string
- Incoming signaling IPv4 local address
- Incoming signaling IPv4 remote address
- Incoming media IPv4 local address
- Incoming media IPv4 remote address
- Incoming dial peer
- Outgoing calling-number string
- Outgoing called-number string
- Outgoing signaling IPv4 local address
- Outgoing signaling IPv4 remote address
- Outgoing media IPv4 local address
- Outgoing media IPv4 remote address

- Outgoing dial peer

The string pattern for calling and called numbers can be either a complete telephone number or a partial telephone number with wildcard digits, represented by a period (.) character.

You can define matching conditions as follows:

- Exact match—All related debug output is filtered until all conditions in the match list are explicitly met. This option is the best choice for most situations because it produces the most concise output.
- Partial match—No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because of the large amount of debug output until matches explicitly fail.

See the "Exact and Partial Matching" section on page 6 for more information on matching conditions and filtering voice calls.

Configuring SIP Debug Filtering

This section contains the following procedures:

- Configuring Call Filters, page 37 (required)
- Enabling SIP Debug Output Filtering, page 39 (required)
- Verifying SIP Debug Output Filtering Support, page 41 (optional)

Configuring Call Filters

This task configures the conditions for filtering SIP calls.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call filter match-list** *number* **voice**
4. incoming calling-number *string*
5. incoming called-number *string*
6. **incoming signaling** {**local** | **remote**} **ipv4** ***ip-address***
7. **incoming media** {**local** | **remote**} **ipv4** ***ip-address***
8. **incoming dialpeer** ***tag***
9. **outgoing calling-number** ***string***
10. **outgoing called-number** ***string***
11. **outgoing signaling** {**local** | **remote**} **ipv4** ***ip-address***
12. **outgoing media** {**local** | **remote**} **ipv4** ***ip-address***
13. **outgoing dialpeer** ***tag***
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**call filter match-list** *number* **voice** **Example:** `Router(config)# call filter match-list 1 voice`	Enters call filter match list configuration mode to define the filter conditions. • *number*—Numeric label that uniquely identifies the match list. Range is 1 to 16. **Note** At least one of the following optional parameters (Step 4 to Step 13) for call filtering must be configured.
Step 4	**incoming calling-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming calling-number 408555`	(Optional) Specifies the incoming calling number to be filtered. • *string*—Numeric string that identifies all or part of the incoming calling number.
Step 5	**incoming called-number** *string* **Example:** `Router(conf-call-filter-mlist)# incoming called-number 408555`	(Optional) Specifies the incoming called number to be filtered. • *string*—Numeric string that identifies all or part of the incoming called number.
Step 6	**incoming signaling** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# incoming signaling local ipv4 192.168.10.255`	(Optional) Specifies the incoming signaling IPv4 address. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 7	**incoming media** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# incoming media local ipv4 192.168.10.255`	(Optional) Specifies the incoming media IPv4 address for the voice gateway receiving the media stream. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 8	**incoming dialpeer** *tag* **Example:** `Router(conf-call-filter-mlist)# incoming dialpeer 14`	(Optional) Specifies the incoming dial peer to be filtered. • *tag*—Digits that define a specific dial peer. Valid entries are 1 to 2147483647.

	Command or Action	Purpose
Step 9	**outgoing calling-number** *string* **Example:** `Router(conf-call-filter-mlist)# outgoing calling-number 408555`	(Optional) Specifies the outgoing calling number to be filtered; this number goes out after number translation and expansion are complete. • *string*—Numeric string that identifies all or part of the outgoing calling number.
Step 10	**outgoing called-number** *string* **Example:** `Router(conf-call-filter-mlist)# outgoing called-number 408555`	(Optional) Specifies the outgoing called number to be filtered; this number goes out after number translation and expansion are complete. • *string*—Numeric string that identifies all or part of the outgoing called number.
Step 11	**outgoing signaling** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# outgoing signaling local ipv4 192.168.10.255`	(Optional) Specifies the outgoing signaling IPv4 address for the gatekeeper managing the signaling. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 12	**outgoing media** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# outgoing media local ipv4 192.168.10.255`	(Optional) Specifies the outgoing media IPv4 address for the voice gateway receiving the media stream. • **local**—Local voice gateway • **remote**—Remote IP device • *ip-address*—IP address of the local voice gateway.
Step 13	**outgoing dialpeer** *tag* **Example:** `Router(conf-call-filter-mlist)# outgoing dialpeer 14`	(Optional) Specifies the outgoing dial peer to be filtered. • *tag*—Digits that define a specific dial peer. Valid entries range from1 to 2147483647.
Step 14	**end** **Example:** `Router(conf-call-filter-mlist)# end`	Exits to privileged EXEC mode.

What to Do Next

After the conditions are set for filtering the SIP call debug output, **debug** commands can be enabled. Proceed to the "Enabling SIP Debug Output Filtering" section.

Enabling SIP Debug Output Filtering

This task enables debug filtering for SIP-related output.

Prerequisites

The conditions for the SIP call debug filter must be set as described in the "Configuring Call Filters" section on page 37.

SUMMARY STEPS

1. **enable**
2. **debug condition match-list** *number* {**exact-match** | **partial-match**}
3. debug ccsip {all | calls | events | messages | preauth | states}

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug condition match-list** *number* **{exact-match | partial-match)** **Example:** `Router# debug condition match-list 1 exact-match`	Enables the filter match list for the set conditions. • *number*—Numeric label that uniquely identifies the match list. Range is 1 to 16. • **exact-match**—All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise. • **partial-match**—No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because a large amount of debug output is generated before matches explicitly fail.
Step 3	**debug ccsip {all | calls | events | messages | preauth | states}** **Example:** `Router# debug ccsip all`	Enables the appropriate SIP call debug commands. • *See the Cisco IOS Debug Command Reference* for detailed descriptions of these debug commands. • The debug output begins at this point.

Verifying SIP Debug Output Filtering Support

Perform this task to verify debug output filtering.

SUMMARY STEPS

1. **show debug**
2. **show call filter components**
3. show call filter match-list

DETAILED STEPS

Step 1 **show debug**

Use this command to display the debugs that are enabled, for example:

```
Router# show debug

CCSIP SPI:SIP Call Statistics tracing is enabled       (filter is ON)
CCSIP SPI:SIP Call Message tracing is enabled  (filter is ON)
CCSIP SPI:SIP Call State Machine tracing is enabled    (filter is ON)
CCSIP SPI:SIP Call Events tracing is enabled   (filter is ON)
CCSIP SPI:SIP error debug tracing is enabled   (filter is ON)
CCSIP SPI:SIP info debug tracing is enabled    (filter is ON)
CCSIP SPI:SIP media debug tracing is enabled   (filter is ON)
CCSIP SPI:SIP Call preauth tracing is enabled (filter is ON)
```

Step 2 **show call filter components**

Use this command to display which components are registered with the GCFM, the internal module that controls which components are filtered:

```
Router# show call filter components

 The following components registered in GCFM:
     ISDN
     VTSP
     CCAPI
     TGRM
     DIAL-PEER
     NUMBER-TRANSLATION
     SSAPP
     VOICE-IVR-V2
     H323
     SIP
     CRM
```

Step 3 **show call filter match-list**

Use this command to display the criteria set for the specified match list. The command shows a list of all the match lists, shows which ones are enabled, and shows whether they are enabled for partial or exact matching.

```
Router# show call filter match-list

****************************************
call filter match-list 1 voice
****************************************
  incoming called-number 5551200
```

Configuration Examples for SIP Debug Filtering

This section provides the following configuration examples:

Configuring Call Filters: Example

The following example shows how to configure call filters. In the example, the Cisco AS5300 configuration defining a match list and specifying the incoming called number to be filtered is shown.

```
Router# show running-config

Building configuration...

Current configuration :3944 bytes
!
! Last configuration change at 13:07:08 EST Tue Sep 2 2003
!
version 12.3
no service pad
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
no service password-encryption
service internal
!
hostname Router-1
!
logging buffered 500000 debugging
enable password lab
!
!
!
resource-pool disable
clock timezone EST -5
clock summer-time EST recurring
!
no aaa new-model
ip subnet-zero
ip domain name cisco.com
ip host CALLGEN-SECURITY-V2 10.73.75.86 8.52.0.0
ip name-server 10.44.11.21
!
!
isdn switch-type primary-dms100
no scripting tcl init
no scripting tcl encdir
!
!
voice call carrier capacity active
!
voice service voip
 no supplementary-service h450.12
 sip
!
!
voice class codec 1
 codec preference 1 g711ulaw
 codec preference 2 g729r8
!
```

```
!
voice hpi capture buffer 10000
no voice hpi capture destination
!
!
fax interface-type modem
call-history-mib retain-timer 500
call-history-mib max-size 500
!
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
clock source line secondary 1
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 2
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 3
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
!
interface Ethernet0
 no ip route-cache
 no ip mroute-cache
 shutdown
 no cdp enable
!
interface Serial0:23
 no logging event link-status
isdn switch-type primary-dms100
 isdn incoming-voice modem
 no cdp enable
!
interface Serial1:23
 no logging event link-status
 isdn switch-type primary-dms100
 isdn incoming-voice modem
 no cdp enable
!
interface Serial2:23
 no logging event link-status
 isdn switch-type primary-dms100
 isdn incoming-voice modem
 no cdp enable
!
interface Serial3:23
 isdn switch-type primary-dms100
 no cdp enable
!
interface FastEthernet0
 ip address 172.18.195.43 255.255.0.0
 no ip route-cache
```

```
no ip mroute-cache
 duplex auto
 speed auto
 fair-queue 64 256 235
 no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 FastEthernet0
ip route 10.0.0.0 255.0.0.0 172.18.193.3
ip route 10.0.0.0 255.0.0.0 172.18.195.1
ip route 172.18.0.0 255.255.0.0 172.18.193.1
ip http server
!
!
!
control-plane
!
!
!
call filter match-list 1 voice
incoming called-number 4085559876
!
voice-port 0:D
!
voice-port 1:D
!
voice-port 2:D
!
voice-port 3:D
!
!
dial-peer cor custom
!
!
!
dial-peer voice 1100 pots
 destination-pattern 55511..
 direct-inward-dial
 port 0:D
 prefix 55511
!
dial-peer voice 1200 pots
 destination-pattern 55512..
direct-inward-dial
 port 1:D
 prefix 55512
!
dial-peer voice 444 pots
 destination-pattern 444....
 direct-inward-dial
 port 2:D
 prefix 444
!
dial-peer voice 5100 voip
 destination-pattern 55551..
 session protocol sipv2
 session target ipv4:172.18.207.10:5060
 dtmf-relay rtp-nte
 codec g711ulaw
!
dial-peer voice 5200 voip
 destination-pattern 55552..
 session protocol sipv2
 session target ipv4:172.18.207.10:5060
```

```
 dtmf-relay rtp-nte
!
dial-peer voice 5300 voip
 destination-pattern 55553..
 session protocol sipv2
 session target ipv4:10.0.0.5
 dtmf-relay rtp-nte
!
dial-peer voice 5400 voip
 destination-pattern 55554..
 session protocol sipv2
 session target ipv4:10.0.0.5
 dtmf-relay rtp-nte
!
dial-peer voice 2100 voip
 destination-pattern 55521..
 session protocol sipv2
 session target ipv4:172.18.193.88
 dtmf-relay rtp-nte
 codec g711ulaw
!
dial-peer voice 2200 voip
 destination-pattern 55522..
 session protocol sipv2
 session target ipv4:172.18.207.10:5060
dtmf-relay rtp-nte
!
gateway
!
sip-ua
 retry invite 3
 retry response 3
 retry bye 3
 retry cancel 3
 retry prack 3
 retry comet 3
 retry rel1xx 3
 retry notify 3
 timers trying 750
!
!
line con 0
 exec-timeout 0 0
 transport preferred none
line aux 0
line vty 0 4
 exec-timeout 0 0
 password lab
login
 transport preferred none
!
ntp clock-period 17180086
ntp server 172.68.10.150 prefer
!
end
```

Enabling SIP Debug Output Filtering: Example

The following example shows how to enable SIP debug output filtering. In the following example, the match-list configuration, enabled debugs, and debug output for a Cisco AS5300 are shown.

Note When the exact match condition is used for SIP call debug filtering, all related debug output is filtered until all conditions in the match list are explicitly met.

```
Router# debug condition match-list 1 exact-match
Router# debug ccsip all

Router# show debug

CCSIP SPI:SIP Call Statistics tracing is enabled       (filter is ON)
CCSIP SPI:SIP Call Message tracing is enabled  (filter is ON)
CCSIP SPI:SIP Call State Machine tracing is enabled    (filter is ON)
CCSIP SPI:SIP Call Events tracing is enabled   (filter is ON)
CCSIP SPI:SIP error debug tracing is enabled   (filter is ON)
CCSIP SPI:SIP info debug tracing is enabled    (filter is ON)
CCSIP SPI:SIP media debug tracing is enabled   (filter is ON)
CCSIP SPI:SIP Call preauth tracing is enabled  (filter is ON)

Router# Debug filtering is now on
Building configuration...
!
!
!
call filter match-list 1 voice
    incoming called-number 4085551221
!
end
```

The following lines show partial debug output with the filter on. Some debug output generated during a call may not have GUID information. These debugs, represented by /xxxxxxxxxxx/ characters in the debug header, are not filtered and always appear.

```
Sep  2 13:11:05.395://-1/xxxxxxxxxxx/SIP/Error/httpish_msg_process_network_msg:Content
Length 222, Bytes Remaining 233
Sep  2 13:11:05.395://-1/xxxxxxxxxxx/SIP/Info/HandleUdpSocketReads:Msg enqueued for SPI
with IP addr:10.102.16.214:56587
Sep  2 13:11:05.395://-1/xxxxxxxxxxx/SIP/Info/sipSPISetDateHeader:Converting TimeZone EST
to SIP default timezone = GMT
Sep  2 13:11:05.399://-1/xxxxxxxxxxx/SIP/Error/sipSPI_validate_own_ip_addr:ReqLine IP
addr does not match with host IP addr
Sep  2 13:11:05.399://-1/xxxxxxxxxxx/SIP/Event/sipSPIEventInfo:Queued event from SIP SPI
:SIPSPI_EV_SEND_MESSAGE
```

The next lines show the debug filtering turned off.

```
Router# no debug condition match-list 1
Router# show debug

CCSIP SPI:SIP Call Statistics tracing is enabled       (filter is OFF)
CCSIP SPI:SIP Call Message tracing is enabled  (filter is OFF)
CCSIP SPI:SIP Call State Machine tracing is enabled    (filter is OFF)
CCSIP SPI:SIP Call Events tracing is enabled   (filter is OFF)
CCSIP SPI:SIP error debug tracing is enabled   (filter is OFF)
CCSIP SPI:SIP info debug tracing is enabled    (filter is OFF)
CCSIP SPI:SIP media debug tracing is enabled   (filter is OFF)
CCSIP SPI:SIP Call preauth tracing is enabled  (filter is OFF)
```

With filtering turned off, all the debug output is displayed.

```
Sep  2 13:12:31.112://-1/xxxxxxxxxxxx/SIP/Info/HandleUdpSocketReads:Msg enqueued for SPI
with IP addr:10.102.16.214:56589
Sep  2 13:12:31.112://-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Received:
SUBSCRIBE sip:3100802@172.18.193.99:5060 SIP/2.0
Via:SIP/2.0/UDP  172.18.193.100:5060;branch=z9hG4bK1348
From:"3100801" <sip:3100801@172.18.193.100>;tag=19AE8-A6C
To:<sip:3100802@172.18.193.99>;tag=19AE8-A6C
Date:Mon, 30 Oct 2000 04:47:31 GMT
Call-ID:96ACB04F-AD5611D4-800894FA-5B905DB@172.18.193.100
Supported:timer,100rel
Min-SE: 1800
Cisco-Guid:2496939846-2908099028-2147680272-2073165500
User-Agent:Cisco-SIPGateway/IOS-12.x
Allow:INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, COMET, REFER, SUBSCRIBE, NOTIFY, INFO,
UPDATE
CSeq:101 SUBSCRIBE
Max-Forwards:6
Remote-Party-ID:<sip:3100801@172.18.193.100>;party=calling;screen=no;privacy=off
Timestamp:972881251
Contact:<sip:3100801@172.18.193.100:5060>
Expires:60
Allow-Events:telephone-event
Content-Type:application/sdp
Content-Length:233

v=0
o=CiscoSystemsSIP-GW-UserAgent 2904 6070 IN IP4 172.18.193.100
s=SIP Call
c=IN IP4 172.18.193.100
t=0 0
m=audio 18862 RTP/AVP 18 19
c=IN IP4 172.18.193.100
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:19 CN/8000
a=ptime:20

Sep  2 13:12:31.112://-1/94D447468003/SIP/State/sipSPIChangeState:0x631570C8 :State change
from (STATE_NONE, SUBSTATE_NONE)  to (STATE_IDLE, SUBSTATE_NONE)
Sep  2 13:12:31.112://-1/xxxxxxxxxxxx/SIP/Info/sipSPISetDateHeader:Converting TimeZone EST
to SIP default timezone = GMT
Sep  2 13:12:31.116://-1/xxxxxxxxxxxx/SIP/Error/sipSPI_validate_own_ip_addr:ReqLine IP
addr does not match with host IP addr
Sep  2 13:12:31.116://-1/xxxxxxxxxxxx/SIP/Event/sipSPIEventInfo:Queued event from SIP SPI
:SIPSPI_EV_SEND_MESSAGE
Sep  2 13:12:31.116://-1/94D447468003/SIP/Info/sipmwiEventCleanup:Removing CCB with mwi
clientID(3100802)
Sep  2 13:12:31.116://-1/94D447468003/SIP/State/sipSPIChangeState:0x631570C8 :State change
from (STATE_IDLE, SUBSTATE_NONE)  to (STATE_DEAD, SUBSTATE_NONE)
Sep  2 13:12:31.116://-1/94D447468003/SIP/Info/ccsip_deregister_call_filter:Deregistering
call from GCFM
Sep  2 13:12:31.120://-1/94D447468003/SIP/Info/sipSPIFlushEventBufferQueue:There are 0
events on the internal queue that are going to be free'd
Sep  2 13:12:31.120://-1/94D447468003/SIP/Info/sipSPIUfreeOneCCB:Freeing ccb 631570C8
Sep  2 13:12:31.120://-1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg:
Sent:
SIP/2.0 489 Bad Event - 'Missing Event:field'

Via:SIP/2.0/UDP
172.18.193.100:5060;branch=z9hG4bK1348;received=10.102.16.214
```

```
From:"3100801" <sip:3100801@172.18.193.100>;tag=19AE8-A6C

To:<sip:3100802@172.18.193.99>;tag=19AE8-A6C

Call-ID:96ACB04F-AD5611D4-800894FA-5B905DB@172.18.193.100

CSeq:101 SUBSCRIBE

Timestamp:972881251

Content-Length:0
```

MGCP Call Centric Debug

The MGCP Call Centric Debug feature enables the filtering of debug output based on selected criteria and this feature also standardizes the format of the MGCP debug header. By sharing a common header format all MGCP debug information for a single call can be identified and correlated across the various layers in Cisco IOS software. Filtering debug output reduces extraneous information displayed on the console port, making it easier to locate the correct information and reducing the impact to platform performance, while mitigating lost data because of buffer limits.

This section contains the following information:

Restrictions for MGCP Call Centric Debug

Filtering conditions that are set for other Cisco IOS modules also impact the debug output for the MGCP module.

Information About MGCP Call Centric Debug

To use the MGCP Call Centric Debug feature, you should understand the following concepts:

MGCP Debug Commands that Support Debug Filtering

- **debug mgcp all**
- **debug mgcp endpoint**
- **debug mgcp endptdb**
- **debug mgcp errors**
- **debug mgcp events**
- **debug mgcp gcfm**
- **debug mgcp inout**
- **debug mgcp media**
- **debug mgcp src**
- **debug mgcp state**
- **debug mgcp voipcac**

Note Debug filtering is not supported for the **debug mgcp nas**, **debug mgcp packets**, or **debug mgcp parser** commands.

See the *Cisco IOS Debug Command Reference*, Release 12.4T for more information about MGCP debug commands.

Match Conditions for MGCP Debug Filtering

To filter calls, you must first define a list of conditions on which to match. The attributes associated with a call are compared to the configured list of match conditions. Debug output that matches all or some of the conditions in the list is displayed, depending on whether the match criteria is set to either exact or partial match.

The MGCP Call Centric Debug feature supports filtering based on the following conditions:

- Incoming signaling IPv4 local address
- Incoming signaling IPv4 remote address
- Incoming media IPv4 local address
- Incoming media IPv4 remote address
- Incoming dial peer
- Outgoing signaling IPv4 local address
- Outgoing signaling IPv4 remote address
- Outgoing media IPv4 local address
- Outgoing media IPv4 remote address

See the "Creating Match Lists for MGCP Filtering Conditions" section on page 52 for information on configuring match conditions for filtering MGCP calls.

Trace Levels for MGCP Debug Output

The MGCP Call Centric Debug feature introduces trace levels for MGCP debug output. Trace levels allow you to control the amount of information that is displayed by debug commands based on the importance of the content. Trace levels are associated with priority levels that categorize MGCP debug output depending on the information it contains. The output for each debug command is categorized within three priority levels: high, medium, and low.

The following trace levels can be selected to indicate the priority of the information that is displayed:

- Critical—Displays only MGCP debug information marked as high priority.
- Moderate—Displays MGCP debug information marked as medium or high priority.
- Verbose—Displays all MGCP debug information. This is the default level.

Note The **debug mgcp errors** and **debug mgcp packets** commands do not support trace levels. Their debug output is set to the highest priority and is displayed for all trace level values.

You can set the desired trace level for an MGCP debug session by using the **tracelevel** keyword in individual MGCP debug commands or by setting a global trace level using the **debug mgcp tracelevel-default** command.

Note Setting the trace level for an endpoint using the **mgcp debug endpoint** command is independent of the global trace level. The endpoint level takes precedence over the global level. For example, the **debug mgcp event tracelevel moderate** command used with the **debug mgcp endpoint aaln/S2/SU0/0 event tracelevel verbose** command sets the trace level to verbose for that specific endpoint while all of the other endpoints have event debugs set at a moderate level. If the global debug is disabled, the per-endpoint debug remains enabled and vice versa.

Tips on Collecting Debug Output

Logging debug output to the console has disadvantages such as being slower and dropping data more easily than logging to a buffer. Collecting debug information by logging output to a buffer instead of the console reduces the impact to gateway performance and decreases the incidence of dropped data.

To log debug output to a buffer instead of the console, use the **no logging console** and **logging buffered** commands. These commands can only be used, however, if there is enough memory available on the gateway to create a large enough buffer to collect the debug output. To display debug output that was collected and sent to the configured buffer, use the **show logging** command.

Logging debug output to the console may also consume an excessive amount of CPU resources if the **logging console guaranteed** command is enabled, which is the default setting. It is recommended that you disable this functionality by using the **no logging console guaranteed** command when sending debug output to the console.

You may also want to use the **service timestamps debug** and **service timestamps log** commands to control how the timestamps are displayed in the debug output.

How to Enable MGCP Call Centric Debug

This section contains the following procedures:

Modifying the Debug Header Format for MGCP Debug Output

Perform this procedure to modify the standardized header format for MGCP debug output. Debug output is correlated based on this unique header which is common to all debugs belonging to the same call.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice call debug** {**full-guid** | **short-header**}
4. **mgcp debug-header**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

Step 3	`voice call debug {full-guid | short-header}`
	
	Example:
	`Router(config)# voice call debug full-guid`
		(Optional) Specifies the full GUID or short header for debug output.
		• **full-guid**—Displays the GUID in a 16-byte header.
		Note Using the **no voice call debug full-guid** command displays the short 6-byte header.
		• **short-header**—Displays the CallEntry ID in the header without displaying the GUID or module-specific parameters. This is the default.
		Note For more information, see the "Debug Header Format" section on page 18.

	Command or Action	Purpose

Step 4	`mgcp debug-header`
	
	Example:
	`Router(config)# mgcp debug-header`
		(Optional) Enables the MGCP module-dependent information in the debug header.
		
		Note This command is enabled by default. This step is included to illustrate how to enable the command if it was previously disabled.

Step 5	`exit`
	
	Example:
	`Router(config)# exit`
		Exits to privileged EXEC mode.

Creating Match Lists for MGCP Filtering Conditions

Perform this procedure to define match conditions that are used for filtering MGCP calls.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call filter match-list** *number* **voice**
4. **incoming signaling** {**local** | **remote**} **ipv4** *ip-address*
5. **incoming media** {**local** | **remote**} **ipv4** *ip-address*
6. **incoming dialpeer** *tag*
7. **outgoing signaling** {**local** | **remote**} **ipv4** *ip-address*
8. **outgoing media** {**local** | **remote**} **ipv4** *ip-address*
9. **end**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

	Command or Action	Purpose
Step 3	**call filter match-list** *number* **voice** **Example:** `Router(config)# call filter match-list 1 voice`	Enters call filter match list configuration mode to define the filter conditions. • *number*—Numeric label that uniquely identifies the match list. Range is 1 to 16. **Note** At least one of the following optional parameters for call filtering (Step 4 to Step 8) must be configured.
Step 4	**incoming signaling** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# incoming signaling local ipv4 192.168.10.255`	(Optional) Specifies the incoming signaling IPv4 address. • **local**—Local voice gateway. • **remote**—MGCP call agent. • *ip-address*—IP address of the local voice gateway or remote call agent.
Step 5	**incoming media** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# incoming media local ipv4 192.168.10.255`	(Optional) Specifies the incoming media IPv4 address for the voice gateway receiving the media stream. • **local**—Local voice gateway. • **remote**—Remote voice gateway. • *ip-address*—IP address of the local or remote voice gateway.
Step 6	**incoming dialpeer** *tag* **Example:** `Router(conf-call-filter-mlist)# incoming dialpeer 14`	(Optional) Specifies the incoming telephony dial peer to be filtered. • *tag*—Digits that define a specific dial peer. Range is 1 to 2147483647. **Note** Telephony dial peers are configured using the **dial-peer voice** command.
Step 7	**outgoing signaling** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# outgoing signaling local ipv4 192.168.10.255`	(Optional) Specifies the outgoing signaling IPv4 address for the gatekeeper managing the signaling. • **local**—Local voice gateway. • **remote**—MGCP call agent. • *ip-address*—IP address of the local gateway or remote call agent.
Step 8	**outgoing media** {**local** | **remote**} **ipv4** *ip-address* **Example:** `Router(conf-call-filter-mlist)# outgoing media local ipv4 192.168.10.255`	(Optional) Specifies the outgoing media IPv4 address for the voice gateway receiving the media stream. • **local**—Local voice gateway. • **remote**—Remote voice gateway. • *ip-address*—IP address of the local or remote voice gateway.
Step 9	**end** **Example:** `Router(conf-call-filter-mlist)# end`	Exits to privileged EXEC mode.

Enabling MGCP Debug Filtering Using Match Lists

Perform this procedure to enable the match conditions for filtering MGCP debug output.

Prerequisites

The filtering conditions for the debug output must be set as described in the "Creating Match Lists for MGCP Filtering Conditions" section on page 52.

Restrictions

- The **debug mgcp nas**, **debug mgcp packets**, and **debug mgcp parser** commands do not support debug filtering.
- Debug output that is outside the context of a call, for example, RSIP, audit, and some endpoint database information does not support filtering.

SUMMARY STEPS

1. **enable**
2. **debug condition match-list** *number* {**exact-match** | **partial-match**}
3. **debug mgcp** {**all** | **endpoint** | **endptdb** | **errors** | **events** | **gcfm** | **media** | **src** | **state** | **voipcac**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug condition match-list** *number* {**exact-match** | **partial-match**} **Example:** `Router# debug condition match-list 1 exact-match`	Enables the filter match list for the set conditions. • *number*—Numeric label that uniquely identifies the match list. Range is 1 to 16. This number is set using the **call filter match-list** command. • **exact-match**—All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise. • **partial-match**—No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because a large amount of debug output is generated before matches explicitly fail. **Note** This command impacts all enabled debug commands that support call filtering.

Step 3	**debug mgcp** {**all** | **endpoint** | **endptdb** | **errors** | **events** | **gcfm** | **inout** | **media** | **src** | **state** | **voipcac**}
	
	Example:
	`Router# debug mgcp errors`
	`Router# debug mgcp events`
	`Router# debug mgcp endpoint aaln/s2/su0/1/1/10`
	`Router# debug mgcp media`
		Enables the appropriate MGCP debug commands.
		• *See the Cisco IOS Debug Command Reference* for detailed descriptions of these debug commands.
		Note When enabling MGCP debug commands, you can also set a trace level to further filter output based on the importance of the information. For information, see the "Enabling MGCP Debug Trace Levels" section on page 56.

Verifying the MGCP Debug Filtering Configuration

To verify debug filtering conditions, use the following commands:

- **show debug**—Displays the debugs that are enabled.
- **show call filter components**—Displays the components that register internally with the filtering module. This command shows which components are registered with the GCFM, which is the internal module that controls which components are filtered.
- **show call filter match-list**—Displays the criteria set for the specified match list. It shows a list of all the match lists, shows which ones are enabled, and shows whether they are enabled for partial or exact matching.

See the *Cisco IOS Voice Command Reference* for more information about these commands.

Enabling MGCP Debug Trace Levels

Perform this procedure to enable trace levels for restricting MGCP debug output based on the priority of the information.

Restrictions

Trace levels are not supported for MGCP errors or packets debugging because all of the output from these commands is set to high priority.

SUMMARY STEPS

1. **enable**
2. **debug mgcp tracelevel-default** {**critical** | **moderate** | **verbose**}
3. **debug mgcp endpoint** *endpoint-name* {{**all** | **events** | **media**} [**tracelevel** {**critical** | **moderate** | **verbose**}] | {**errors** | **packets**}}
4. **debug mgcp** {**all** | **endptdb** | **events** | **gcfm** | **inout** | **media** | **nas** | **parser** | **src** | **state** | **voipcac**} [**tracelevel** {**critical** | **moderate** | **verbose**}]

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`debug mgcp tracelevel-default {critical | moderate | verbose}`
	
	Example:
	`Router# debug mgcp tracelevel-default critical`
		(Optional) Enables the trace level globally for all MGCP debug commands and endpoints.
		• **critical**—Only high priority debug information is displayed.
		• **moderate**—Medium and high priority debug information is displayed.
		• **verbose**—All debug information is displayed. This is the default trace level.

	Command or Action	Purpose
Step 3	**debug mgcp endpoint** *endpoint-name* {{**all** | **events** | **media**} [**tracelevel** {**critical** | **moderate** | **verbose**}] | {**errors** | **packets**}} **Example:** `Router# debug mgcp endpoint aaln/s2/su0/1/1/10`	(Optional) Enables the trace level for a specific endpoint for events or media debug commands. • *endpoint-name*—Name of the MGCP endpoint for which to enable debugging. Must be a fully specified and supported endpoint.

Step 4	**debug mgcp** {**all** | **endptdb** | **events** | **gcfm** | **inout** | **media** | **nas** | **parser** | **src** | **state** | **voipcac**} [**tracelevel** {**critical** | **moderate** | **verbose**}]
	
	Example:
	`Router# debug mgcp events tracelevel critical`
	`Router# debug mgcp state tracelevel moderate`
	`Router# debug mgcp media moderate`
		(Optional) Enables the trace level for a specific MGCP debug command.

Configuration Examples for MGCP Call Centric Debug

This section contains the following examples:

- Match-List Configuration for MGCP Debug Filtering: Example, page 57
- Enabling MGCP Debug Filtering: Example, page 60

Match-List Configuration for MGCP Debug Filtering: Example

The following example shows a configuration with a match list defined to filter MGCP debug output.

```
Router# show running-config

Building configuration...

Current configuration : 2068 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
service internal
!
hostname Router
!
boot-start-marker
boot system flash:Router.ios.bin
boot-end-marker
!
logging buffered 10000000 debugging
enable secret 5 $1$abcd
enable password sample
!
no aaa new-model
!
resource policy
!
no network-clock-participate slot 1
no network-clock-participate slot 2
ip cef
```

```
!
!
!
no ip domain lookup
ip host callagenthost 192.168.1.200
voice-card 1
 no dspfarm
!
voice-card 2
 dspfarm
!
!
!
!
!
!
!
controller T1 1/0
 framing esf
 clock source internal
 linecode b8zs
 ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 1/1
 shutdown
 framing esf
 clock source internal
 linecode b8zs
 ds0-group 0 timeslots 1-24 type none service mgcp
!
!
!
!
interface FastEthernet0/0
 ip address 192.168.1.79 255.255.255.0
 no ip mroute-cache
 speed auto
 half-duplex
 no cdp enable
!
interface FastEthernet0/1
 no ip address
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
!
ip http server
!
snmp-server community public RO
snmp-server enable traps tty
!
!
!
control-plane
!
!
!
call filter match-list 1 voice
 incoming media local ipv4 192.168.1.12
 outgoing media local ipv4 192.168.1.11
!
```

```
voice-port 1/0:0
!
voice-port 1/1:0
!
voice-port 2/0/0
!
voice-port 2/0/1
!
voice-port 2/0/2
!
voice-port 2/0/3
!
voice-port 2/1/0
!
voice-port 2/1/1
!
voice-port 2/1/2
!
voice-port 2/1/3
!
!
mgcp
mgcp call-agent callagenthost 7979 service-type mgcp version 1.0
mgcp package-capability mf-package
mgcp package-capability rtp-package
mgcp package-capability script-package
mgcp sdp simple
!
mgcp profile default
!
!
!
dial-peer voice 211 pots
 service mgcpapp
 port 2/1/1
!
dial-peer voice 213 pots
 service mgcpapp
 port 2/1/3
!
dial-peer voice 210 pots
 service mgcpapp
 port 2/1/0
!
dial-peer voice 200 pots
 service mgcpapp
 port 2/0/0
!
dial-peer voice 212 pots
 service mgcpapp
 port 2/1/2
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 password temp
 login
!
!
end
```

Enabling MGCP Debug Filtering: Example

The following example shows how to enable filtering and trace levels for MGCP debug output.

```
Router# debug condition match-list 1 exact-match
Router# debug mgcp tracelevel-default critical
Router# debug mgcp errors
Media Gateway Control Protocol errors debugging for all endpoints is on

Router# debug mgcp media
Media Gateway Control Protocol media events debugging for all endpoints is on, trace-level
Critical

Router# debug mgcp state tracelevel verbose
Media Gateway Control Protocol state transition debugging for all endpoints is on,
trace-level Verbose

Router# debug mgcp endpoint S1/ds1-0/1 events tracelevel moderate
Media Gateway Control Protocol events debugging for endpoint s1/ds1-0/1 is on, trace-level
Moderate

Router# show debug

MGCP:
  Media Gateway Control Protocol media events debugging is on, trace level Critical
  Media Gateway Control Protocol errors debugging is on
  Media Gateway Control Protocol state transition debugging is on, trace level Verbose
MGCP: Event debugging for endpoint S1/DS1-0/1 is on, tracelevel is Moderate

Router# show call filter match-list

****************************************
call filter match-list 1 voice
****************************************
  incoming media local ipv4  192.168.1.12
  outgoing media local ipv4  192.168.1.11
debug condition match-list is set to EXACT_MATCH
```

Cisco VoIP Internal Error Codes

The Cisco VoIP Internal Error Codes feature generates internal error codes (IECs) for gateway-detected errors that cause the gateway to release or refuse a call. IECs enhance troubleshooting for VoIP networks by helping to determine the source and reason for call termination.

Prerequisites for Cisco VoIP Internal Error Codes

Before this feature can be operational, a basic VoIP network must be configured.

Restrictions for Cisco VoIP Internal Error Codes

- Memory usage increases slightly when this feature is implemented, depending upon the number of subsystems that support IECs and upon the number of error codes defined for each subsystem.
- IECs are reported only in RADIUS accounting records. They are not supported in syslog accounting.

Information About Cisco VoIP Internal Error Codes

To configure the Cisco VoIP Internal Error Codes feature, you should understand the following concepts:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Internal Error Code Notation, page 7

Benefits of Cisco VoIP Internal Error Codes

- Allows the service provider to see the cause of call disconnect in the accounting record.
- Provides enhanced diagnostic and troubleshooting capability for VoIP networks.
- Supports the development of enhanced analysis tools that can determine if patterns of call failures exist.
- Improves network reliability by enabling more effective monitoring and call management.
- Internal Error Code (IEC) reporting has been enhanced in Cisco IOS Release 12.4(4)T to provide better tracking and diagnostic capability for networks, and specifically for gatekeepers.

Feature Design of Cisco VoIP Internal Error Codes

Prior to the implementation of IECs, Q.850-based disconnect cause codes were used to track and diagnose network problems. These cause codes, defined by ITU Recommendation Q.850, were more applicable to traditional PSTN networks than to packet networks and were too generic to be useful for diagnosing and isolating faulty VoIP network components.

The Cisco VoIP Internal Error Codes feature allows an error code to be generated for a gateway-detected error that causes the gateway to release or refuse a call or call attempt. The error may not actually cause the call to fail; for example, in the case of rotary attempts, a subsequent attempt may result in the call completing. The error does not necessarily indicate a problem on the gateway itself, but may be due, for example, to a protocol error detected in a message, or to a timeout while communicating with a nonresponding party. IECs are not generated for normal calls that are released without an error; for example, no answer, busy, and user hangs up.

Each internal error in the voice signaling path that leads to the release of a call is assigned an IEC value. Fields within the IEC identify which network entity and subsystem originated the error, and specify the error code within the subsystem. The IEC mechanism maintains error counters and allows you to use command-line interface (CLI) commands to collect, display, and offload error counters. The CLI also allows you to clear counters. The IEC mechanism also generates a CLI-enabled syslog message and a new RADIUS vendor-specific attribute (VSA) whenever an IEC is generated.

The IEC feature supports a mechanism for enforcing disconnect cause code consistency for internal errors by providing a configurable mapping table to translate the IEC error category to an appropriate disconnect cause code.

In addition to generating IECs, this feature set makes use of enhanced release source indicators (RSIs) to report gatekeeper-released and route server-released calls. For more information on RSIs, refer to *Call Release Source Reporting in Gateway-Generated Accounting Records.*

Note IECs are not generated for the following types of calls: VoiceXML, fax, MGCP or SGCP, and SS7 continuity (COT).

IEC Reporting

Cisco implements IEC reporting by logging IEC values into the following records:

- VSAs in RADIUS accounting records
- Call history records
- Dial Control MIB
- Syslog messages

The gateway sends VSAs in RADIUS accounting stop records. Because each IEC is associated with a call leg, an IEC is reported only in the stop record for one of the legs in a call. VSAs are also sent by the gatekeeper. The gateway collects IECs for all call legs involved in a call and reports them to the gatekeeper, which inserts the IECs in its accounting stop record. In some scenarios, multiple errors may be encountered for a particular call leg; for example, multiple attempts to connect to an alternate endpoint. Up to five IECs may be generated per call.

Because IECs are reported through accounting records, if there is no voice call association or context, no IEC is generated. This scenario occurs, for example, if the gateway receives an ISDN setup message and the ISDN layer fails to allocate resources to process the setup message. In this instance there is no indication to the Voice Telephony Service Provider (VTSP) layer and no creation of a call-leg or call-history record, so no IEC is generated.

Gatekeeper Behavior and Cisco VoIP Internal Error Codes

Gatekeeper behavior for RADIUS accounting for start and stop records changes with the introduction of VoIP internal error codes. Prior to the Cisco VoIP Internal Error Codes feature, the gatekeeper generated two records for intrazone calls: one start record, based on the originating admission reject (ARJ) message, and one stop record, based on the first incoming disengage request (DRQ) message. This limitation resulted in data from the DRQ of the other gateway not being included in the accounting.

For intrazone calls the Cisco VoIP Internal Error Codes feature allows the gatekeeper to generate start and stop records based on each ARQ and DRQ; that is, two start and stop records are generated for each call.

Gatekeeper IEC Logging

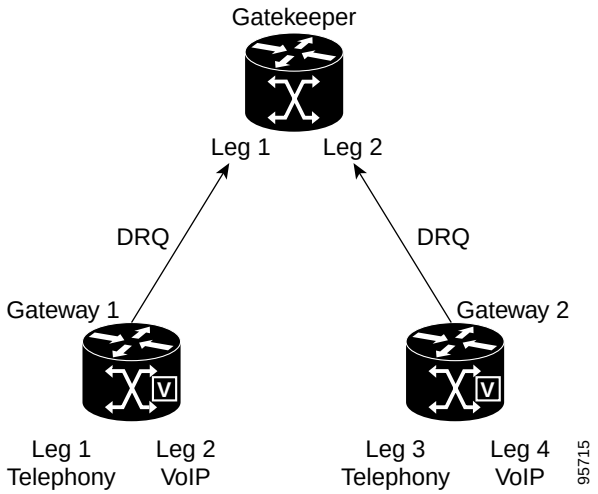
The gateway collects IECs for all call legs involved in a call and reports them to the gatekeeper in a DRQ message during call release. The gateway also sends RSI information in the DRQ message. The gatekeeper then logs the RSI and IEC information in RADIUS accounting stop records.

IEC Differences in Gateway and Gatekeeper Accounting

On the gateway an IEC is logged in to a stop accounting record for the call leg that encountered the error. If an error occurred in the VTSP call leg, an IEC is logged in the telephony stop record; no IEC is recorded in the VoIP stop record, and vice versa.

Figure 9 shows the differences between gateway and gatekeeper accounting. On the gatekeeper, the two call legs, telephony and VoIP, are treated as one call leg, with IEC information merged from both originating gateway (Gateway 1) call legs. The DRQ message to the gatekeeper therefore contains IECs combined from both the telephony and VoIP call legs for a particular call. From the gatekeeper perspective, the second call leg is the terminating gateway (Gateway 2) call leg. This call leg records accounting information received as well.

Figure 9 Differences in Gateway and Gatekeeper Accounting



IEC and RSI Format in DRQ

IEC and RSI information is communicated in the RasnonStdUsageInformation field in the usageInformation information element (IE) of the DRQ message. The following example shows a partial DRQ message:

```
value RasnonStdUsageInformation ::=
    {
      rasMessageSpecificData drqRasnonStdUsageData :
      {
        callReleaseSource internalReleaseInVoipLeg :NULL
        iecInfo
        {
          '10105480022C0000'H
```

The 64-bit IECs are communicated as an array of eight characters (of size eight bits). The callReleaseSource is communicated as an enumerated value.

Gatekeeper-Initiated Release Scenario

Prior to Cisco IOS Release 12.4(4)T, if the gatekeeper forcefully initiates a release for an active call by sending a DRQ message, then no IEC is generated. The IEC feature does not support gatekeeper-generated IECs. The ReleaseSource VSA for this scenario indicates a value of gatekeeper; however, there is no InternalErrorCode VSA.

Starting with Cisco IOS Release 12.4(4)T, the capability was expanded so that gatekeeper-detected errors that cause the gateway to release or refuse a call are covered. The IEC generated at the gatekeeper is sent in the ARJ/DRQ RAS message to the gateway. The gateway then sends the IEC in a RADIUS accounting record. The gatekeeper IEC clearly identifies:

- Physical network entity that encountered the error
- Type of error (category or class)
- Subsystem within that entity
- Subsystem-defined error code
- Private diagnostic code to allow developers to better pinpoint the software point of failure

Release Source Extension

Prior to Cisco IOS Release 12.3(2)T, both gatekeeper and Gatekeeper Transaction Message Protocol (GKTMP) server-released calls were treated as gatekeeper-released calls, and were indicated by an RSI value of external call control agent. The Cisoc IOS Release 12.3(2)T version provides extended release source values for GKTMP server and gatekeeper. RSI information is passed in the NonStandardUsageData parameter field of the ARJ and DRQ messages from the gatekeeper to gateway during call tear down.

There is no change in the GKTMP interface; instead the context of the release scenario is used to determine the RSI value at the gatekeeper. For example, the receipt of a RESPONSE.ARJ message from the route server results in an RSI value of external gktmp server. Similarly, a forced release from gatekeeper using the **clear h323 gatekeeper call** command results in the RSI value of gatekeeper.

Release Source Values

With respect to a single network, the following release sources are possible:

- Calling party located in the PSTN
- Calling party located in the VoIP network
- Called party located in the PSTN
- Called party located in the VoIP network
- Internal release in a POTS leg
- Internal release in a VoIP leg
- Internal call-control application (for example, Tool Command Language (Tcl) or Voice eXtensible Markup Language (VXML) script
- Internal release in VoIP authentication, authorization, and accounting (AAA)
- CLI or Man Machine Language (MML)
- External RADIUS server
- External network management application
- External call control agent
- Gatekeeper
- External GKTMP server

Obtaining IECs

Choose one or more of the following options to obtain IEC information:

- Display IECs as they are encountered in real time by enabling syslog messages. The IEC is not included in syslog-accounting records. For more information on enabling syslog messages, refer to the chapter “Task 2. Enabling Syslog” of *Enabling Management Protocols: NTP, SNMP, and Syslog*.
- Display running and interval IEC counters, and IEC descriptor strings using CLI commands.
- Export IEC counts to a specified server. For more information, refer to *Voice Call Performance Statistics on Cisco Gateways.*
- Retrieve IEC and RSI information using Tcl IVR 2.0 scripts. For more information on using Tcl scripts with the IEC feature, refer to *Supplemental Tcl IVR API Version 2.0 Programmer’s Guide.*

If you use call detail recording (CDR) templates to filter VSAs that are included in accounting records to the RADIUS server, you must add the IEC VSA to the CDR template if you want to display IEC VSAs.

Sample IEC Syslog Message

The following example shows an IEC-generated syslog message:

```
Oct 14 17:13:21.534:%VOICE_IEC-3-GW:CCAPI:Internal Error (Trunk-group select
 fail):IEC=1.1.182.1.23.8 on callID 62 GUID=11C79B82DECF11D68044C61A8D4F75E3
```

Sample IEC Syslog Message for Gatekeeper

The following example shows an IEC-generated system logging (syslog) message for gatekeeper:

```
Oct 14 17:13:21.534:%VOICE_IEC-3-GK:Internal Error (“DRQ in progress”):IEC=1.2.182.1.23.0
on ConfID 243 GUID=123a2b0912345678
```

If there is no call leg context, the ConfID is -1, and the GUID field is blank.

Sample RADIUS VSA Internal Error Code

The following example shows a partial RADIUS stop accounting record for an IEC:

```
[Vendor 9/1] cisco-avpair = "internal-error-code=1.1.179.2.37.0"
```

Sample Call History Record

The **show call history voice** command displays VSA information in the following format:

```
InternalErrorCode=1.1.128.7.47.0
```

Sample Dial Control MIB Entry

The IEC entry is controlled by the following indexes:

- cCallHistoryIndex, which indicates IECs related to a specific call history record.
- cCallHistoryIecIndex, which is used if there is more than one IEC for a call history record.

The following example shows a partial Dial Control MIB table entry for an IEC:

```
CCallHistoryIecEntry ::=
        SEQUENCE {
            cCallHistoryIecIndex         Unsigned32,
            cCallHistoryIec              SnmpAdminString
        }
```

The following example shows the use of the management tool command **getmany** to obtain the IEC:

```
getmany 10.7.102.32 cCallHistoryIec
cCallHistoryIec.5.1 = 1.1.180.1.26.0

getmany 10.7.102.32 cCallHistory
cCallHistorySetupTime.5 = 8540739
cCallHistoryPeerAddress.5 = 4085550190
cCallHistoryPeerSubAddress.5 =
cCallHistoryPeerId.5 = 1112224
cCallHistoryPeerIfIndex.5 = 213
cCallHistoryLogicalIfIndex.5 = 108
cCallHistoryDisconnectCause.5 = 3F
cCallHistoryDisconnectText.5 = service or option not available, unspecified (63)
cCallHistoryConnectTime.5 = 0
cCallHistoryDisconnectTime.5 = 8540740
cCallHistoryCallOrigin.5 = answer(2)
cCallHistoryChargedUnits.5 = 0
cCallHistoryInfoType.5 = speech(2)
cCallHistoryTransmitPackets.5 = 0
cCallHistoryTransmitBytes.5 = 0
```

```
cCallHistoryReceivePackets.5 = 0
cCallHistoryReceiveBytes.5 = 0
cCallHistoryReleaseSrc.5 = calledPartyInVoip(4)
cCallHistoryIec.5.1 = 1.1.180.1.26.0
```

In the preceding example, 5 is the index of the call history record and 1 is the index of the IEC for that record.

The following example shows the use of the indexes and the management tool command **getone** to obtain the IEC directly:

```
getone 10.7.102.32 cCallHistoryIec.5.1
cCallHistoryIec.5.1 = 1.1.180.1.26.0
```

Internal Error Code Notation

The IEC value takes the form of a dotted string of decimal numbers: version.entity.category.subsystem.errorcode.diagnosticcode.

Table 11 describes the six fields that identify the components of the IEC.

Table 11 IEC Fields

IEC Field	Field Definition
version	Indicates the IEC version. The value 1 indicates the current version.
entity	Indicates the network physical entity (hardware system) that generated the IEC. The value 1 is assigned to the gateway.
category	Indicates an error category, defined in terms of ITU-based Q.850 cause codes and VoIP network errors.
subsystem	Indicates the specific subsystem within the physical entity where the IEC was generated.
error code	Identifies the error code within the subsystem.
diagnostic code	Indicates a Cisco internal diagnostic value. Report this value to Cisco Technical Assistance Center (TAC).

Entity

The entity field indicates the network signaling entity that generated the IEC. A value of 1 in this field indicates the IEC is generated by the gateway.

Category Codes

Cisco VoIP IEC category codes

Cisco VoIP IEC categories range from 1 to 278, allowing an exact category of error to be specified in the category field of an IEC. With the Cisco VoIP Internal Error Codes feature, the concept of error categories combines and extends the existing Q.850 cause codes to handle VoIP-specific errors as well.

IEC category codes are specified as follows:

- The value range 1 to 127 is equivalent to ITU-based Q.850 cause codes defined for PSTN networks.
- The value range 128 to 278 is defined based on VoIP network errors. A mapping is maintained between these error categories to corresponding Q.850 codes (1 to 127 range).

Note Only the H.323 and Session Initiation Protocol (SIP) subsystems implement an approach to generate disconnect cause codes or Q.850 PSTN cause codes based on error categories. The disconnect cause is chosen based on the mapping from the corresponding error category. You can configure this mapping using CLI. This correspondence of IEC error category and Q.850 disconnect cause is implemented only for SIP and H.323 internal errors, and is not implemented for other subsystems in this release. For more information on SIP and H.323 cause codes, refer to *Internal Cause Code Consistency Between SIP and H.323.*

Table 12 shows the category codes outside the Q.850 range, their descriptions, and the default Q.850 cause code used for each error category. The Q.850 cause codes for these categories can be changed using CLI.

Table 12 VoIP Error Category Codes

Category	Description	Default Q.850 code
128	Destination address resolution failure	3
129	Call setup timeout	102
178	Internal communication error	41
179	External communication Error	41
180	Software error	47
181	Software resources unavailable	47
182	Hardware resources unavailable	47
183	Capability exchange failure	41
184	QoS error	49
185	RTP/RTCP receive timer expired or bearer layer failure	41
186	Signaling socket failure	38
187	Gateway or signaling interface taken out of service	38
228	User denied access to this service	50
278	Media negotiation failure due to nonexisting codec	65

Gatekeeper Category Codes

Cisco gatekeeper IEC categories range from 1 to 24, allowing an exact category of error to be specified in the category field of an IEC. Table 13 shows the category codes for gatekeeper IECs.

Table 13 Gatekeeper IEC Category Codes

Category	Description
1	Called party not registered
2	Invalid permission
3	Request denied
4	Undefined reason
5	Caller not registered
6	Route call to gatekeeper
7	Invalid endpoint ID
8	Resource unavailable
9	Security denial
10	QoS control not supported
11	Incomplete address
12	Alias inconsistent
13	Route call to SCN
14	Exceeds capacity
15	Error while collecting destination
16	Error while collecting PIN
17	Generic data reason
18	Needed feature unsupported
19	Software resource unavailable
20	External communication error
21	Software error
22	Socket failure
23	Normal disconnect
24	Force disconnect

Subsystem Codes

Together the subsystem and error codes pinpoint the exact error that cause the call to be released. IECs are reported for the subsystems defined in Table 14.

Table 14 Subsystem Codes

Subsystem Code	Subsystem	Description
1	CCAPI	Call control messaging layer that sits between the session applications and the signaling-protocol legs.
2	Tcl IVR	Session applications that are scripted in Tcl IVR 2.0.
3	Application Framework (AFW)	Library that implements Tcl verbs and VXML tags. Executes functionality such as placing a call, collecting digits, playing prompts, and so on.
4	Default Session Application (SSAPP)	Formerly the default session application that controls the call when an inbound-matched dial peer is not configured with any application or with application default.
5	H.323	Subsystem that performs call signaling for the H.323 VoIP leg.
7	SIP	Subsystem that performs call signaling for the SIP VoIP leg.
9	VTSP	Subsystem that performs call signaling for the telephony leg.
10	Application Framework Session Application (AFSAPP)	Default session application that controls the call when an inbound-matched dial peer is not configured with any application or with application default.

Error Codes

The Error Code field of the IEC dotted-decimal string value indicates the subsystem-defined error code. Codes 1 through 20 are common to all subsystems and may occur in several places within a subsystem; for these errors, the point of failure can be further isolated by referring to the unique diagnostic code field. Subsystem-specific error codes begin at 21.

Note The diagnostic code field is a Cisco internal code. Report this code to Cisco TAC for troubleshooting assistance.

The following tables, Table 15 through Table 22, are organized by subsystem and show error code values, descriptors, associated explanation, and category codes.

Table 15 Error Codes for Subsystem 1 (CCAPI)

Code	Descriptor	Explanation	Category
1	No memory	Dynamically allocated memory on the gateway is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181

Table 15 Error Codes for Subsystem 1 (CCAPI) (continued)

Code	Descriptor	Explanation	Category
3	CPU high	Call is rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call is rejected because default or configured memory usage threshold has been exceeded.	181
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	There were insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call.	185
9	Invalid arguments	Invalid arguments passed to a function. This condition usually indicates an internal software error.	180
10	Invalid State	Some unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This condition usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	Dial peer connections exceeded	An outbound dial peer could not be used because the configured maximum number of connections for the dial peer had been reached.	181
22	Incompatible number type	An outbound dial peer could not be used because the configured numbering type did not match the type specified in the call.	28
23	Trunk-group select fail	The system failed to select an available interface among the trunk group specified for use by a matching dial peer.	182
24	Caller-ID processing failure	An error occurred in processing caller ID information.	180

Table 15 Error Codes for Subsystem 1 (CCAPI) (continued)

Code	Descriptor	Explanation	Category
25	Resource busy	A resource needed to service the call was busy.	181
26	No application	The system could not find an application to take the incoming call. Check your call application and dial peer configurations.	180
27	Application no longer exists	The event points to a session application that no longer exists and is being discarded.	180
28	Incoming loop	An incoming call setup indication was received, bearing the same globally unique identifier (GUID) as a call in existence. The call is being rejected because a loop is suspected.	180
29	Call spike threshold	An incoming call was rejected because configured call spike thresholds were exceeded.	181
30	Inbound dial peer blocked	A matched dial peer could not be used to find an inbound application because the permission setting on it blocked its use as an inbound dial peer. As a result, no application could be found to handle the call.	181
31	Outbound dial peer blocked	A matched dial peer could not be used to place the call because the configured permission on it contradicted its use as an outbound dial peer.	181
32	Handoff depth reached	The maximum number of handoffs between applications for a single call has been exceeded. Check your application scripts to make sure there is no infinite loop within the applications.	180
33	Incompatible apps for handoff	A call handoff attempt between applications failed because the applications were incompatible. Tcl IVR 1.0 applications are incompatible with Tcl IVR 2.0 or VXML applications.	180
34	No dial peer interface	A matched dial peer could not be used for the outbound leg because there was no appropriate interface for the dial peer type. This condition may be a software or configuration error. The tag identifier for the problematic dial peer is provided in the diagnostic field (the last component) of the six-part IEC string. Check your dial peer configuration.	180
35	System init error	Some data structure or process that should have been created at system initialization is missing. Report the IEC to customer support.	180

Table 16 Error Codes for Subsystem 2 (Tcl IVR)

Code	Descriptor	Explanation	Category
1	No Memory	Dynamically allocated memory on the gateway is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call is rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call is rejected because default or configured memory usage threshold has been exceeded.	181
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	There were insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call. This is logged by the script when it specifies media_inactivity_err as the IEC to be used for the disconnect.	185
9	Invalid arguments	Invalid arguments were passed to a function. This condition usually indicates an internal software error.	180
10	Invalid state	An unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Please report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187

Table 16 Error Codes for Subsystem 2 (Tcl IVR) (continued)

Code	Descriptor	Explanation	Category
21	Script syntax	An error was detected while parsing a Tcl script. Enable the **debug voip ivr error** command for more detailed information.	180
22	Bad FSM event	A Tcl IVR script specified an unrecognized event in the definition of the finite state machine (FSM). Enable the **debug voip ivr error** command for more detailed information.	180
23	Invalid args in script	A Tcl IVR script specified invalid arguments when invoking a Tcl command procedure. Enable the **debug voip ivr error** command for more detailed information.	180
24	Unsupported infotag	A Tcl script tried to access an unrecognized infotag, or it may have tried to use a recognized infotag in an unsupported mode (for example, issues a **get** command on a set-only infotag or vice versa). Enable the **debug voip ivr error** command for more detailed information.	180
25	Invalid action in script	A Tcl script tried to execute an action or command that was invalid, or invalid given the state it was in. Enable the **debug voip ivr error** command for more detailed information.	180
26	Call blocked by CLI	This call was rejected because it matched the profile defined for calls to be blocked.	228
27	Settlement check failure	An inbound call was rejected because it failed OSP settlement checking, due to one of the following conditions: • An OSP token was required and no valid one was found. • An OSP token was included in the SETUP indication when none was expected.	228
28	vxmldialog failed	The Tcl IVR application failed to initiate the VXML dialog. Turn on VXML debugging for more detailed information.	180
29	Can't set up prompt	A Tcl script terminated execution on failure of the **media play** command because the prompt initialization failed. Possible causes: • A syntax error in the specification of prompt tokens • Misconfiguration of language prompt-file locations. Enable the **debug voip ivr dynamic** and **debug voip ivr error** commands for more detailed information.	47

Table 16 Error Codes for Subsystem 2 (Tcl IVR) (continued)

Code	Descriptor	Explanation	Category
30	Wrong state for media	A Tcl script requested a media operation, for example, play, stop, or seek, on one or more legs that were in a conferenced state, or where there was a VXML dialog active.	180
31	Get infotag failed	A Tcl script terminated because an infotag retrieval failed. Enable the **debug voip ivr error** command for more information.	180
32	Set infotag failed	A Tcl script terminated because an infotag set operation failed. Enable the **debug voip ivr error** command for more information.	180
33	TCL script error	An error was encountered while interpreting a 180 Tcl script. Enable the **debug voip ivr error** command for more information.	180
34	Bad callinfo params	The application was unable to use one of the callinfo parameters for setup; for example, the octet 3 or octet 3a fields, redirect IE, and GUID. Enable the **debug voip ivr error** command for more information.	180
35	Version mismatch	The application could not run because it required an incompatible version of Tcl IVR.	180
36	Media request failed	A Tcl script terminated a call because an error status was reported by the media layer in the ev_media_done event. This indicates a failure in the execution of media play or some other media operation requested by the script. The script may choose to ignore the error, or it can opt to terminate the call, specifying this IEC, media_done_err, as the reason for the disconnect.	181
37	Digit collect failed	The error is logged by the Tcl script when it fails to collect digits in response to a prompt and decides to terminate the call because of the failure. The failure may be normal, that is, the caller did not enter any digits, or it may be due to an actual error in software or hardware. This IEC is logged by the script when it specifies collectdigits_done_err as the IEC associated with the disconnect.	179
38	Accounting conn err	The error code is set by the Tcl script when it terminates the call because it has received an indication that connectivity to the accounting server is lost. This IEC is logged when the script specifies accounting_conn_err as the IEC associated with the disconnect.	179

Table 16 Error Codes for Subsystem 2 (Tcl IVR) (continued)

Code	Descriptor	Explanation	Category
39	Authentication err	The error code is set by the Tcl script when it terminates a call because of error status reported on an ev_authenticate_done event. The script logs this error by specifying authenticate_done_err as the IEC associated with the disconnect.	179
40	Authorization err	The error code is set by the Tcl script when it terminates a call because of error status reported on an ev_authorize_done event. The script logs this error by specifying authorize_done_err as the IEC associated with the disconnect.	179
41	AAA invalid attribute type	The error is logged by the Tcl script when the attribute type in the AAA av pair specified in the script is not supported.	180

Table 17 Error Codes for Subsystem 3 (Application Framework)

Code	Descriptor	Explanation	Category
1	No Memory	Dynamically allocated memory on the gateway is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call was rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call was rejected because default or configured memory usage threshold has been exceeded.	181
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	There were insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call.	185
9	Invalid arguments	Invalid arguments passed to a function. This condition usually indicates an internal software error.	180

Table 17 Error Codes for Subsystem 3 (Application Framework) (continued)

Code	Descriptor	Explanation	Category
10	Invalid State	An unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This condition usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	Leg connections maxed	The maximum number of connections for the leg has been exceeded. An attempt to bridge yet another connection on the leg failed.	180
22	Handoff app not found	The specified target application for a call handoff was not found on the gateway.	180
23	Incompatible protocols	A matched dial peer could not be used for the outbound leg because the gateway cannot translate between the inbound and outbound protocols.	47
24	OSP Fail	OSP settlement checking failed for an outbound call.	228
25	dial peer deleted	A dial peer that was being used for a call setup was deleted (through CLI) before the call could be initiated.	47
26	Interface busy	An outbound dial peer matching this call's parameters specified an interface that was in use and unavailable.	182
27	App can't handoff	An application tried to place an outbound call using a dial peer configured with an outbound application. However, the first application does not support call handoff, so cannot pass the call to the second application.	180
28	Illegal "setup continue"	A Tcl script tried to issue a **leg setup continue** command when a previous **setup continue** command was still outstanding.	180
29	Call blocked by CLI	The call was rejected because it matched the profile defined for calls to be blocked.	228

Table 18 Error Codes for Subsystem 4 (Default Session Application)

Code	Descriptor	Explanation	Category
1	No Memory	Dynamically allocated memory on the gateway is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call rejected because default or configured memory usage threshold has been exceeded.	181
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	Insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call.	185
9	Invalid arguments	Invalid arguments were passed to a function. This condition usually indicates an internal software error.	180
10	Invalid State	An unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This condition usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	Leg connections maxed	A loop was detected while processing a redirected call. The new destination matches a previously seen redirect address.	128

Table 18 Error Codes for Subsystem 4 (Default Session Application) (continued)

Code	Descriptor	Explanation	Category
22	Handoff app not found	Either an OSP token was detected in the setup message, or the dial peer configuration specified that settlement is to be used for this call. However, the default or session application configured to handle this call does not support the OSP protocol. Check the dial peer configuration and ensure that an OSP-capable application is defined.	47
23	Incompatible protocols	The call was rejected because it matched the profile defined for incoming calls to be blocked.	228
24	OSP Fail	An outbound dial peer matching this call's parameters specified an interface that was in use and unavailable.	182
25	dial peer deleted	Either the default application timed out waiting for the user to enter digits for the called number, or an INFO message arrived with zero-length called number.	28
26	Interface busy	The user entered an excessive number of digits for the called number.	28
27	App can't handoff	Digit collection is not supported on the interface or protocol that originated this call.	79
28	Illegal "setup continue"	The number of calls serviced by this gateway has exceeded the total number permitted, as defined by the **call threshold global total-calls** command.	181
29	Call blocked by CLI	The maximum number of redirects (call forwarding) allowed for a call has been exceeded.	128

Table 19 Error Codes for Subsystem 5 (H.323)

Code	Descriptor	Explanation	Category
1	No memory	Dynamically allocated memory on the gateway is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call rejected because default or configured memory usage threshold has been exceeded.	181

Table 19 Error Codes for Subsystem 5 (H.323) (continued)

Code	Descriptor	Explanation	Category
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	There were insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call.	185
9	Invalid arguments	Invalid arguments passed to a function. This condition usually indicates an internal software error.	180
10	Invalid state	An unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This condition usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	H323 interworking error	The H.323 subsystem routinely provides specific IEC information, depending upon the source of an error. This IEC indicates that the exact source of an error is not available in this instance.	127
22	No usr responding, H225 timeout	Timeout occurred waiting for the callproc or alerting messages. The calling party is given the response “No user responding,” and the called party is given the response “Recovery on timer expiry” as specified by Q931.	18
23	No answer from user	Setup was sent; callproc, alert, or progress messages were already received; and timeout occurred waiting for connect message.	19
24	ARQ wait timeout	A timeout occurred while waiting for response for admission request sent to the gatekeeper.	41

Table 19 Error Codes for Subsystem 5 (H.323) (continued)

Code	Descriptor	Explanation	Category
25	BRQ wait timeout	A timeout occurred while waiting for response for bandwidth request sent to the gatekeeper.	41
26	ANNEX E restart remote	The system received an Annex E RESTART message from the remote end. All calls with CRVs corresponding with destination address were cleared.	41
27	H225 invalid msg	The H.225 message was received with one of the following: • An invalid CRV • Parse error • Mandatory IE missing • Message out of sequence • Wrong IE length • Wrong IEC content	95
28	Setup no called no	Received H.225 setup message and the mandatory field, called number, was not present.	96
29	H225 ASN error	The H.225 message received on parsing the H.225 message found an ASN decode error.	100
30	Wait RAS Cfm msg bad	The system received an unexpected message in a state waiting for RAS CFM message.	101
31	ACF, call redirected	In response to the ARQ, the gatekeeper returned 0.0.0.0 as the destination IP address in the Admission confirm (ACF). This is an attempt to redirect the call by the gatekeeper.	128
32	Setup, DNS fail	During an originating call attempt, the DNS/Enum resolution fails.	128
33	Setup no alternate	During call setup attempt using an alternate endpoint, the gatekeeper found that there are no alternate endpoints to try.	128
34	Setup, GW not registered	A new call is not allowed due to RAS not ready, that is, the gateway is not registered to the gatekeeper.	179
35	SETUP, next CRV invalid	Received setup message at terminating endpoint, failed to get a valid unique CRV value.	180
36	TCS encode send	Encoding and sending of terminal capability request failed.	180
37	End session ack send	Encoding and sending of end session acknowledgement PDU failed.	180
38	End session send	Encoding and sending of end session PDU failed.	180

Table 19 Error Codes for Subsystem 5 (H.323) (continued)

Code	Descriptor	Explanation	Category
39	Userinput send	Encoding and sending of user input signal PDU failed.	180
40	Userinput upd send	Encoding and sending of user input signal update PDU failed.	180
41	Userinput alpha send	Encoding and sending of user input alpha signal PDU failed.	180
42	TCS ack fail	The H.245 capability state machine failed to send TCS acknowledgement for the received TCS request.	180
43	TCS rej send fail	The H.245 capability state machine failed to send TCS reject for the received TCS request.	180
44	TCS rel sent	The H.245 capability state machine received a TCS request, and received an internal event to send the TCS release request.	180
45	SETUP send resource fail	During H.225 PDU send operation, an error occurred in memory allocation or socket queue was full.	180
46	ALERT send failed	Encoding and sending of ALERT PDU failed.	180
47	CallProc send failed	Encoding and sending of Call Proceeding PDU failed.	180
48	PROGRESS send failed	Encoding and sending of PROGRESS PDU failed.	180
49	NOTIFY send failed	Encoding and sending of NOTIFY PDU failed.	180
50	INFO send failed	Encoding and sending of INFO PDU failed.	180
51	USER INFO send failed	Encoding and sending of USER INFO PDU failed.	180
52	FACILITY send failed	Encoding and sending of FACILITY PDU failed.	180
53	SUSPEND send failed	Encoding and sending of SUSPEND PDU failed.	180
54	SUSPEND REJ send failed	Encoding and sending of SUSPEND REJECT PDU failed.	180
55	RESUME send failed	Encoding and sending of RESUME PDU failed.	180
56	PASSTHRU send failed	Encoding and sending of PASSTHRU PDU failed.	180
57	CONNECT send failed	Encoding and sending of CONNECT PDU failed.	180
58	SETUP ACK send failed	Encoding and sending of SETUP ACK PDU failed.	180
59	RSCMSM interface unavail	RSCMSM call admission control (CAC) interface unavailable due to resource failure.	181

Table 19 Error Codes for Subsystem 5 (H.323) (continued)

Code	Descriptor	Explanation	Category
60	H245 sock start fail	H.245 listening socket failed to start.	181
61	Call entry no mem	During the outgoing call, a resource failure occurred for call entry data structure.	181
62	Timeout h245 conn	H.245 connection wait timeout occcurred.	183
63	TCS ack wait timeout	In the capability state machine, timer expiry occurred waiting for the TCS ACK message.	183
64	MS status indetermine	In the MSD state machine, the MSD request is received from remote, and the result of master slave status is indeterminate. This status occurs if sent and received random numbers are the same, or if both the local and remote terminal types are same.	183
65	MSD result disagreement	In the MSD state machine, MSD ACK is received from remote end but there is a disagreement in the MSD result.	183
66	MSD/MSD ACK Timeout	The gateway sent the MSD request, but neither the incoming MSD or MSD ACK message was received.	183
67	MSD ACK timeout	The gateway sent the MSD request, the incoming MSD was received from the remote end, and MSD ACK was sent to the remote end in response. The expected MSD ACK message was not received from the remote.	183
68	MSD rej received	In the MSD state machine, the MSD request was sent and the MSD reject was received from the remote end. MSD requests are sent for a fixed maximum number of retries before release.	183
69	MSD rel received	In the MSD state machine, the MSD request was sent, and the MSD release indication was received from the remote end.	183
70	OLC ACK T103 timeout	In the OLC state machine, T103 timer expired waiting for the OLC ACK message in response to the sent OLC message.	183
71	IPIP QoS Failure	Received QoS failure for non sync RSVP on IP-IP gateway, indicating minimum QoS was provided, not best effort.	184
72	BW > config, min QoS not best	The bandwidth in bearer capability exceeds the maximum configured, and minimum QoS was provided, not best effort. This error occurs during build of nonstandard QoS IE for setup or call processing messages.	184

Table 19 Error Codes for Subsystem 5 (H.323) (continued)

Code	Descriptor	Explanation	Category
73	NonStd min QoS not best	Received setup or call processing message with QoS in nonstandard parameter; the remote end did not have enough bandwidth to support RSVP. The acceptable QoS for audio was not best effort, and remote minimum QoS was provided, not best effort.	184
74	RSVP fail treat abort	Received RSVP failure and QoS treatment specifies that the gateway abort the call, because the minimum QoS was not best effort.	184
75	Fast QoS mismatch	Received fast start setup for QoS and remote minimum QoS was not best effort, but desired QoS was best effort.	184
76	Slow QoS mismatch	The H.225 state machine received a slow start H.225 Setup, with no H.245 address in Setup; that was not a sigonly call and remote minimum QoS was not best effort.	184
77	H225, QoS release	Received external QoS release from QoS resource manager for either the outgoing or incoming H.225 QoS call setup request.	184
78	Fallback chk fail	In the H.225 state machine, the fallback check failed.	184
79	H225 chn, sock fail	During H.225 connection establishment, the channel connection failed due to TCP socket error. The session target in the dial peer directly points to the remote VoIP endpoint.	186
80	Alt h225 chn sock fail	During H.225 connection establishment, in a call attempt to an alternate endpoint, the channel connection failed due to TCP socket error.	186
81	H225 chn, sock fail in RAS	During H.225 connection establishment (new connection), the channel connection failed due to TCP socket error. The dial peer has a session target of RAS.	186
82	H245, chn sock fail	During H.245 connection establishment, the channel connection failed due to TCP socket error.	186
83	SETUP send sock fail	An error occurred during the setup PDU send operation on socket connection for H..225. This error occurs under the following conditions: • If the remote IP is a reachable address for pinging, but is not a valid H.323 endpoint. • If there is an ASN.1 encoding error for setup PDU.	186
84	Preauth fail	Preauthentication attempt failed.	228

Table 19 Error Codes for Subsystem 5 (H.323) (continued)

Code	Descriptor	Explanation	Category
85	OLC bandwidth exceeded	Received an OLC with bandwidth requirement that exceeds the configured value for acc-QoS for that media type.	278
86	OLC ind asymmetric codec	Received OLC indication when waiting for OLC ACK; the codecs in OLC did not match. Also the connection attempt was an asymmetric codec retry.	278
87	Received OLC rej	The H.245 state machine received an OLC reject message.	278
88	Fast codec mismatch	The H.225 state machine received an H.225 fast SETUP message during build of an OLC ACK and found that there was no matching codec.	278
89	OLC m/c, rcvd bw rej	The OLC state machine received a bandwidth reject message.	278
90	TCS ACK neg codec none	Received TCS ACK, but the negotiated codec result was none when call type was not passthrough.	278
91	Cap not supported	In the capability state machine, codec capabilities received from the remote end in the incoming TCS are not supported.	278
92	TCS rej received	In the capability state machine, after sending a TCS request, a TCS reject was received from the remote end.	278
93	Negotiated codec/T-man none	There was no negotiated codec or DTMF relayed mode based on the local or remote capabilities.	278
94	MSD send fail	Encoding and sending of Master Slave Request failed.	180

Table 20 Error Codes for Subsystem 7 (SIP)

Code	Descriptor	Explanation	Category
1	No memory	Dynamically allocated memory on the gateway is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call rejected because default or configured memory usage threshold has been exceeded.	181

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	Insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call.	185
9	Invalid arguments	Invalid arguments passed to a function. This condition usually indicates an internal software error.	180
10	Invalid state	An unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This condition usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	SIP interworking error	The CCSIP subsystem provides IEC information depending upon the source of an error. This IEC indicates that specific information was not available. Use SIP debug tools to help in troubleshooting.	127
22	Hold/Retrieve Timeout	A call was placed on hold with a configurable timer started, and a timeout occurred for the retrieve operation.	41
23	Request, CallID unused	An incoming request message was received with a CallID that is not currently in use; that is, there was a mismatch in associating CallID with the current call control block.	95
24	2xx, dest SDP null	An INVITE was sent, a 2xx response was received but the destination Session Description Protocol (SDP) body was unavailable.	96

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
25	ACK, dest SDP null	An ACK request was received but the destination SDP body was unavailable for the delayed media call.	96
26	2xx, dest SDP null	A mid-call INVITE was sent, a 2xx response was received, but the destination SDP body was unavailable.	96
27	Redirect contact null	The SIP contact header was missing in incoming SIP redirect (3xx) or 485 messages.	96
28	Request, missing From/To	An incoming request message was received with the following conditions: • From, to, or both mandatory fields were missing • There was an error in parsing the from and to fields	96
29	Request, missing Via	An incoming request message was received, and the mandatory field Via was missing.	96
30	Request, missing CSeq	An incoming request message was received, and the mandatory field CSeq was missing.	96
31	Request, missing Contact	An incoming request message was received, and the mandatory field Contact was missing.	96
32	Request, unknown method	An incoming request message was received with an unknown or invalid SIP method.	97
33	Request, Version bad	An incoming request message was received with a SIP version that was not supported on the user agent.	97
34	18x, invalid disptype	Invalid or unsupported Content-Disposition with mandatory handling was received in an 18x session progress message.	100
35	INVITE, invalid IE content	INVITE with either invalid header contents, SDP, or VIA parameters was received.	100
36	Request, parse Via	An incoming request message was received and encountered an error in parsing the Via field.	100
37	Request, parse CSeq	An incoming request message was received that generated an error in parsing the CSeq field.	100
38	Request, parse Contact	An incoming request message was received that generated an error in parsing the Contact field.	100
39	Request, extension bad	An incoming request message was received with a Require header field containing an option tag with an unsupported extension.	100

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
40	Request, Record-Route bad	An incoming request message was received with a Record-Route header field in a malformed format.	100
41	Request, Diversion bad	An incoming request message was received with a Diversion header field in a malformed format.	100
42	Sent INVITE, bad msg	An unknown SIP response message was received while the system was waiting for an INVITE response.	101
43	Bad msg for state	During an outgoing resource reservation state, an unexpected SIP response message was received for the current call state.	101
44	Session trgt null	During an outgoing call, a session target found null.	128
45	Session trgt parse	During an outgoing call, a session target parse failed.	128
46	Session trgt invalid	During an outgoing call, an invalid session target type occurred.	128
47	DNS query fail	For an outgoing call, a DNS lookup of session target failed.	128
48	INVITE, DNS qry fail	A failure response, rcvd target addr null, was received for the DNS query that was sent to resolve the contact in the received invite/FQDN in SDP.	128
49	ACK, DNS qry fail	A failure response, rcvd target addr null, was received for the DNS query that was sent to resolve the contact in the received FQDN message in SDP after the 200 OK message was sent	128
50	MID-INVITE, dns qry fail	A failure response, rcvd target addr null, was received for the DNS query that was sent to resolve the contact or SDP FQDN message in the received mid-INVITE request.	128
51	100, DNS qry fail	DNS lookup failure for the Contact header/FQDN message that was received in the SIP response message.	128
52	DNS qry fail	A failure response, rcvd target addr null, was received for the DNS query that was sent for contact resolution, after a QoS progress message has been sent.	128
53	3xx, redirect loop	Upon the system receiving a 3xx response on an outbound call during redirect procedure, a redirect loop was encountered.	128

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
54	3xx, redirect max	Upon the system receiving a 3xx response on an outbound call during redirect procedure, the maximum number of redirects was exceeded.	128
55	3xx, redirect exhaust	Upon the system receiving a 3xx response on an outbound call during redirect procedure, all contact choices were exhausted.	128
56	Enum resolution fail	A failure response, rcvd contact list null, was received for the query that was sent for enum resolution.	128
57	Contact not resolved	The INVITE Contact or Record Route was not resolved to an IP address and port due to one of the following: • The user answered the call. • A loopback event was received from the session application during the connection attempt.	128
58	Setup conn timeout	No response was received for the socket connection request.	129
59	1xx wait timeout	Retries were exhausted for sending INVITEs while waiting for 1xx response, and no redirect information was available.	129
60	200 wait timeout	Retries were exhausted for PRACK retransmission.	129
61	200 wait timeout	Retries were exhausted for COMET retransmission.	129
62	PRACK wait timeout	Retries were exhausted for sending rel1xx messages and waiting for PRACK.	129
63	PRACK wait timeout & state bad	This condition occurs when the system tries to resend the rel1xx while waiting for a PRACK message, but the call state is wrong.	129
64	Session app rsp timeout	An INVITE request was received and timeout occurred while the system waited for a response to the SETUP sent from the session application.	129
65	1xx wait timeout	Retries were exhausted for sending midcall INVITE, while waiting for 1xx response and not trying DNS.	129
66	ACK wait timeout	Retries were exhausted after sending 200 OK message and waiting for ACK.	129
67	ACK wait timeout	Error occurs if the connection attempt is in an active state and ACK is not received after retries were exhausted sending 200 OK for the initial incoming INVITE.	129

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
68	200 wait timeout	Retries were exhausted sending INVITE and waiting for the 200 OK.	129
69	Xfer 2xx wait timeout	Failed call transfer, system timed out while waiting response for NOTIFY request.	129
70	Connect wait timeout	A timeout occurred after receiving a 200 OK in response to an INVITE and trying to request a UDP/TCP connection to the endpoint specified in the Contact or Record Route headers in order to send the ACK.	129
71	Info Req wait timeout	Timeout occurred while waiting for the Info request on a UDP/ TCP connection. Can occur for user agent client (UAC) or user agent server (UAS).	129
72	Send 200, rsrc fail	Received invite request, resource error in sending 200 OK response.	180
73	Send 200, rsrc fail	Received PRACK message, resource error in sending 200 OK response.	180
74	Send PRACK, rsrc fail	Sending PRACK message failed.	180
75	Send COMET, rsrc fail	Sending COMET message failed during retransmission.	180
76	Send 183, rsrc fail	Sending 183 (progress) response message failed during transmission.	180
77	Send 180, rsrc fail	Sending of 180 response message failed during transmission.	180
78	Rcvd 3xx, contact parse	Internal error or malformed Contact header encountered during SIP redirect (3xx) response processing.	180
79	Rsrc process media	Resource failure during processing of the media changes.	180
80	Err launch dns	Encountered a resource error in launching a DNS query.	180
81	Err reinserting ccb	Resource error in reinserting the associated call control block into table.	180
82	INVITE send fail	A send operation for invite request failed.	180
83	NOTIFY send fail	A send operation for notify request failed.	180
84	ACK send fail	A send operation for ACK request failed.	180
85	REFER send fail	A send operation for refer request failed.	180
86	REFER response send fail	A send operation for refer response failed.	180
87	Call Hold fail	A hold operation failed on an active call while resending the invite to the peer.	180

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
88	RSCMSM interface unavail	The voice CPU and memory resource monitor, RSCMSM CAC interface, is unavailable due to resource failure.	181
89	Call entry no mem	While creating a call entry, a resource failure occurred during call origination.	181
90	Redirect info no mem	Memory allocation failure for a redirect info structure creation during a SIP redirect (3xx) message process.	181
91	Setup, QoS mismatch	During an outgoing call, a mismatch occurred in QoS or invalid reliable provisional response and QoS configuration.	184
92	1xx, QoS mismatch	After 1xx Session progress receipt, QoS failure in negotiation occurred while the system checked the configured req and acc QoS values against values in incoming message.	184
93	RSVP failure outgoing	Resource allocation failure occurred at RSVP layer for outgoing call.	184
94	QoS retries crossed	Retries were exhausted for sending QoS PROGRESS or resource reservation requests.	184
95	RSVP failure incoming	Resource allocation failure occurred at RSVP layer for incoming call.	184
96	INVITE, QoS mismatch	During handling of INVITE, QoS failure in negotiation occurred while checking the configured req and acc QoS values against values in incoming message.	184
97	PRACK, QoS mismatch	During handling of PRACK, QoS failure in negotiation occurred while the system checked the configured req and acc QoS values against values in incoming message.	184
98	COMET, QoS mismatch	During handling of COMET, QoS failure in negotiation occurred while the system checked the local QoS values with those in received a=QoS:line in COMET.	184
99	Fback chk fail	Fallback check failure for IP network quality occurred at either the originating or terminating gateway.	184
100	ACK send sock err	The success response for INVITE send has been received; the socket returned an error for sending ACK.	186
101	Connection to contact fail	Sent INVITE and received 200 OK; TCP/UDP connection to the endpoint specified in the contact or record route failed.	186
102	Socket conn refused	A connection refused error occurred during a send operation with error 146.	186

Table 20 Error Codes for Subsystem 7 (SIP) (continued)

Code	Descriptor	Explanation	Category
103	INVITE, Preauth fail	Received INVITE; preauthentication attempt failed.	228
104	180, codec mismatch	Media negotiation failure occurred for incoming 180 Alerting responses.	278
105	183, codec mismatch	Media negotiation failure occurred for incoming 183 Session Progress responses.	278
106	200, codec mismatch	Media negotiation failure occurred for incoming 200 OK responses.	278
107	RE-INVITE, codec mismatch	A media information mismatch occurred for the media information received in re-INVITE with the media information previously received in INVITE.	278
108	ACK, codec mismatch	During processing of the ACK message in response to 200 OK, media negotiation failed due to a codec mismatch in delayed media processing.	278
109	2xx, codec mismatch	Sent mid-INVITE; received 2xx response and the media negotiation failed due to a codec mismatch.	278
110	INVITE, codec mismatch	Media negotiation failure occurred for an incoming INVITE request.	278
111	PRACK, codec mismatch	Media negotiation failure occurred for an incoming PRACK message.	278

Table 21 Error Codes for Subsystem 9 (VTSP)

Code	Descriptor	Explanation	Category
1	No Memory	Dynamically allocated memory on the gateway was exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory is exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call rejected because the default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call rejected because the default or configured memory usage threshold has been exceeded.	181
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	There were insufficient DSP resources to handle the call.	182

Table 21 Error Codes for Subsystem 9 (VTSP) (continued)

Code	Descriptor	Explanation	Category
7	Socket error	An error occurred on a socket interface.	179
8	RTP inactivity error	Media (RTP/RTCP) inactivity timer expired for the call.	185
9	Invalid arguments	Invalid arguments passed to a function. This condition usually indicates an internal software error.	180
10	Invalid state	An unexpected event was received while in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	DSP mode change	An attempt to change the DSP mode failed.	182
22	DSP unknown failure	An unspecified failure occurred in DSP interaction.	182
23	No DSP memory available	DSP could not allocate chunk memory, indicating a temporary overload or a memory leak caused by faulty software.	182
24	No DSP resource available	The call was disconnected because no DSP resources were available.	182
25	Bad DSP parameters	The call was disconnected because of some code error path.	182
26	Codec incompatible	The call failed because of incompatible codec types.	182
27	DSP alarm	DSP sent an alarm. Possible causes are: • Receiving a bad packet • Receiving a wrong message • A software problem	182
28	No voice cut through	The call failed because the voice path could not be cut through.	182

Table 21 Error Codes for Subsystem 9 (VTSP) (continued)

Code	Descriptor	Explanation	Category
29	Tie line misconfiguration	A tie-line call failed because of a misconfiguration of the tie line on the voice port. Check the tie-line string.	180
30	Invalid call mode	An unknown call mode was specified to set up a call. This condition usually indicates an internal software error.	180
31	Interface deleted	Failure to set up a call occurred on a deleted interface. This condition may happen if a call comes on an interface while the interface is being hot-swapped.	182
32	TDM hairpinning failed	TDM hairpinning failed. This condition may occur because of data structure allocation failure or because of actual hairpinning failure.	182
33	Set digit mode failed	Attempt to set the DSP to the specific digit mode failed. This condition also occurs when memory is exhausted.	182
34	Setup indication failed	This condition occurs when memory is exhausted.	180
35	DSP timeout	Call failed because of a time out on waiting for DSP action.	182

Table 22 Error Codes for Subsystem 10 (AFSAPP)

Code	Descriptor	Explanation	Category
1	No memory	Dynamically allocated memory on the gateway was exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
2	No buffers	Packet or buffer memory was exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	181
3	CPU high	Call rejected because default or configured CPU usage threshold has been exceeded.	181
4	Low memory	Call rejected because default or configured memory usage threshold has been exceeded.	181
5	No dial peer match	No dial peer satisfied the match criteria for accepting or handling the call. This condition usually indicates a dial peer misconfiguration.	128
6	No DSP resource	There were insufficient DSP resources to handle the call.	182
7	Socket error	An error occurred on a socket interface.	179
8	RTP Inactivity Error	Media (RTP/RTCP) inactivity timer expired for the call.	185

Table 22 Error Codes for Subsystem 10 (AFSAPP)

Code	Descriptor	Explanation	Category
9	Invalid arguments	Invalid arguments were passed to a function. This condition usually indicates an internal software error.	180
10	Invalid State	Some unexpected event was received while the system was in a state that was inappropriate for processing such an event.	180
11	Timeout	The software timed out waiting for some response or event to happen.	179
12	Inter-process communication	An internal process communication error occurred. This condition usually indicates some software error, but may also mean that some process was not running because of misconfiguration.	178
13	Software Error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	180
14	Gateway or Interface OOS	The gateway or signaling interfaces are being taken out of service (forcefully or gracefully). A possible cause may be the signaling interface required to support the call has already been administratively shut down.	187
21	OSP Fail	OSP settlement checking failed for an outbound call.	228
22	Call blocked by CLI	The call was rejected because it matched the profile defined for calls to be blocked. Diagnostic codes are the following: • 1—unassigned number • 17—user-busy • 21—call reject • 28—invalid number	228
23	Media request failed	Indicates a failure in media play or some other media operation.	181
24	Digit collect failed	The error occurred due to the session application failure to collect digits. The failure may be normal; that is, the caller did not enter any digits, or it may be due to an actual error in software or hardware. To interpret the diagnostic code, see Tcl IVR cd_xxx status codes in the "Events and Status Codes" chapter in the *Tcl IVR API Version 2.0 Programming Guide*.	179

Table 22 Error Codes for Subsystem 10 (AFSAPP)

Code	Descriptor	Explanation	Category
25	Call setup failed	Call setup was not successful. To interpret the diagnostic code, see Tcl IVR ls_xxx status codes in the "Events and Status Codes" chapter in the *Tcl IVR API Version 2.0 Programming Guide*.	179
26	Credit time has expired	OSP settlement allocated a limited time for call usage. The total time of the call has exceeded that usage.	228

Table 23 shows the standard internal error codes (numbered 1 through 14) and the new gatekeeper-specific error codes (numbered 21 through 45) added in Release 12.4(4)T.

Table 23 Error Codes for the Gatekeeper Subsystem

Code	Descriptor	Explanation	Category
1	No memory	Dynamically allocated memory on the gateway was exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	19
2	No buffers	Packet or buffer memory was exhausted. This condition may indicate a temporary overload or a memory leak caused by faulty software.	19
3	Timeout	Call rejected because the default or configured CPU usage threshold has been exceeded.	20
4	Software error	An internal software error occurred. Report the entire IEC string, including the diagnostic code field, to customer support.	21
5	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
6	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
7	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
8	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
9	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
10	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
11	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
12	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0

Table 23 Error Codes for the Gatekeeper Subsystem (continued)

Code	Descriptor	Explanation	Category
13	Code not assigned, reserved for future use	Error code not assigned—reserved for future use.	0
14	Call threshold exceeded	ARQ came, but because the call threshold is exceeded, this ARQ cannot be processed.	8
21	GK server error	Error in processing the GKTMP message. Server is trying to modify a field that should not be modified.	20
22	GW out of resource	Gateway is out of resources, so no more calls can be routed.	14
23	Could not find an available GW for routing	Address resoultion was not successful. Could not find the GW to route.	8
24	LRQ fail	LRQ/LRQs were sent to the remote GK, but the GK could not resolve.	20
25	Invalid endpoint ID	Mandatory endpoint identifier field in the incoming ARQ is invalid or not present.	7
26	Bad message from server	Badly formatted server message. GK/GKAPI could not process.	20
27	Proxy selection failed	Proxy selection failed.	8
28	No session bandwidth	There is no session bandwith to process the incoming ARQ.	3
29	No total bandwidth	There is no total bandwith to process incoming ARQ.	3
30	Invalid CAT token present	Incoming ARQ did not have a valid CAT to authenticate.	9
31	Endpoint killed	GK had sent a request AAA/Route server/OSP server. When the response came, the endpoint was deleted.	8
32	DRQ in progress	GK had sent a request AAA/Route server/OSP server. When the response came, DRQ is in progress.	8
33	No server responded	GK sent a request to the route server, but no server responded.	20
34	Dest proxy not found	Proxy session failed, so admission is denied.	8
35	Incomplete address	No destination Info alias and no destination IP address, and no pointers to remote zones or carriers.	11
36	Bandwidth not available	Remote or interzone bandwidth is not available.	3
37	Unable to send ACF	ACF was prepared, but could not be sent. ASN or socket error.	22
38	Duplicate CRV	Answer ARQ came with a CRV that is already being processed at the GK.	21

Table 23 Error Codes for the Gatekeeper Subsystem (continued)

Code	Descriptor	Explanation	Category
39	IZCT acc list denied	IZCT access list denied.	9
40	No bandwidth	Bandwidth not available at the terminating GK.	3
41	No bandwidth during update request	When a call is using a proxy or the call is intrazone, the call failed trying to update the bandwidth information in the call record.	3
42	Forced disengage	**call delete** CLI was used to forcefully delete the call.	24
43	GK shutdown	Call was deleted because the GK was shut down.	24
44	Aged call	Call was deleted because of aging.	24
45	Acc list denied	Access list denied.	9

How to Configure IEC Options

This section contains the following procedures:

- Configuring IEC Options, page 38 (optional)
- Verifying IEC Options, page 41 (optional)

Configuring IEC Options

No configuration is required to enable the Cisco VoIP Internal Error Codes feature. Select the following optional configuration tasks:

- Enabling IEC Syslog Reporting, page 38
- Configuring Cause Code Mapping, page 39
- Troubleshooting Tips, page 40

Enabling IEC Syslog Reporting

This task enables IEC syslog reporting.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice iec syslog**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	`voice iec syslog`
	
	Example:
	`Router(config)# voice iec syslog`
		(Optional) Enables syslog messages as IECs occur.

Configuring Cause Code Mapping

This task enables cause code mapping.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice cause-code**
4. error-category *number* q850-cause *number*
5. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** `Router# configure terminal`	Enters global configuration mode.

Step 3	`voice cause-code`
	
	Example:
	`Router(config)# voice cause-code`
		(Optional) Enters voice cause-code configuration mode.

	Command or Action	Purpose
Step 4	`error-category` *number* `q850-cause` *number* **Example:** `Router(conf-voice-cause)# error-category 128 q850-cause 27`	(Optional) Specifies the values to be mapped. • Values for error-category range from 128 to 278. • Values for the Q.850 cause code range from 1 to 127.

Step 5	`exit`
	
	Example:
	`Router(conf-voice-cause)# exit`
		Exits voice cause-code configuration mode.

Troubleshooting Tips

The IEC feature is itself a troubleshooting tool. By enabling the **voice iec syslog command you can display IECs logged in real time, which allows you to isolate a failure cause without turning on debugging. Then, based on the IEC reported, you can selectively enable the appropriate debug tool to gather additional information.**

The IEC feature also provides a Cisco.com diagnostic tool that allows you to enter an IEC dotted string and receive an explanation of the component fields. The explanation includes a description of the problem, and depending upon the error type, a recommended course of action. Use the IEC lookup tool at http://www.cisco.com/univercd/cc/td/doc/product/voice/vtgemd.htm.

To troubleshoot specific subsystems that do not generate corresponding IECs, use the following **debug** and **show** commands:

- To learn whether the ISDN link is up or down, use the **show isdn status** command.
- To display information about whether the ISDN link is receiving SETUP, CALLPRO, ALERT, CONNECT, and RELEASE COMPLETE messages, use the **debug isdn q931** command.
- To display information about H.225 and RAS messages exchanged between a gateway and gatekeeper, use the **debug h225 asn1** command. H.225 debug output for the terminating side, in the initial stage when a setup message is being received, provides an indication if messages are being received from the IP side and if H.323 service is operational. If the H.225 connection is not established from the incoming side, then no IECs are generated.

What to Do Next

Proceed to the section "Verifying IEC Options, page 41."

New and Modified Configuration Commands for Gatekeeper IECs in Cisco IOS Release 12.4(4)T

To enable the enhanced capabilities of the gatekeeper-specific IECs in Release 12.4(4)T, there is one new command and one modified command. This section describes only the new information. For complete information on the commands for voice gateways and gatekeepers, refer to the *Cisco IOS Voice Configuration Library* on Cisco.com.

This release introduces a new command for the gatekeeper configuration that causes retention of call history and enables you to specify the number of records to be kept in the history table.

In gatekeeper configuration mode, enter:

gatekeeper(config)# **call-history max-size** *number*

The *number* argument in this syntax can be any number from 0 to 1200. The default is 15. This represents the maximum number of records of old calls to be stored and available for display.

To display the historical information, enter the following command on the gatekeeper:

gatekeeper# **show gatekeeper calls history**

This command has been modified with the addition of the **history** keyword. This keyword was added to display call history information along with internal error codes at the gatekeeper. The number of disconnected calls displayed in response to this command is the *number* value specified in the **call-history max-size** *number* command. Use of this **max-size** number helps to reduce excessive CPU usage in the storage and reporting of this information.

Verifying IEC Options

You can use **show** command output to display IEC option configuration, to verify that the feature is working, and to display IEC counter information.

Prerequisites

Before you can display IEC counter information, you must configure voice statistics settings.

SUMMARY STEPS

1. **enable**
2. voice statistics type iec
3. **voice statistics max-storage-duration** {**day** *number-of-days* | **hour** *number-of-hours* | **minute** *number-of-minutes*}
4. **voice statistics time-range periodic** *interval-length* [**start** *hh*:*mm*] [**end** *hh*:*mm*] [**days-of-week** *days*]
5. voice statistics time-range since-reset

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`voice statistics type iec`
	
	Example:
	`Router# voice statistics type iec`
		(Optional) Configures the collection of IEC statistics.

	Command or Action	Purpose

Step 3	`voice statistics max-storage-duration {day` *number-of-days*`| hour` *number-of-hours* `| minute` *number-of-minutes*`}`
	
	Example:
	`Router# voice statistics max-storage-duration day 1`
		(Optional) Configures how long interval counters are kept for display.
		• If you want to display counters for past intervals, you must configure a storage duration for expired counters. Otherwise, once the interval has expired, the counters are no longer available.

Step 4	**voice statistics time-range periodic** *interval-length* [**start** *hh*:*mm*] [**end** *hh*:*mm*] [**days-of-week** *days*] **Example:** `Router# voice statistics time-range periodic 30minutes`	(Optional) Specifies IEC collection intervals. • The *interval-length* argument takes one of the following values: – 5minutes – 15minutes – 30minutes – 60minutes – 1day • The range for ***hh*:*mm*** is 00:00 to 23:59. The default for the **start** keyword is 00:00. The default for the **end** keyword is 00:00. • The *days* argument takes one of the following values: – friday—Friday – monday—Monday – saturday—Saturday – sunday—Sunday – thursday—Thursday – tuesday—Tuesday – wednesday—Wednesday – daily—Every day of the week – weekdays—Monday thru Friday – weekend—Saturday and Sunday The default is **daily**.

Step 5	`voice statistics time-range since-reset`
	
	Example:
	`Router# voice statistics time-range since-reset`
		(Optional) Enables the collection of call statistics information accumulated since the last resetting of IEC counters.

Displaying IEC Options

Perform this task to verify that the Cisco VoIP Internal Error Codes feature is working.

SUMMARY STEPS

1. **enable**

2. **show running-config**
3. show voice cause-code category-q850
4. **show voice iec description *string***
5. show voice statistics iec {interval *number* | since-reboot | since-reset}
6. clear voice statistics iec

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`show running-config`
	
	Example:
	`Router# show running-config`
		(Optional) Displays the configuration information currently running on the router.

Step 3	`show voice cause-code category-q850`
	
	Example:
	`Router# show voice cause-code category-q850`
		(Optional) Displays Q.850 cause code mapping.

Step 4	`show voice iec description string`
	
	Example:
	`Router# show voice iec description 1.1.128.1.5.0`
		(Optional) Displays an IEC description.

Step 5	`show voice statistics iec {interval number| since-reboot|since-reset}`
	
	Example:
	`Router# show voice statistics iec interval 15`
		(Optional) Displays IEC statistics.
		• Specify the following displays: statistics by selected time interval, or statistics since the last router reboot, or statistics since the last instance when counters were cleared.

Step 6	`clear voice statistics iec`
	
	Example:
	`Router# clear voice statistics iec`
		(Optional) Resets IEC counters.

Configuration Examples for Cisco VoIP Internal Error Codes

This section provides configuration examples for the tasks identified in the previous section.

- Enabling IEC Syslog Reporting and Configuring Cause Code Mapping: Example, page 44
- Verifying IEC Configuration: Example, page 44

Enabling IEC Syslog Reporting and Configuring Cause Code Mapping: Example

In the following example, IEC syslog reporting and cause-code mapping are enabled:

```
enable
configure terminal
    voice iec syslog
    voice cause-code
        error-category 128 q850-cause 27
```

Verifying IEC Configuration: Example

In the following examples, the output is displayed for each command used to verify IEC configuration:

Sample Output from the show running-config Command: Example

```
Router# show running-config

Building configuration...

Current configuration :2791 bytes
!
version 12.2
no parser cache
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
!
hostname GW-1
!
no boot startup-test
!
!
resource-pool disable
spe default-firmware spe-firmware-1
!
!
aaa new-model
!
!
aaa group server radius h323
!
aaa authentication login h323 group radius group h323
aaa authorization config-commands
aaa authorization exec h323 group h323
aaa accounting connection h323 start-stop group radius group h323
aaa session-id common
!
isdn switch-type primary-5ess
!
```

The following lines show optional IEC configuration information:

```
voice iec syslog
voice cause-code
 error-category 128 q850-cause 27
 error-category 129 q850-cause 38
!
```

```
!
!
controller T1 3/0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 3/1
 shutdown
 framing sf
 linecode ami
gw-accounting aaa
!
!
!
interface FastEthernet0/0
 ip address 172.18.195.28 255.255.255.0
 no ip route-cache
 no ip mroute-cache
 duplex full
 speed 100
 no cdp enable
 h323-gateway voip interface
 h323-gateway voip id GK-1 ipaddr 172.18.195.41 1718
 h323-gateway voip h323-id GW-1
!
interface FastEthernet0/1
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
interface Serial0/0
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 clockrate 2000000
 no cdp enable
!
interface Serial0/1
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 clockrate 2000000
 no cdp enable
!
interface Serial3/0:23
 no ip address
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 172.18.195.1
no ip http server
!
!
!
```

```
radius-server host 172.18.200.222 auth-port 1645 acct-port 1646
radius-server key lab
radius-server authorization permit missing Service-Type
radius-server vsa send accounting
call rsvp-sync
!
!
voice-port 3/0:D
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 100 pots
 destination-pattern 1#919....
 direct-inward-dial
 port 3/0:D
 prefix 919
!
dial-peer voice 301 voip
 destination-pattern 7190003
 session target ras
!
!
gateway
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
 logging synchronous
line vty 0 4
 password lab
line 1/00 1/59
 no flush-at-activation
 modem InOut
!
scheduler allocate 10000 400
end
```

Sample Output from the show voice iec description Command: Example

```
Router# show voice iec description 1.1.128.1.5.0

    IEC Version:1
    Entity:1 (Gateway)
    Category:128 (Destination address resolution failure)
    Subsystem:1 (CCAPI)
    Error:5 (No dial peer match)
    Diagnostic Code:0
```

Sample Output from the show voice statistics iec Command: Example

```
Router# show voice statistics iec since-reset

Internal Error Code counters
```

```
---------------------------
Counters since last reset (2002-11-28T01:55:31Z):
  SUBSYSTEM CCAPI [subsystem code 1]
      [errcode   6] No DSP resource                              5

  SUBSYSTEM SSAPP [subsystem code 4]
      [errcode   5] No dial peer match                           2
      [errcode   3] CPU high                                    96

  SUBSYSTEM H323 [subsystem code 5]
      [errcode  22] No Usr Responding, H225 timeout              1
      [errcode 27] H225 invalid msg                              1
      [errcode 79] H225 chn, sock fail                          27

  SUBSYSTEM VTSP [subsystem code 9]
      [errcode   6] No DSP resource                             83


Router# show voice statistics iec since-reboot
Internal Error Code counters
---------------------------
Counters since reboot:
  SUBSYSTEM CCAPI [subsystem code 1]
      [errcode   6] No DSP resource                             93

  SUBSYSTEM SSAPP [subsystem code 4]
      [errcode   5] No dial peer match                         830
      [errcode   3] CPU high                                  1423

  SUBSYSTEM H323 [subsystem code 5]
      [errcode  21] No Usr Responding, H225 timeout             21
      [errcode  23] H225 invalid msg                            17
      [errcode  39] H225 chn, sock fail                       2073

  SUBSYSTEM VTSP [subsystem code 9]
      [errcode   6] No DSP resource                            429
```

Before using the **show voice statistics iec interval** command, first determine the intervals available for display by using the **show voice statistics interval-tag** command:

```
Router# show voice statistics interval-tag
Current Time:2002-11-28T06:04:21Z
INTERVAL-TAG   START TIME             END TIME
============== ===================== =====================
1              2002-11-28T02:00:00Z   2002-11-28T02:30:01Z
2              2002-11-28T02:30:01Z   2002-11-28T03:00:01Z
3              2002-11-28T03:00:01Z   2002-11-28T03:30:01Z
4              2002-11-28T03:30:01Z   2002-11-28T04:00:01Z
5              2002-11-28T04:00:01Z   2002-11-28T04:30:01Z
6              2002-11-28T04:30:01Z   2002-11-28T05:00:01Z
7              2002-11-28T05:00:01Z   2002-11-28T05:30:01Z
8              2002-11-28T05:30:01Z   2002-11-28T06:00:01Z
9              2002-11-28T06:00:01Z   2002-11-28T06:04:21Z
```

The following example shows **interval-tag 8** statistics:

```
Router# show voice statistics iec interval 8

Internal Error Code counters
---------------------------
Counters for interval 8, beginning 2002-11-28T05:30:01Z,ending 2002-11-28T06:00:01Z:
  SUBSYSTEM CCAPI [subsystem code 1]
```

```
      [errcode   6] No DSP resource                          1
  SUBSYSTEM SSAPP [subsystem code 4]
      [errcode   3] CPU high                                15
  SUBSYSTEM H323 [subsystem code 5]
      [errcode  23] H225 invalid msg                         1
      [errcode  39] H225 chn, sock fail                      1
  SUBSYSTEM VTSP [subsystem code 9]
      [errcode   6] No DSP resource                          6
```

Sample Output from the clear voice statistics Command: Example

The following examples show **voice statistics iec since-reset** output before and after you issue the **clear voice statistics** command:

```
Router# show voice statistics iec since-reset

Internal Error Code counters
----------------------------
Counters since last reset (2002-11-28T01:55:31Z):
  SUBSYSTEM CCAPI [subsystem code 1]
      [errcode   6] No DSP resource                          5

  SUBSYSTEM SSAPP [subsystem code 4]
      [errcode   5] No dial peer match                        2
      [errcode   3] CPU high                                96

  SUBSYSTEM H323 [subsystem code 5]
      [errcode  21] No Usr Responding, H225 timeout          1
      [errcode  23] H225 invalid msg                         1
      [errcode  39] H225 chn, sock fail                     27

  SUBSYSTEM VTSP [subsystem code 9]
      [errcode   6] No DSP resource                         83

Router# clear voice statistics iec
Router# show voice statistics iec since-reset
        Internal Error Code counters
        ----------------------------
        Counters since last reset (2002-12-12T22:33:25Z):
        No errors.
```

Sample Output from the show voice cause-code category-q850 Command: Example

```
Router# show voice cause-code category-q850

The Internal Error Category to Q850 cause code mapping table:-

 Error Configured Default    Description
Category  Q850     Q850
   128     27        3  Destination address resolution failure
   129     38      102  Call setup timeout
   178     41       41  Internal Communication Error
   179     41       41  External communication Error
   180     47       47  Software Error
   181     47       47  Software Resources Unavailable
   182     47       47  Hardware Resources Unavailable
   183     41       41  Capability Exchange Failure
   184     49       49  QoS Error
   185     41       41  RTP/RTCP receive timer expired or bearer layer failure
   186     38       38  Signaling socket failure
   187     38       38  Gateway or signaling interface taken out of service
```

```
228     50       50  User is denied access to this service
278     65       65  Media Negotiation Failure due to non-existing Codec
```

Troubleshooting VoIP Networks Using Cisco VoIP Internal Error Codes

IECs are generated for errors that cause the gateway to release or refuse a call. This section provides procedural and reference information used to troubleshoot gateway-detected errors and resolve problems on the gateway and with other VoIP network entities.

Because fields within the IEC identify which network entity and subsystem originated an error, they can be used to diagnose and isolate failures that can cause call disconnects.

This section discusses troubleshooting scenarios using IECs to diagnose and resolve the problems described in the following sections:

- Troubleshooting Two-Stage Dialing Failures, page 49
- Troubleshooting Socket Failures, page 53

Troubleshooting Two-Stage Dialing Failures

The following example shows how to troubleshoot a two-stage dialing failure.

Symptom

The Cisco router or gateway rejects a call placed by a PSTN ISDN user after all the digits have been dialed.

Problem Description

The PSTN user enters a destination number that is routed through the PSTN ISDN switch, which sends an ISDN SETUP message to the router. The router tags the incoming call leg and sends back an ISDN CONNECT message. The caller receives second dial tone. The router then enters the digit collection stage to use the collected digits to route the call to the next hop, at which point the router rejects the call.

Troubleshooting Tasks

Perform the following steps to determine the reason for call failure.

Step 1 Use the **voice iec syslog** command to enable displaying of IECs as they are encountered in real-time.

Step 2 Use the **voice iec statistics type iec** command to configure the collection of IEC statistics.

Step 3 Use the **show running-config** command to verify IEC, ISDN, and dial-peer configuration, as shown in the following partial sample output:

```
Router> show running-config

Building configuration...
!
```

```
voice rtp send-recv
!
voice service voip
```

The following lines show the IEC configuration:

```
voice iec syslog
no voice hpi capture buffer
no voice hpi capture destination
voice statistics type iec
!
!
!
```

The following lines show the T1 configuration:

```
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
```

The following lines show the ISDN configuration:

```
interface Serial0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice modem
 no cdp enable
!
```

The following lines show the dial-peer configuration. Because the dial-peer voice 1 is not configured for direct inward dialing (DID), the inbound call leg is considered to be configured for two-stage dialing, and the router returns a second dial tone.

```
dial-peer voice 1 pots
 incoming called-number .
 port 0:D
!
dial-peer voice 2 voip
 destination-pattern 83101
 session target ipv4:172.69.85.107
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 ip qos dscp cs5 media
!
!end
```

Step 4 **Use the show controller t1** command to display T1 status. Verify that the T1 is UP and that there are no errors.

```
Router> show controller t1 0

T1 0 is up.
  Applique type is Channelized T1
  Cablelength is short 133
  No alarms detected.
  alarm-trigger is not set
  Version info of slot 0:  HW: 1, PLD Rev: 11
  Framer Version: 0x8
!
!
```

```
!
Framing is ESF, Line Code is B8ZS, Clock Source is Line Primary.
  Data in current interval (0 seconds elapsed):
     0 Line Code Violations, 0 Path Code Violations
     0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
     0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
```

Step 5 Use the **show isdn status** command to display ISDN status. Verify the ISDN Layer 2 status is MULTIPLE_FRAME_ESTABLISHED.

```
Router> show isdn status

Global ISDN Switchtype = primary-ni
ISDN Serial0:23 interface
        dsl 0, interface ISDN Switchtype = primary-ni
    Layer 1 Status:
        ACTIVE
    Layer 2 Status:
        TEI = 0, Ces = 1, SAPI = 0, State = MULTIPLE_FRAME_ESTABLISHED
    Layer 3 Status:
        0 Active Layer 3 Call(s)
    Active dsl 0 CCBs = 0
    The Free Channel Mask:  0x807FFFFF
    Number of L2 Discards = 0, L2 Session ID = 3
    Total Allocated ISDN CCBs = 0
```

Step 6 Use the s**how isdn service** command to display the status of each ISDN channel. Verify that the channels are IDLE and IN-SERVICE.

```
Router> show isdn service
PRI Channel Statistics:
ISDN Se0:23, Channel [1-24]
  Configured Isdn Interface (dsl) 0
   Channel State (0=Idle 1=Proposed 2=Busy 3=Reserved 4=Restart 5=Maint_Pend)
    Channel :  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
    State   :  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3
   Service State (0=Inservice 1=Maint 2=Outofservice)
    Channel :  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
    State   :  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
```

Step 7 Use the **show dial-peer voice summary** command to display voice dial peer information. Verify that Admin and Operation status are up and up.

```
Router> show dial-peer voice summary
             AD                                      PRE PASS
TAG    TYPE  MIN  OPER PREFIX    DEST-PATTERN      FER THRU SESS-TARGET   PORT
1      pots  up   up                               0                      0:D
2      voip  up   up              83101            0  syst ipv4:172.69.85.107
```

Step 8 Use the **debug isdn q931** command to display information about call setup and teardown of ISDN network connections. Use the **debug vtsp dsp**, **debug vtsp session**, and **debug voip ccapi inout** commands to get digit collection information, as shown in the following partial output.

```
Router# debug isdn q931
Router# debug vtsp dsp
Router# debug vtsp session
Router# debug voip ccapi inout

Aug 18 23:56:20.125: ISDN Se0:23 Q931: RX <- SETUP pd = 8  callref = 0x0226
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
```

```
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Calling Party Number i = 0x2181, '40855501124'
                Plan:ISDN, Type:National
        Called Party Number i = 0x80, '83101'
                Plan:Unknown, Type:Unknown
Aug 18 23:56:20.133: VDEV_ALLOCATE: 1/1 is allocated
!
!
!
```

The following lines show the digit collection process, starting with the digit 8:

```
Aug 18 23:56:28.265: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_dsm_digit_begin_cb: Digit
begin: 8
Aug 18 23:56:28.265: //6/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPtr=0x0,
dstCallId=0xFFFFFFFF, srcCallId=0x6,
!
!
!
```

The router collects the next digit, 3, followed by 1, 0 and 2:

```
Aug 18 23:56:30.253: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_dsm_digit_begin_cb: Digit
begin: 3
Aug 18 23:56:30.253: //6/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPtr=0x0,
dstCallId=0xFFFFFFFF, srcCallId=0x6,
    digit=3, digit_begin_flags=0x1, rtp_timestamp=0xFFFFFE70
    rtp_expiration=0x0, dest_mask=0x1)
!
!
!
Aug 18 23:56:30.885: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_dsm_digit_begin_cb: Digit
begin: 1
Aug 18 23:56:30.885: //6/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPtr=0x0,
dstCallId=0xFFFFFFFF, srcCallId=0x6,
    digit=1, digit_begin_flags=0x1, rtp_timestamp=0xFFFFFE70
    rtp_expiration=0x0, dest_mask=0x1)
!
!
!
Aug 18 23:56:31.913: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_dsm_digit_begin_cb: Digit
begin: 0
Aug 18 23:56:31.913: //6/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPtr=0x0,
dstCallId=0xFFFFFFFF, srcCallId=0x6,
    digit=0, digit_begin_flags=0x1, rtp_timestamp=0xFFFFFE70
    rtp_expiration=0x0, dest_mask=0x1)
!
!
!
Aug 18 23:56:33.185: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_dsm_digit_begin_cb: Digit
begin: 2
Aug 18 23:56:33.185: //6/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPtr=0x0,
dstCallId=0xFFFFFFFF, srcCallId=0x6,
    digit=2, digit_begin_flags=0x1, rtp_timestamp=0xFFFFFE70
    rtp_expiration=0x0, dest_mask=0x1)
!
!
!
Aug 18 23:56:33.265: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_report_digit_control:
digit reporting disabled
Aug 18 23:56:33.265:
//6/65F920768011/DSM:(0:D):0:112:4386/dsp_stream_mgr_register_disposition: Ev:
E_DSM_DSP_DTMF_DIGIT_BEGIN Disp: DS
```

```
M_DISP_IGNORE
Aug 18 23:56:33.265:
//6/65F920768011/DSM:(0:D):0:112:4386/dsp_stream_mgr_register_disposition: Ev:
E_DSM_DSP_DTMF_DIGIT Disp: DSM_DISP_IGNORE
Aug 18 23:56:33.269: //6/xxxxxxxxxxxx/CCAPI/cc_api_call_report_digits_done:
(vdbPtr=0x639DB450, callID=0x6, disp=0)
Aug 18 23:56:33.269: //6/65F920768011/VTSP:(0:D):0:112:4386/vtsp_get_digit_timeouts: Inter
digit = 10, Initial digit = 10
```

Because the router does not have an outgoing dial peer with destination pattern 83102, the call fails and an IEC is generated.

```
Aug 18 23:56:33.269: %VOICE_IEC-3-GW: AFSAPP: Internal Error (Digit collect failed):
IEC=1.1.179.10.24.6 on callID 6 GUID=65F92076D10E11D7801100B0640E6622
```

Step 9 Use the **show voice iec description** command to display the IEC definition:

```
Router> show voice iec description 1.1.179.10.24.6

    IEC Version: 1
    Entity:
    Category: 179 (External communication Error)
    Subsystem: 10 (AFSAPP)
    Error: 24 (Digit collect failed)
    Diagnostic Code: 6
```

IEC field definitions pinpoint the problem. Category code 179 indicates an external communication error, and an error code 24 indicates digit collection failure. For more information on IEC field definitions, see the "Internal Error Code Notation" section on page 7.

Troubleshooting Socket Failures

The following example, which describes a TCP session failure, shows how errors detected by the gateway can be used to troubleshoot other devices on the VoIP network.

Symptom

An inbound call from an IP phone to the H.323 gateway fails.

Problem Description

A call is initially routed to the gateway and fails when a TCP session to Cisco CallManager session target cannot be established. The router pings Cisco CallManager, sending a TCP synchronization packet and receiving an ICMP destination unreachable error. Cisco CallManager cannot be pinged because the Cisco CallManager IP address is incorrect. After the IP address for Cisco CallManager is corrected, a second call fails, due to a different socket error. The router tries to establish another TCP session and sends an H225 setup message but Cisco CallManager drops the connection.

Troubleshooting Tasks

Perform the following steps to determine the reasons for both call failures.

Step 1 Use the **voice iec syslog** command to enable display of IECs as they are encountered in real-time.

Step 2 Use the **voice iec statistics type iec** command to configure the collection of IEC statistics.

Step 3 Use the s**how running-config** command to verify IEC, ISDN, and dial-peer configuration, as shown in the following partial sample output:

```
Router> show running-config
Building configuration...

Current configuration : 3466 bytes
!
```

The following lines show the IEC configuration:

```
voice service voip
!
voice iec syslog
no voice hpi capture buffer
no voice hpi capture destination
voice statistics type iec
!
!
```

The following lines show the T1 configuration:

```
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
```

The following lines show the ISDN configuration:

```
interface Serial0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice modem
 no cdp enable
!
ivoice-port 0:D
!
```

The following lines show the dial-peer configuration, including the destination gateway IP address of Cisco CallManager:

```
dial-peer voice 1 pots
 incoming called-number
 direct-inward-dialed
 port 0:D
!
dial-peer voice 2 voip
 destination-pattern 83101
 session target ipv4:10.1.1.1
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 ip qos dscp cs5 media
!
!
end
```

Step 4 Use the **debug isdn q931** command to display information about call setup and teardown of ISDN network connections.

```
Router# debug isdn q931
Aug 19 01:46:02.886: ISDN Se0:23 Q931: RX <- SETUP pd = 8  callref = 0x022D
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Calling Party Number i = 0x2181, '4085550111'
                Plan:ISDN, Type:National
        Called Party Number i = 0x80, '83101'
                Plan:Unknown, Type:Unknown
```

The following lines show the IEC and specify a network problem.

```
Aug 19 01:46:03.342: %VOICE_IEC-3-GW: H323: Internal Error (SETUP send sock fail):
IEC=1.1.186.5.83.0 on callID 14 GUID=B99ACE6ED11D11D7801500B0640E6622
Aug 19 01:46:03.350: ISDN Se0:23 Q931: TX -> CALL_PROC pd = 8  callref = 0x822D
        Channel ID i = 0xA98381
                Exclusive, Channel 1
Aug 19 01:46:03.362: ISDN Se0:23 Q931: TX -> DISCONNECT pd = 8  callref = 0x822D
        Cause i = 0x80A6 - Network out of order
Aug 19 01:46:03.374: ISDN Se0:23 Q931: RX <- RELEASE pd = 8  callref = 0x022D
        Cause i = 0x82E4 - Invalid information element contents
Aug 19 01:46:03.374: ISDN Se0:23 Q931: TX -> RELEASE_COMP pd = 8  callref = 0x822D
```

Step 5 Use the following commands to further isolate the problem:

- The **show voice iec description** command displays the IEC definition.
- The **debug ip tcp transaction** command displays output for packets the router sends and receives.
- The **debug cch323 h225** command provides the trace of the state transition of the H.225 state machine based on the processed events.

The following partial sample outputs from each command help you to isolate the cause of the network out of order message:

In the following example, the IEC definition indicates a category code of 186, a signaling socket failure, and shows that an error occurred during the SETUP PDU operation. The explanation for the error code 83 states that this error can happen if the remote IP address is a reachable address for pinging but is not a valid H.323 endpoint.

```
Router# show voice iec description 1.1.186.5.83.0
    IEC Version: 1
    Entity: 1
    Category: 186
    Subsystem: 5
    Error: 83
    Diagnostic Code: 0
```

Because the IEC specifies a signaling socket failure as the reason for call failure, you should enable the following **debug** commands to get more information.

```
Router# debug ip tcp transaction

TCP special event debugging is on

Router# debug cch323 h225

H225 State Machine tracing is enabled

Router# terminal monitor
```

```
% Console already monitors

Router#
Aug 19 01:46:28.746: ISDN Se0:23 Q931: RX <- SETUP pd = 8  callref = 0x022E
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Calling Party Number i = 0x2181, '4085551090'
                Plan:ISDN, Type:National
        Called Party Number i = 0x80, '83101'
                Plan:Unknown, Type:Unknown
Aug 19 01:46:29.198: TCB63D2DAC8 created
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_PID (8) 63A2C044
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_NO_DELAY (1) 63A2C048
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_TOS (11) 63A2C070
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_NONBLOCKING_WRITE (10) 63A2C0D0
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_NONBLOCKING_READ (14) 63A2C0D0
Aug 19 01:46:29.198: TCB63D2DAC8 setting property unknown (15) 63A2C0D0
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_NO_DELAY (1) 63A2C08C
Aug 19 01:46:29.198: TCB63D2DAC8 setting property TCP_ALWAYSPUSH (17) 63A2C08C
Aug 19 01:46:29.198: TCB63D2DAC8 bound to 172.16.13.16.11005
Aug 19 01:46:29.198: TCP: sending SYN, seq 3651477840, ack 0
Aug 19 01:46:29.198: TCP0: Connection to 10.1.1.1:1720, advertising MSS 536
Aug 19 01:46:29.198: TCP0: state was CLOSED -> SYNSENT [11005 -> 10.1.1.1(1720)]
```

The following lines show the 10.1.1.1 CallManager address is unreachable as it is configured, and the network out of order IEC is generated:

```
Aug 19 01:46:29.202: TCP0: ICMP destination unreachable received
!
!
!Aug 19 01:46:29.206: %VOICE_IEC-3-GW: H323: Internal Error (SETUP send sock fail):
IEC=1.1.186.5.83.0 on callID 16 GUID=C904C18BD11D11D7801600B0640E6622
Aug 19 01:46:29.206: TCB 0x63D2DAC8 destroyed
Aug 19 01:46:29.206: //16/C904C18B8016/H323/run_h225_sm: Received event H225_EV_CONN_LOST
while at state H225_IDLE
Aug 19 01:46:29.214: ISDN Se0:23 Q931: TX -> CALL_PROC pd = 8  callref = 0x822E
        Channel ID i = 0xA98381
                Exclusive, Channel 1
Aug 19 01:46:29.218: //16/C904C18B8016/H323/run_h225_sm: Received event H225_EV_RELEASE
while at state H225_IDLE
Aug 19 01:46:29.218: //16/C904C18B8016/H323/cch323_h225_set_new_state: Changing from
H225_IDLE state to H225_IDLE state
Aug 19 01:46:29.226: ISDN Se0:23 Q931: TX -> DISCONNECT pd = 8  callref = 0x822E
        Cause i = 0x80A6 - Network out of order
!
!
```

Step 6 Use the **show ip route** command to display static routes, then use the **ping** command to check network connectivity to the destination address 10.1.1.1 using the **ping** command.

```
Router> show ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
```

```
Gateway of last resort is 172.16.13.3 to network 0.0.0.0

     172.16.0.0/27 is subnetted, 1 subnets
C       172.16.13.0 is directly connected, Ethernet0
S*   0.0.0.0/0 [1/0] via 172.16.13.3
```

The following lines show that the reason for Cisco CallManager TCP session failure is the incorrect 10.1.1.1 IP address:

```
Router# ping 10.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
..U.U
Success rate is 0 percent (0/5)
Router# ping 10.1.1.1
```

Step 7 Configure the correct IP address for Cisco CallManager and verify the configuration using the **show running-config** command.

```
Router# config term
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# dial-peer voice 2
Router(config-dial-peer)# session target ipv4:172.31.85.107
Router(config-dial-peer)# end

Router# show running-config
!
!
!
dial-peer voice 2 voip
 destination-pattern 83101
 session target ipv4:172.31.85.107
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 ip qos dscp cs5 media
```

Step 8 Use the **show debug** command to display call traces enabled during a second call attempt.

```
Router# show debug

CSM Voice:
  Voice Call Switching Module debugging is on


The following ISDN debugs are enabled on all DSLs:

debug isdn error is             ON.
debug isdn q931 is              ON.   (filter is OFF)


Router#
Aug 19 02:05:36.349: ISDN Se0:23 Q931: RX <- SETUP pd = 8  callref = 0x0237
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Calling Party Number i = 0x2181, '4085550111'
                Plan:ISDN, Type:National
        Called Party Number i = 0x80, '83101'
                Plan:Unknown, Type:Unknown
```

```
Aug 19 02:05:36.353: VDEV_ALLOCATE: 1/2 is allocated
Aug 19 02:05:36.353: csm_vtsp_init_tdm: vdev@ 0x6355AAF4, voice_vdev@ 0x6355AA80
Aug 19 02:05:36.353: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 1, dsp 2,
dsp_channel 1
Aug 19 02:05:36.353: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 4, channel 3, bank
2, bp_channel 1, bp_stream 255
Aug 19 02:05:36.357: VDEV_DEALLOCATE: slot 1, port 4 is deallocated
Aug 19 02:05:36.805: ISDN Se0:23 Q931: TX -> CALL_PROC pd = 8  callref = 0x8237
        Channel ID i = 0xA98381
                Exclusive, Channel 1
```

The second call attempt also fails, generating a different IEC:

```
Aug 19 02:05:36.845: %VOICE_IEC-3-GW: H323: Internal Error (Socket error):
IEC=1.1.186.5.7.6 on callID 8 GUID=750AFC91D12011D7800600B0640E6622
Aug 19 02:05:36.865: ISDN Se0:23 Q931: TX -> DISCONNECT pd = 8  callref = 0x8237
        Cause i = 0x80A6 - Network out of order
Aug 19 02:05:36.873: ISDN Se0:23 Q931: RX <- RELEASE pd = 8  callref = 0x0237
        Cause i = 0x82E4 - Invalid information element contents
Aug 19 02:05:36.877: ISDN Se0:23 Q931: TX -> RELEASE_COMP pd = 8  callref = 0x8237
```

The following lines show that the subsequent call failed due to a different socket error. However, in this instance sending a ping to the remote IP address is successful:

```
Router> show voice iec description 1.1.186.5.7.6

    IEC Version: 1
    Entity: 1 (Gateway)
    Category: 186 (Signaling socket failure)
    Subsystem: 5 (H323)
    Error: 7 (Socket error)
    Diagnostic Code: 6

Router> ping 172.69.85.107

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.69.85.107, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
```

Use the **debug ip tcp transaction**, **show debug**, and **debug cch323 h225** commands again.

```
Router# debug ip tcp transaction

TCP special event debugging is on

Router# show debug

CSM Voice:
  Voice Call Switching Module debugging is on
TCP:
  TCP special event debugging is on


The following ISDN debugs are enabled on all DSLs:

debug isdn error is             ON.
debug isdn q931 is              ON.   (filter is OFF)


Router# debug cch323 h225

H225 State Machine tracing is enabled
```

```
Router#
Aug 19 02:06:36.637: ISDN Se0:23 Q931: RX <- SETUP pd = 8  callref = 0x0238
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Calling Party Number i = 0x2181, '4085550111'
                Plan:ISDN, Type:National
        Called Party Number i = 0x80, '83101'
                Plan:Unknown, Type:Unknown
Aug 19 02:06:36.641: VDEV_ALLOCATE: 1/3 is allocated
Aug 19 02:06:36.641: csm_vtsp_init_tdm: vdev@ 0x6355B240, voice_vdev@ 0x6355B1CC
Aug 19 02:06:36.641: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 1, dsp 3,
dsp_channel 1
Aug 19 02:06:36.641: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 4, channel 5, bank
2, bp_channel 2, bp_stream 255
Aug 19 02:06:36.645: VDEV_DEALLOCATE: slot 1, port 8 is deallocated
Aug 19 02:06:37.089: TCB63604218 created
Aug 19 02:06:37.089: TCB63604218 setting property TCP_PID (8) 632036B4
Aug 19 02:06:37.089: TCB63604218 setting property TCP_NO_DELAY (1) 632036B8
Aug 19 02:06:37.089: TCB63604218 setting property TCP_TOS (11) 632036E0
Aug 19 02:06:37.089: TCB63604218 setting property TCP_NONBLOCKING_WRITE (10) 63203740
Aug 19 02:06:37.089: TCB63604218 setting property TCP_NONBLOCKING_READ (14) 63203740
Aug 19 02:06:37.089: TCB63604218 setting property unknown (15) 63203740
Aug 19 02:06:37.089: TCB63604218 setting property TCP_NO_DELAY (1) 632036FC
Aug 19 02:06:37.089: TCB63604218 setting property TCP_ALWAYSPUSH (17) 632036FC
Aug 19 02:06:37.089: TCB63604218 bound to 172.16.13.16.11001
Aug 19 02:06:37.089: TCP: sending SYN, seq 1593750728, ack 0
Aug 19 02:06:37.093: TCP0: Connection to 172.31.85.107:1720, advertising MSS 536
Aug 19 02:06:37.093: TCP0: state was CLOSED -> SYNSENT [11001 -> 172.31.85.107(1720)]
Aug 19 02:06:37.093: TCP0: state was SYNSENT -> ESTAB [11001 -> 172.31.85.107(1720)]
Aug 19 02:06:37.093: TCP0: Connection to 172.31.85.107:1720, received MSS 1460, MSS is 536
Aug 19 02:06:37.093: //10/98FA43DC8007/H323/run_h225_sm: Received event H225_EV_SETUP
while at state H225_IDLE
Aug 19 02:06:37.093: //10/98FA43DC8007/H323/check_qos_and_send_setup: Setup ccb 0x635F80E8
Aug 19 02:06:37.093: //10/98FA43DC8007/H323/run_h225_sm: Received event H225_EV_FS_SETUP
while at state H225_IDLE
Aug 19 02:06:37.093: //10/98FA43DC8007/H323/idle_fsSetup_hdlr: Setup ccb 0x635F80E8
Aug 19 02:06:37.097: //10/98FA43DC8007/H323/generic_send_setup: sending calling IE
Aug 19 02:06:37.097: //10/98FA43DC8007/H323/generic_send_setup: ====== PI = 0
Aug 19 02:06:37.097: //10/98FA43DC8007/H323/generic_send_setup: Send infoXCap=128,
infoXRate=157, rateMult=89
Aug 19 02:06:37.097: //10/98FA43DC8007/H323/generic_send_setup: src address =
172.16.13.16; dest address = 172.31.85.107
Aug 19 02:06:37.097: //10/98FA43DC8007/H323/cch323_h225_set_new_state: Changing from
H225_IDLE state to H225_REQ_FS_SETUP state
Aug 19 02:06:37.101: TCP0: FIN processed
```

The following lines show that the router was able to initialize the TCP session to Cisco CallManager, and the session went into the established state. After the router sent an H225 SETUP message it received a TCP RESET message to tear down the TCP session, indicating Cisco CallManager dropped the connection.

```
Aug 19 02:06:37.101: TCP0: state was ESTAB -> CLOSEWAIT [11001 -> 172.31.85.107(1720)]
Aug 19 02:06:37.101: TCP0: RST received, Closing connection
Aug 19 02:06:37.101: TCP0: state was CLOSEWAIT -> CLOSED [11001 -> 172.31.85.107(1720)]
Aug 19 02:06:37.105: ISDN Se0:23 Q931: TX -> CALL_PROC pd = 8  callref = 0x8238
        Channel ID i = 0xA98381
                Exclusive, Channel 1
```

```
Aug 19 02:06:37.105: %VOICE_IEC-3-GW: H323: Internal Error (Socket error):
IEC=1.1.186.5.7.6 on callID 10 GUID=98FA43DCD12011D7800700B0640E6622
Aug 19 02:06:37.105: TCB 0x63604218 destroyed
Aug 19 02:06:37.105: //10/98FA43DC8007/H323/run_h225_sm: Received event H225_EV_CONN_LOST
while at state H225_REQ_FS_SETUP
Aug 19 02:06:37.113: //10/98FA43DC8007/H323/run_h225_sm: Received event H225_EV_RELEASE
while at state H225_REQ_FS_SETUP
Aug 19 02:06:37.113: //10/98FA43DC8007/H323/cch323_h225_set_new_state: Changing from
H225_REQ_FS_SETUP state to H225_IDLE state
Aug 19 02:06:37.121: ISDN Se0:23 Q931: TX -> DISCONNECT pd = 8  callref = 0x8238
        Cause i = 0x80A6 - Network out of order
Aug 19 02:06:37.133: ISDN Se0:23 Q931: RX <- RELEASE pd = 8  callref = 0x0238
        Cause i = 0x82E4 - Invalid information element contents
Aug 19 02:06:37.137: ISDN Se0:23 Q931: TX -> RELEASE_COMP pd = 8  callref = 0x8238
```

Step 9 Verify Cisco CallManager setting for the H.323 gateway to determine if the H.225 session was rejected by Cisco CallManager because the wrong IP address was configured for the H.323 gateway. Configure the correct IP address for the H.323 gateway, 172.16.13.16. For more information on CallManager configuration and IP address configuration, see the *Cisco CallManager Administration Guide*, Release 3.3(2) "Device Configuration" chapter, "Adding Gateways to Cisco CallManager" and "Adding a Cisco IOS H.323 Gateway" sections.

Step 10 Use **debug** commands to verify that the next call completes, as shown in the following partial **debug** output:

```
Aug 19 02:10:36.707: ISDN Se0:23 Q931: RX <- SETUP pd = 8  callref = 0x0239
        Bearer Capability i = 0x8090A2
                Standard = CCITT
                Transer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Calling Party Number i = 0x2181, '4085550111'
                Plan:ISDN, Type:National
        Called Party Number i = 0x80, '83101'
                Plan:Unknown, Type:Unknown
Aug 19 02:10:36.711: VDEV_ALLOCATE: 1/2 is allocated
Aug 19 02:10:36.711: csm_vtsp_init_tdm: vdev@ 0x6355B98C, voice_vdev@ 0x6355B918
Aug 19 02:10:36.711: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 1, dsp 4,
dsp_channel 1
Aug 19 02:10:36.711: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 1, bank
2, bp_channel 3, bp_stream 255
Aug 19 02:10:36.711: VDEV_DEALLOCATE: slot 1, port 12 is deallocated
Aug 19 02:10:37.155: TCB63604A4C created
Aug 19 02:10:37.159: TCP: sending SYN, seq 3088300316, ack 0
Aug 19 02:10:37.159: TCP0: Connection to 172.69.85.107:1720, advertising MSS 536
Aug 19 02:10:37.159: TCP0: state was CLOSED -> SYNSENT [11003 -> 172.69.85.107(1720)]
Aug 19 02:10:37.163: TCP0: state was SYNSENT -> ESTAB [11003 -> 172.69.85.107(1720)]
Aug 19 02:10:37.163: TCP0: Connection to 172.69.85.107:1720, received MSS 1460, MSS is 536
Aug 19 02:10:37.163: //12/281160848008/H323/run_h225_sm: Received event H225_EV_SETUP
while at state H225_IDLE
Aug 19 02:10:37.163: //12/281160848008/H323/check_qos_and_send_setup: Setup ccb 0x635F80E8
!
!
!
Aug 19 02:10:37.171: ISDN Se0:23 Q931: TX -> CALL_PROC pd = 8  callref = 0x8239
        Channel ID i = 0xA98381
                Exclusive, Channel 1
Aug 19 02:10:38.151: //-1/xxxxxxxxxxxx/H323/cch323_h225_receiver: Received msg of type
CALLPROCIND_CHOSEN
Aug 19 02:10:38.155: //12/281160848008/H323/callproc_ind: ====== PI = 0
Aug 19 02:10:38.155: //12/281160848008/H323/callproc_ind: Call Manager detected
```

```
Aug 19 02:10:38.155: //12/281160848008/H323/cch323_h225_receiver: CALLPROCIND_CHOSEN: src
address = 172.16.13.16; dest address = 172.69.85.107
Aug 19 02:10:38.155: //12/281160848008/H323/run_h225_sm: Received event
H225_EV_CALLPROC_IND while at state H225_REQ_FS_SETUP
Aug 19 02:10:41.347: TCP0: state was SYNSENT -> ESTAB [11004 -> 172.69.85.107(1778)]
Aug 19 02:10:41.347: TCP0: Connection to 172.69.85.107:1778, received MSS 1460, MSS is 536
Aug 19 02:10:41.699: //12/281160848008/H323/run_h225_sm: Received event
H225_EV_H245_SUCCESS while at state H225_WAIT_FOR_H245
Aug 19 02:10:41.703: //12/281160848008/H323/cch323_h225_set_new_state: Changing from
H225_WAIT_FOR_H245 state to H225_ACTIVE state
Aug 19 02:10:41.703: //12/281160848008/H323/setup_cfm_notify: status = 4800261B
Aug 19 02:10:41.703: //12/281160848008/H323/generic_setup_cfm_notify: ====== PI = 0;
status = C800261B
```

In the next lines, the call connects and two-way communication is established.

```
Aug 19 02:10:41.711: ISDN Se0:23 Q931: TX -> CONNECT pd = 8  callref = 0x8239
Aug 19 02:10:41.719: ISDN Se0:23 Q931: RX <- CONNECT_ACK pd = 8  callref = 0x0239
```

The following lines show that after a two-minute call, the IP phone user hangs up and the call is disconnected with normal call clearing:

```
Aug 19 02:10:43.635: TCP0: RST received, Closing connection
Aug 19 02:10:43.635: TCP0: state was ESTAB -> CLOSED [11004 -> 172.69.85.107(1778)]
Aug 19 02:10:43.635: TCB 0x6361C070 destroyed
!
!
!
Aug 19 02:10:43.663: TCP0: sending FIN
Aug 19 02:10:43.663: TCP0: state was FINWAIT1 -> FINWAIT2 [11003 -> 172.69.85.107(1720)]
Aug 19 02:10:43.663: TCP0: FIN processed
Aug 19 02:10:43.663: TCP0: state was FINWAIT2 -> TIMEWAIT [11003 -> 172.69.85.107(1720)]
Aug 19 02:10:43.671: ISDN Se0:23 Q931: TX -> DISCONNECT pd = 8  callref = 0x8239
        Cause i = 0x8090 - Normal call clearing
Aug 19 02:10:43.679: ISDN Se0:23 Q931: RX <- RELEASE pd = 8  callref = 0x0239
Aug 19 02:10:43.683: ISDN Se0:23 Q931: TX -> RELEASE_COMP pd = 8  callref = 0x8239
Router#
Router# undebug all
All possible debugging has been turned off.
```

Troubleshooting Resources

To troubleshoot problems with voice networks, the following resources are required:

Troubleshooting Tools

This section presents information about the variety of tools available to assist you in troubleshooting your voice network, including information on using router diagnostic commands, Cisco network management tools, and third-party troubleshooting tools.

Using Router Diagnostic Commands

Cisco routers provide numerous integrated commands to assist you in monitoring and troubleshooting your internetwork.

- **show** commands help you monitor installation behavior and normal network behavior, and isolate problem areas.
- **debug** commands help you isolate protocol and configuration problems.
- **ping** commands help you determine connectivity between devices on your network.
- **trace** commands provide a method of determining the route by which packets reach their destination.

Using show Commands

show commands are powerful monitoring and troubleshooting tools. You can use **show** commands to perform a variety of functions:

- Monitor router behavior during initial installation.
- Monitor normal network operation.
- Isolate problem interfaces, nodes, media, or applications.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Determine when a network is congested.
- Determine the status of servers, clients, or other neighbors.

The following are some of the most common **show** commands:

- **show version**—Displays the configuration of the system hardware, the software version, the names and sources of configuration files, and the boot images.
- **show running-config**—Displays the router configuration currently running.
- **show startup-config**—Displays the router configuration stored in NVRAM.
- **show interfaces**—Displays statistics for all interfaces configured on the router or access server. The resulting output varies, depending on the network for which an interface has been configured.
- **show controllers**—Displays statistics for interface card controllers.
- **show flash**—Displays the layout and contents of flash memory.
- **show buffers**—Displays statistics for the buffer pools on the router.
- **show memory summary**—Displays memory pool statistics and summary information about the activities of the system memory allocator, and gives a block-by-block listing of memory use.
- **show process cpu**—Displays information about the active processes on the router.
- **show stacks**—Displays information about the stack utilization of processes and interrupt routines, and the reason for the last system reboot.
- **show cdp neighbors**—Provides reachability information for directly connected Cisco devices. This is an extremely useful tool for determining the operational status of the physical and data link layers. Cisco Discovery Protocol (CDP) is a proprietary data link layer protocol.
- **show debugging**—Displays information about the type of debugging that is enabled for your router.

You can always use the **?** at the command line for a list of subcommands.

Like the **debug** commands, some of the **show** commands listed are accessible only at the router's privileged EXEC mode. Debug command usage is explained further in the "Using debug Commands" section on page 2 and also in the "Debug Command Output on Cisco IOS Voice Gateways" chapter.

Hundreds of other **show** commands are available. For details on using and interpreting the output of specific **show** commands, refer to the Cisco IOS command references.

Using debug Commands

The **debug** privileged EXEC commands can provide a wealth of information about the traffic seen (or *not* seen) on an interface, error messages generated by nodes on the network, protocol-specific diagnostic packets, and other useful troubleshooting data.

The following steps are a summary of **debug** command usage. For detailed **debug** command usage, see the "Debug Command Output on Cisco IOS Voice Gateways" chapter.

To access and list the privileged EXEC commands, enter this code:

```
Router> enable
Password: XXXXXX
Router# ?
```

Note the change in the router prompts here. The # prompt (instead of the normal > prompt) indicates that the router is in privileged EXEC mode (enable mode).

Caution Exercise care when using **debug** commands. Many **debug** commands are processor-intensive and can cause serious network problems (such as degraded performance or loss of connectivity) if they are enabled on an already heavily loaded router. When you finish using a **debug** command, remember to disable it with its specific **no debug** command (or use the **no debug all** command to turn off all debugging).

Use **debug** commands to isolate problems, not to monitor normal network operation. Because the high processor overhead of **debug** commands can disrupt router operation, you should use them only when you are looking for specific types of traffic or problems, and have narrowed your problems to a likely subset of causes.

Output formats vary with each **debug** command. Some generate a single line of output per packet, and others generate multiple lines of output per packet. Some generate large amounts of output, and others generate only occasional output. Some generate lines of text, and others generate information in field format.

To minimize the negative impact of using **debug** commands, follow this procedure:

Step 1 Use the **no logging console** global configuration command on your router. This command disables all logging to the console terminal.

Step 2 Telnet to a router port and enter the **enable** EXEC command. The **enable** EXEC command places the router in the privileged EXEC mode. After entering the **enable** password, you receive a prompt that consists of the router name with a # symbol.

Step 3 Use the **terminal monitor** command to copy **debug** command output and system error messages to your current terminal display.

By redirecting output to your current terminal display, you can view **debug** command output remotely, without being connected through the console port.

If you use **debug** commands at the console port, character-by-character processor interrupts are generated, maximizing the processor load already caused by the use of **debug**.

If you intend to keep the output of the **debug** command, spool the output to a file. The procedure for setting up such a **debug** output file is described in the "Debug Command Output on Cisco IOS Voice Gateways" chapter.

This book refers to specific **debug** commands that are useful when you are troubleshooting specific problems. Complete details regarding the function and output of **debug** commands are provided in the *Cisco IOS Debug Command Reference*.

In many situations, using third-party diagnostic tools can be more useful and less intrusive than using **debug** commands. For more information, see the "Third-Party Troubleshooting Tools" section on page 6.

Using the ping Commands

To check host reachability and network connectivity, use the **ping** command, which can be invoked from both user EXEC mode and privileged EXEC mode. After you log in to the router or access server, the router is automatically in user EXEC command mode. The EXEC commands available at the user level are a subset of those available at the privileged level. In general, the user EXEC commands enable you

to connect to remote devices, change terminal settings on a temporary basis, perform basic tests, and list system information. The **ping** command can be used to confirm basic network connectivity on a variety of networks.

For IP, the **ping** command sends Internet Control Message Protocol (ICMP) echo messages. ICMP is the Internet protocol that reports errors and provides information relevant to IP packet addressing. If a station receives an ICMP Echo message, it sends an ICMP echo reply message back to the source.

The extended command mode of the **ping** command permits you to specify the supported IP header options. This mode allows the router to perform a more extensive range of test options. To enter ping extended command mode, enter **yes** at the extended commands prompt of the **ping** command.

It is a good idea to use the **ping** command when the network is functioning properly to see how the command works under normal conditions, so that you have a basis for comparison when you are troubleshooting.

For detailed information on using the **ping** and extended **ping** commands, refer to the *Cisco IOS Configuration Fundamentals and Network Management Command Reference*.

Using the trace Commands

The **trace** user EXEC command discovers the routes that a router's packets follow when traveling to their destinations. The **trace** privileged EXEC command permits the supported IP header options to be specified, allowing the router to perform a more extensive range of test options.

The **trace** command works by using the error message generated by routers when a datagram exceeds its time-to-live (TTL) value. First, probe datagrams are sent with a TTL value of 1. This value causes the first router to discard the probe datagrams and send back "time exceeded" error messages. The **trace** command then sends several probes and displays the round-trip time for each. After every third probe, the TTL is increased by 1.

Each outgoing packet can result in one of two error messages. A "time exceeded**"** error message indicates that an intermediate router has seen and discarded the probe. A "port unreachable" error message indicates that the destination node has received the probe and discarded it because it could not deliver the packet to an application. If the timer goes off before a response comes in, **trace** prints an asterisk (*****).

The **trace** command terminates when the destination responds, when the maximum TTL is exceeded, or when you interrupt the trace with the escape sequence.

As with **ping**, it is a good idea to use the **trace** command when the network is functioning properly to see how the command works under normal conditions, so that you have a basis for comparison when you are troubleshooting.

For detailed information on using the **trace** and extended **trace** commands, refer to the *Cisco IOS Configuration Fundamentals and Network Management Command Reference*.

Using Cisco Network Management Tools

Cisco offers the CiscoWorks 2000 family of management products that provide design, monitoring, and troubleshooting tools to help you manage your internetwork.

The following tools are useful for troubleshooting internetwork problems:

- CiscoView provides dynamic monitoring and troubleshooting functions, including a graphical display of Cisco devices, statistics, and comprehensive configuration information.

- Internetwork Performance Monitor (IPM) empowers network engineers to proactively troubleshoot network response times utilizing real-time and historical reports.
- The TrafficDirector RMON application, a remote monitoring tool, enables you to gather data, monitor activity on your network, and find potential problems.
- The VlanDirector switch management application provides an accurate picture of your VLANs.

CiscoView

CiscoView graphical management features provide dynamic status, statistics, and comprehensive configuration information for Cisco internetworking products (switches, routers, hubs, concentrators, and access servers). CiscoView aids network management by displaying a physical view of Cisco devices and color-coding device ports for at-a-glance port status, allowing you to quickly grasp essential information. Features include the following:

- Graphical displays of Cisco products from a central location, giving network managers a complete view of Cisco products. You do not need to physically check devices at remote sites.
- A continuously updated physical view of routers, hubs, switches, or access servers in a network, regardless of physical location.
- Updated real-time monitoring and tracking of key information relating to device performance, traffic, and usage, with metrics such as utilization percentage, frames sent and received, errors, and a variety of other device-specific indicators.
- The capability to modify configurations such as trap, IP route, VLAN, and bridge configurations.

Internetwork Performance Monitor

The Internetwork Performance Monitor (IPM) is a network management application that enables you to monitor the performance of multiprotocol networks. IPM measures the response time and availability of IP networks on a hop-by-hop (router-to-router) basis. It also measures response time between routers and the mainframe in Systems Network Architecture (SNA) networks.

Use IPM to perform the following tasks:

- Troubleshoot problems by checking the network latency between devices.
- Send Simple Network Management Protocol (SNMP) traps and SNA alerts when a user-configured threshold is exceeded, when a connection is lost and reestablished, or when a timeout occurs.
- Analyze potential problems before they occur by accumulating statistics, which are used for modeling future network topologies.
- Monitor response time between two network endpoints.

The IPM product is composed of three parts: the IPM server, the IPM client application, and the response time reporter (RTR) feature of the Cisco IOS software.

TrafficDirector RMON Application

The TrafficDirector advanced packet filters let you monitor all seven layers of network traffic. Using Cisco IOS embedded RMON agents and SwitchProbe standalone probes, managers can view enterprise-wide network traffic from the link, network, transport, or application layer. The TrafficDirector multilayer traffic summary provides a quick, high-level assessment of network loading

and protocol distributions. Network managers can “zoom in” on a specific segment, ring, switch port, or trunk link and apply real-time analysis and diagnostic tools to view hosts, conversations, and packet captures.

TrafficDirector threshold monitoring enables you to implement a proactive management environment. Thresholds for critical MIB variables are set within the RMON agent. When these thresholds are exceeded, traps are sent to the appropriate management station to notify the network administrator of an impending problem.

VlanDirector Switch Management Application

The VlanDirector switch management application simplifies VLAN port assignment and offers other management capabilities for VLANs. VlanDirector offers the following features for network administrators:

- Accurate representation of the physical network for VLAN design and configuration verification
- Capability to obtain VLAN configuration information on a specific device or link interface
- Discrepancy reports on conflicting configurations
- Capability to use system-level VLANs to troubleshoot and identify individual device configurations that are in error
- Quick detection of changes in VLAN status of switch ports
- User authentication and write protection security

Third-Party Troubleshooting Tools

In many situations, third-party diagnostic tools can be more useful than commands that are integrated into the router. For example, enabling a processor-intensive **debug** command can be extremely detrimental in an environment experiencing excessively high traffic levels. Attaching a network analyzer to the suspect network is less intrusive and is more likely to yield useful information without interrupting the operation of the router. The following are some typical third-party troubleshooting tools used for troubleshooting internetworks:

- Volt-ohm meters, digital multimeters, and cable testers are useful for testing the physical connectivity of your cable plant.
- Time domain reflectometers (TDRs) and optical time domain reflectometers (OTDRs) are devices that assist in the location of cable breaks, impedance mismatches, and other physical cable plant problems.
- Breakout boxes, fox boxes, bit error rate testers (BERTs), and block error rate testers (BLERTs) are useful for troubleshooting problems in peripheral interfaces.
- Network monitors provide an accurate picture of network activity over a period of time by continuously tracking packets crossing a network.
- Network analyzers such as sniffers decode problems at all seven OSI layers. The problems can be identified automatically in real-time and categorized by how critical they are, providing a clear view of network activity.

Volt-Ohm Meters, Digital Multimeters, and Cable Testers

Volt-ohm meters and digital multimeters are at the lower end of the spectrum of cable-testing tools. These devices measure parameters such as AC and DC voltage, current, resistance, capacitance, and cable continuity. They are used to check physical connectivity.

Cable testers (scanners) also enable you to check physical connectivity. Cable testers are available for shielded twisted-pair (STP), unshielded twisted-pair (UTP), 10BASE-T, and coaxial and twinax cables. A given cable tester might be capable of performing any of the following functions:

- Test and report on cable conditions, including near-end crosstalk (NEXT), attenuation, and noise
- Use TDR and perform traffic monitoring and wire map functions
- Display MAC–layer information about LAN traffic, provide statistics such as network utilization and packet error rates, and perform limited protocol testing (for example, TCP/IP tests such as **ping**)

Similar testing equipment is available for fiber-optic cable. Because of the relatively high cost of this cable and its installation, fiber-optic cable should be tested both before installation (on-the-reel testing) and after installation. Continuity testing of the fiber requires either a visible light source or a reflectometer. Light sources capable of providing light at the three predominant wavelengths—850 nanometers (nm), 1300 nm, and 1550 nm—are used with power meters that can measure the same wavelengths and test attenuation and return loss in the fiber.

TDRs and OTDRs

At the top end of the cable testing spectrum are time domain reflectometers (TDRs). These devices can quickly locate open and short circuits, crimps, kinks, sharp bends, impedance mismatches, and other defects in metallic cables.

A TDR works by bouncing a signal off the end of the cable. Opens, shorts, and other problems reflect the signal back at different amplitudes, depending on the problem. A TDR measures how much time it takes for the signal to reflect and calculates the distance to a fault in the cable. TDRs can be used to measure the length of a cable, and some TDRs can also calculate the propagation rate based on a configured cable length.

Fiber-optic measurement is performed by an optical TDR (OTDR). OTDRs can accurately measure the length of the fiber, locate cable breaks, measure the fiber attenuation, and measure splice or connector losses. An OTDR can be used to take the signature of a particular installation, noting attenuation and splice losses. This baseline measurement can then be compared with future signatures when a problem in the system is suspected.

Breakout Boxes, Fox Boxes, BERTs and BLERTs

Breakout boxes, fox boxes, and bit/block error rate testers (BERTs/BLERTs) are digital interface testing tools used to measure the digital signals present at PCs, printers, modems, the channel service unit/data service unit (CSU/DSU), and other peripheral interfaces. These devices can monitor data line conditions, analyze and trap data, and diagnose problems common to data communication systems. Traffic from data terminal equipment (DTE) through data communications equipment (DCE) can be examined to help isolate problems, identify bit patterns, and ensure that the proper cabling has been installed. These devices cannot test media signals such as Ethernet, Token Ring, or FDDI.

Network Monitors

Network monitors continuously track packets crossing a network, providing an accurate picture of network activity at any moment, or a historical record of network activity over a period of time. They do not decode the contents of frames. Monitors are useful for baselining, in which the activity on a network is sampled over a period of time to establish a normal performance profile.

Monitors collect information such as packet sizes, the number of packets, error packets, overall usage of a connection, the number of hosts and their MAC addresses, and details about communications between hosts and other devices. This data can be used to create profiles of LAN traffic and to assist in locating traffic overloads, planning for network expansion, detecting intruders, establishing baseline performance, and distributing traffic more efficiently.

Network Analyzers

A network analyzer (also called a protocol analyzer or “sniffer”) decodes the various protocol layers in a recorded frame and presents the frames as readable abbreviations or summaries. The analyzer indicates which layer is involved (physical, data link, and so forth) and what function each byte or byte content serves.

Most network analyzers can perform many of the following functions:

- Filter traffic that meets certain criteria so that, for example, all traffic to and from a particular device can be captured
- Time-stamp captured data
- Present protocol layers in an easily readable form
- Generate frames and send them onto the network
- Incorporate an “expert” system in which the analyzer uses a set of rules, combined with information about the network configuration and operation, to diagnose and solve, or offer potential solutions for network problems

Other Troubleshooting Resources

Refer to the following resources to assist in troubleshooting:

- Internetworking Troubleshooting Handbook, page 8
- TAC Troubleshooting Tools on Cisco.com, page 8

Internetworking Troubleshooting Handbook

The *Internetworking Troubleshooting Handbook* is a good resource for general network troubleshooting. It is available online:

http://www.cisco.com/univercd/cc/td/doc/cisintwk/itg_v1/index.htm

TAC Troubleshooting Tools on Cisco.com

A variety of troubleshooting tools can be found at the following website:

http://www.cisco.com/kobayashi/support/tac/tools_trouble.shtml

Note You need to register to use these tools on Cisco.com.

This site has links to the following tools:

- 2600/3600/3700 Memory Calculator—Compute the memory required for Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series routers.
- Bitswapping Tool—Instantly convert Ethernet MAC addresses to Token Ring MAC addresses.
- BPX/IGX Firmware Compatibility Tool—Evaluates BPX and IGX firmware versions for switch software compatibility using the dspcds screen output.
- Cisco-centric Open Source Initiative (COSI)—An open source exchange community where Cisco-centric developers and customers publish, discuss, and release their open source tools, scripts, and utilities.
- Cisco Feature Navigator II—A web-based application that allows you to quickly find the right Cisco IOS software release for the features you want to run on your network.
- Command Lookup Tool—Look up a detailed description or configuration guidelines for a particular Cisco IOS, Cisco Catalyst, or PIX command. This tool is for Cisco IOS software releases 12.0, 12.1, and 12.2 only.
- Custom Document Generator for Cisco IOS—Generate customized command reference documents.
- Error Message Decoder—Look up explanations for console error messages listed in the *Cisco Software System Messages* guide.
- IOS Upgrade Planner—Browse for your preferred software.
- IP Subnet Calculator—Plan your subnetting and addressing strategy online.
- Networking Professionals Connection—Discussion forums, Tech Talks, and Ask the Expert forums are available in which you can share questions, suggestions, and information about networking solutions, products, and technologies.
- Output Interpreter—Receive instant troubleshooting analysis and course of action for your router, switch, or PIX device using collected **show** command output.
- Product Alert Tool—Set up a profile to receive email updates about reliability, safety, network security, and end-of-sale issues for the Cisco products you specify.
- RIF Decoder Tool—Use the RIF decoder to automatically interpret and decode a hex string.
- SNMP Object Navigator—Translate SNMP object identifiers (OIDs) into object names, search object names and descriptions, browse the OID tree, and download MIB files.
- Software Advisor—Choose software for your network device.
- Software Bug Toolkit—Search for software bugs based on version and feature sets or bug ID.
- Software Search Engine—Search for software images and image metadata.
- Stack Decoder for Cisco IOS—This link takes you to the Cisco Output Interpreter tool, which now offers all the functionality formerly available in the Stack Decoder.
- TAC Case Collection (Troubleshooting Assistant)—Interactively diagnose common problems involving hardware, configuration, and performance issues with solutions provided by TAC engineers.
- Voice Codec Bandwidth Calculator—Use the Voice Codec Bandwidth Calculator to determine the bandwidth used by different codecs with various voice protocols over different media.

- Wireless Troubleshooting Center—Interactive support for questions related to Cisco Aironet wireless LAN products.

Troubleshooting Voice Telephony

Troubleshooting Analog Voice Interfaces to the IP Network

If you are troubleshooting an analog connection, you must understand what type of circuit and interface your voice port is using. Analog voice port interfaces connect routers in packet-based networks to analog two-wire or four-wire analog circuits in telephony networks. Two-wire circuits connect to analog telephone or fax devices, and four-wire circuits connect to PBXs. Analog voice telephony interfaces include foreign exchange office (FXO), foreign exchange station (FXS), and receive and transmit (E&M). Direct Inward Dialing (DID) is a service offered by telephone companies that enables callers to dial directly to an extension on a PBX without the assistance of an operator or automated call attendant.

To troubleshoot analog voice interfaces, see the following sections:

If you are troubleshooting a connection to a PBX, you might find the PBX interoperability notes useful. These notes contain configuration information for Cisco gateways and several types of PBXs. To access these notes, use the following website:

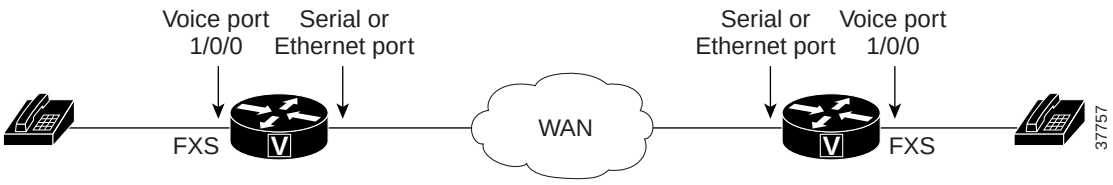
http://www.cisco.com/warp/public/779/largeent/avvid/inter_operability/

FXS Interfaces

An FXS interface connects the router or access server to end-user equipment such as telephones, fax machines, and modems. The FXS interface supplies ring, voltage, and dial tone to the station and includes an RJ-11 connector for basic telephone equipment, keysets, and PBXs. In Figure 10, FXS signaling is used for end-user telephony equipment, such as a telephone or fax machine.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Figure 10 FXS Signaling Interfaces



If you are having trouble with an FXS port, check the following sections:

- FXS Hardware Troubleshooting, page 2
- Ring Voltage Problems, page 4
- Unbreakable Dial Tone, page 6

FXS Hardware Troubleshooting

An FXS interface connects directly to a standard telephone, fax machine, or similar device and supplies ring, voltage, and dial tone.

Troubleshoot FXS hardware by checking the following sections:

- Software Compatibility, page 2
- Cabling, page 2
- Shutdown Port, page 3
- Disabling a Port on a Multiple Port Card, page 4

Software Compatibility

To ensure that your FXS card is compatible with your software, check the following:

- For network modules inserted into Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series, check the compatibility tables in the "Overview of Cisco Network Modules" chapter in the *Cisco Network Modules Hardware Installation Guide.*
- For interface cards inserted into Cisco 1600 series, Cisco 1700 series, Cisco 2600 series, Cisco 3600 series, Cisco 3700 series, and Cisco ICS 7750 platforms, check the compatibility tables in the "Overview of Cisco Interface Cards" chapter in the *Cisco Interface Cards Installation Guide.*

Cabling

Two types of cabling are supported for Cisco FXS interfaces. They are described in the following sections:

- RJ-11 Connectors, page 3
- RJ-21 Connectors on the High-Density Analog Telephony Network Module, page 3

Note For FXS connections, use a 2-wire (RJ-11) cable. A 4-wire cable can cause the second port to busy out.

RJ-11 Connectors

The two-port and four-port FXS interface cards support the RJ-11 connector. Illustrations of the connector ports are shown in Figure 11 and Figure 12. Information about LEDs can be found in the "Connecting Voice Interface Cards to a Network" chapter of the *Cisco Interface Card Hardware Installation Guide*.

Figure 11 Two-Port FXS Card Front Panel

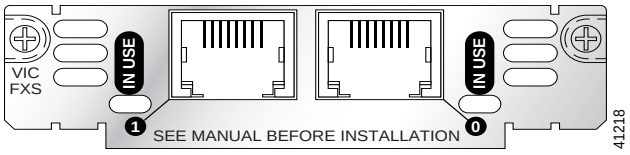
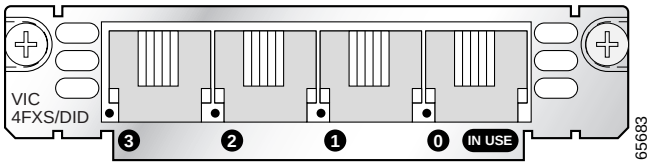


Figure 12 Four-Port FXS/DID Card Front Panel

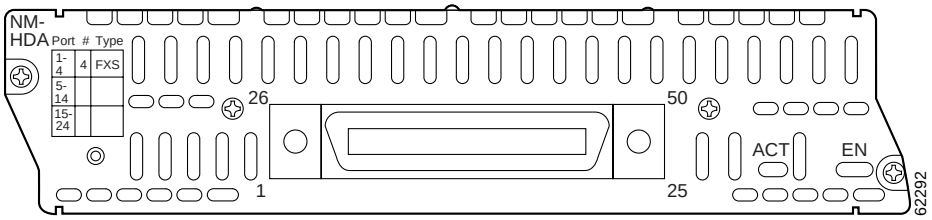


For information about the VIC-2FXS interface card, refer to *Understanding Foreign Exchange Station (FXS) Voice Interface Cards*, document ID 7938.

RJ-21 Connectors on the High-Density Analog Telephony Network Module

The High-Density Analog Telephony network module supports an RJ-21 connector. This network module supports both FXS and FXO traffic. An illustration of the connector port is shown in Figure 13. Information about LEDs and pinouts can be found in the "Connecting High-Density Analog Telephony Network Modules to a Network" chapter of the *Cisco Network Modules Hardware Installation Guide*.

Figure 13 High-Density Analog Telephony Network Module



Shutdown Port

If the port is not working, be sure the port is not shut down. Enter the **show voice port** command with the voice port number that you are troubleshooting. The output will tell you:

- If the voice port is up. If it is not, use the **no shutdown** command to make it active.
- What parameter values have been set for the voice port, including default values (which do not appear in the output from **the show running-config command**). If these values do not match those of the telephony connection you are making, reconfigure the voice port.

Disabling a Port on a Multiple Port Card

If you shut down a port on a multiple-port card, you can disable all of the ports on that card. If only one port is bad and the others are working, in many cases you can disable the bad port and then use the working ports until a replacement arrives. To disable a bad port, use one of the following methods:

- On a Cisco universal gateway, such as the Cisco AS5350, Cisco AS5400, Cisco AS5800, and Cisco AS5850, busy out the port using the **busyout** command. This setting allows the port to be taken out of service without disrupting the Cisco IOS configuration. See the product documentation for details:
  - Cisco AS5350 product documentation
  - Cisco AS5400 product documentation
  - Cisco AS5800 product documentation
  - Cisco AS5850 product documentation
- On other Cisco gateways, remove the port from the dial peer. Refer to "Dial Peer Features and Configuration" in the *Dial Peer Configuration on Voice Gateway Routers* document to configure the dial peer.

Ring Voltage Problems

Telephone exchanges and FXSs need to supply DC battery and AC ringing to enable the connected telephone equipment to transmit speech energy and to power the telephone equipment's ringing device. This section discusses what voltages are supplied by various Cisco FXS interfaces and how to overcome some known issues regarding voltage levels.

Ringing Voltages

The industry standard for PBX and key systems requires that the ring detection circuit be able to detect a ringing signal as low as 40 Vrms. This voltage takes into account the effects of load and cabling voltage drop on a ringing signal generated from a central office (CO). Conversely, the CO (exchange) must supply ringing with enough power to drive the maximum load over the maximum cable length. In order to meet this requirement, a CO-based unit must present a ringing signal with an amplitude of approximately 85 to 100 Vrms. Cisco voice gateways are intended for use as on premise services (ONS) equipment that is colocated or fairly close to equipment that detects ringing, so it can therefore use a lower ringing voltage and still meet the 40 Vrms 5 Ringer Equivalence Number (REN) requirement.

Idle Battery Voltage

Cisco voice gateways were designed for ONS connections and by default the FXS interface supplies either –24 Vdc or –36 Vdc idle battery, whereas off premise services, such as a CO, would require voltages of –48 V because it might have to interconnect over much greater cable lengths. Certain Cisco FXS interfaces can be configured to supply higher voltages.

Idle Line Voltages

Table 24 shows idle line voltages supplied by various Cisco gateway FXS interfaces.

Table 24 FXS Idle Voltage

FXS Interface	Idle Voltage
VG248	–36V
VIC-2FXS	–26V
VIC-2DID	–24V (low) –48V (high)
ASI 81 and ASI 160	–24V (low) –48V (high)
IAD 24xx-FXS	–24V (low) –48V (high)
1730 IAD	–24V (low) –48V (high)
VIC-4FXS/DID	–24V (low) –48V (high)

Ring Voltage Problems

Voltage problems can cause three types of problems:

- Answering and Call Initiation Problems with Automated Telephony Devices, page 5
- Ringing Problems, page 5
- FXS Ring Failure in the United Kingdom, page 6

Certain automated devices, such as fax machines, answer machines, multiline phones and voice mail systems, look at the line voltage in order to deduce if the line is busy or idle. If another device is off hook, then the line voltage drops, and the automated system does not answer or initiate a call. If the threshold being used is close to –24 V or higher, this can cause the device not to work as expected.

Certain phones might not ring when the default ring voltage and ring frequency are applied from the Cisco FXS interface.

Answering and Call Initiation Problems with Automated Telephony Devices

In voice port configuration mode, configure the **idle-voltage** command on the voice port of the FXS to increase idle battery voltage from –24 V to –48 V. The **idle-voltage low** setting designates -24 V and the **idle-voltage high** setting designates -48 V.

Note This option is not available on VG248, VIC-2FXS, and WS-x6624 FXS interfaces.

Ringing Problems

Phone manufacturers sometimes use frequency filters known as antitinkle circuits to prevent ringer devices from sounding while the user is dialing. Sometimes it is necessary to adjust the frequency of the ring to suit the connected device.

Configure the ring frequency for Cisco modular access routers by issuing the following command:

```
Router(config-voiceport)# ring frequency ?
25 ring frequency 25 Hertz
50 ring frequency 50 Hertz
```

Configure the ring frequency for the Cisco IAD2400 platform by issuing the following command:

```
Router(config-voiceport)# ring frequency ?
20 ring frequency 20 Hertz
30 ring frequency 30 Hertz
```

To prevent ringer devices from sounding, you can also provide a voltage threshold so that the lower voltages, which can be produced during dialing, are ignored. Increasing the voltage can overcome this.

Configure the DC offset voltage on Cisco IAD2400 series routers by issuing the following command:

```
Router(config-voiceport)# ring dc-offset ?
10-volts Ring DC offset 10 volts
20-volts Ring DC offset 20 volts
24-volts Ring DC offset 24 volts
```

Note This command sequence can be used only for Cisco IAD2400 series routers. The 24-V ring DC offset setting is available for Cisco IOS 12.2(11)T and later releases.

FXS Ring Failure in the United Kingdom

A telephone approved for the United Kingdom might fail to ring when connected to a Cisco FXS port. The failure results from a physical interoperability issue and is independent of Cisco hardware or software. British Telecom did not implement RJ-11 type connectors when it adopted plug-and-socket connection methodology. RJ-11 connectors allow parallel connectivity for the transmission path and the ringer circuit. They were not used because older telephones needed to have their ringer circuits connected in series due to a requirement for high current.

Outside the United Kingdom, ringer circuitry is self-contained in each phone. The U.K. implementation puts the capacitor, which provides the AC ring path, and the antitinkle feature (prevents the bell or ringer from sounding when pulse dialing is used) externally in the first socket, connected to the local loop.

In the United Kingdom, certain British Approval Board for Telecommunications (BABT) telephones fail to ring when they are connected to FXS ports on Cisco voice-enabled routers and switches. Outgoing calls can be made and voice communication in both directions can be established. However, incoming calls do not ring the telephone. These telephones functioned correctly before they were connected to the FXS ports.

Because a proprietary connection system is implemented, you must use an adapter to connect the telephone to an FXS port. The adapter must be a *master* that contains the capacitor, or the telephone fails to ring.

For a schematic and more information, refer to *Understanding Why Telephones in the United Kingdom Connected to Cisco FXS Interfaces May Fail to Ring*, document ID 25800.

Unbreakable Dial Tone

A common problem encountered in a VoIP network is being unable to break dial tone. The router seizes a line on the local PBX but when digits are dialed, the dial tone stays. The calling party is unable to pass the dual-tone multifrequency (DTMF) tones or digits to the terminating device, resulting in callers being unable to dial the desired extension or interact with a device that needs DTMF tones such as a voice mail or an interactive voice response (IVR) application. This problem can result from a number of sources, for example:

- DTMF tones are not passed.
- DTMF tones are not understood.

- DTMF tones are too distorted to be understood.
- Other signaling and cabling issues occur.

Make sure the dial type is set as DTMF on both the router and the PBX. The FXS port does not pass on the digits; therefore, this setting is not available on an FXS port. However, this setting can be changed on FXO and E&M ports:

```
Router(config-voiceport)# dial-type ?
  dtmf   touch-tone dialer
  mf     mf-tone dialer
  pulse  pulse dialer
```

For more information, refer to *Inability To Break Dialtone in a Voice over IP Network*, document ID 22376.

No LED When Phone Off the Hook

Verify if you have an analog or digital card. If you have an analog card like the VIC-2FXS or the VIC2-4FXS, you might have one of the following problems:

- The port is in a shutdown state.
- The port is in a park state.
- The port is bad.

If you have a digital card like the NM-2V, you might have bad DSPs.

Use the following procedure if there is no LED when your phone is off hook:

Step 1 Check the cable to make sure that it is RJ-11 with two pins for the FXS port.

Step 2 Test the LED using a different phone.

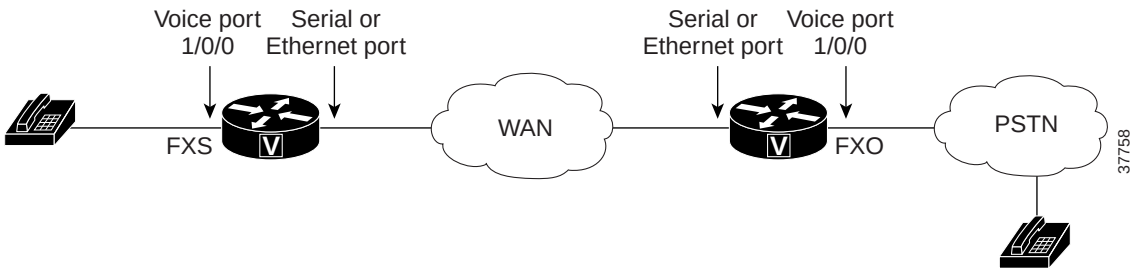
Step 3 Check your Cisco IOS version to make sure that the feature set is either IP Plus or Enterprise Plus.

Step 4 If Steps 1 to 3 do not work, replace the voice interface card (VIC).

FXO Interfaces

An FXO interface is used for trunk, or tie line, connections to a PSTN CO or to a PBX that does not support E&M signaling (when local telecommunications authority permits). This interface is of value for off-premises station applications. Figure 14 shows an FXS connection to a telephone and an FXO connection to the PSTN at the far side of a WAN.

Figure 14 FXS and FXO Signaling Interfaces



If you are having trouble with an FXO port, check the following sections:

- FXO Hardware Troubleshooting, page 8
- FXO Disconnect Failure, page 10
- Troubleshooting FXO Answer and Disconnect Supervision, page 10
- Unbreakable Dial Tone, page 12
- Troubleshooting Caller ID Problems, page 12

FXO Hardware Troubleshooting

An FXO interface is used for trunk connections. Troubleshoot FXO hardware by checking the following sections:

- Software Compatibility, page 8
- Cabling, page 8
- Shutdown Port, page 9
- Disabling a Port on a Multiple Port Card, page 10

Software Compatibility

To ensure that your card is compatible with your software, check the following:

- For network modules inserted into Cisco 2600 series, Cisco 3600 series, and Cisco 3700 series routers, refer to the compatibility tables in the "Overview of Cisco Network Modules" chapter in the *Cisco Network Modules Hardware Installation Guide.*
- For interface cards inserted into Cisco 1600 series, Cisco 1700 series, Cisco 2600 series, Cisco 3600 series, Cisco 3700 series, and Cisco ICS 7750 platforms, refer to the compatibility tables in the "Overview of Cisco Interface Cards" chapter in the *Cisco Interface Cards Installation Guide.*

Cabling

Two types of cabling are supported for Cisco FXO interfaces. They are described in the following sections:

- RJ-11 Connectors, page 3
- RJ-21 Connectors on the High-Density Analog Telephony Network Module, page 3

Note For FXO connections, use a 2-wire (RJ-11) cable. A 4-wire cable can cause the second port to busy out.

RJ-11 Connectors

The two-port and four-port FXO interface cards support the RJ-11 connector. Illustrations of the connector ports are shown in Figure 15 and Figure 16. Information about LEDs can be found in the "Connecting Voice Interface Cards to a Network" chapter of the *Cisco Interface Card Hardware Installation Guide*.

Figure 15 Two-Port FXO Card Front Panel

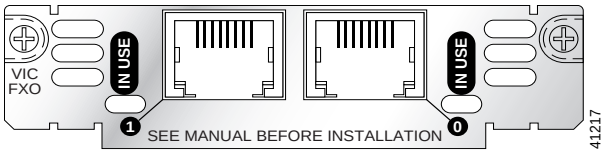
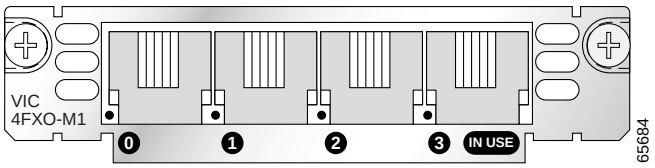


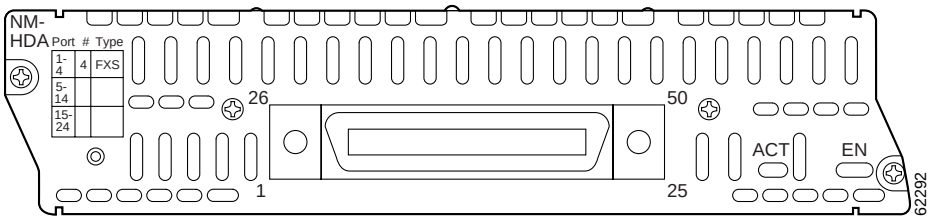
Figure 16 Four-Port FXO Card Front Panel



RJ-21 Connectors on the High-Density Analog Telephony Network Module

The High-Density Analog Telephony network module supports an RJ-21 connector. This network module supports both FXS and FXO traffic. An illustration of the connector port is shown in Figure 17. Information about LEDs and pinouts can be found in the "Connecting High-Density Analog Telephony Network Modules to a Network" chapter of the *Cisco Network Modules Hardware Installation Guide*.

Figure 17 High-Density Analog Telephony Network Module



Shutdown Port

If the port is not working, be sure the port is not shut down. Enter the **show voice port** command with the voice port number that you are troubleshooting. The output will tell you:

- If the voice port is up. If it is not, use the **no shutdown** command to make it active.

- What parameter values have been set for the voice port, including default values (these values do not appear in the output from **the show running-config command**). If these values do not match those of the telephony connection you are making, reconfigure the voice port.

Disabling a Port on a Multiple Port Card

If you shut down a port on a multiple-port card, you can disable all of the ports on that card. If only one port is bad and the others are working, in many cases you can disable the bad port and use the working ports until a replacement arrives. To disable a bad port, use one of the following methods:

- On a Cisco universal gateway, such as the Cisco AS5350, Cisco AS5400, Cisco AS5800, and Cisco AS5850, busy out the port using the **busyout** command. This allows the port to be taken out of service without disrupting the Cisco IOS configuration. Refer to the product documentation for details:
  - For Cisco AS5350 and Cisco AS5400 universal gateways, refer to the "Managing and Troubleshooting the Universal Port Card" chapter in the *Cisco AS5350 and Cisco AS5400 Universal Gateway Software Configuration Guide.*
  - For Cisco AS5800 access servers, refer to the *Managing and Troubleshooting NextPort Services on the AS5800* feature document.
  - For Cisco AS5850 universal gateways, refer to the *Managing Port Services on the Cisco AS5850 Universal Gateway* feature document.
- On other Cisco gateways, remove the port from the dial peer. Refer to "Dial Peer Features and Configuration" in the *Dial Peer Configuration on Voice Gateway Routers* document to configure the dial peer.

FXO Disconnect Failure

When loop-starting signaling is used, an FXO interface looks like a phone to the switch that it is connecting to. The FXO interface closes the loop to indicate off hook. The switch always provides a battery so there is no disconnect supervision from the switch side. Because a switch expects a phone user or modem to hang up the phone when the call is terminated on either side, it also expects the FXO port on the router to hang up. However, the FXO port expects the switch to tell it when to hang up. Because the port relies on the switch, there is no guarantee that a near- or far-end FXO port will disconnect the call once either end of the call has hung up.

The most common symptoms of this problem are phones that continue to ring when the caller has cleared, or FXO ports that remain busy after the previous call should have been cleared.

To troubleshoot this problem, refer to *Understanding FXO Disconnect Problem*, document ID 8120.

Troubleshooting FXO Answer and Disconnect Supervision

This section describes troubleshooting the FXO Answer and Disconnect Supervision feature for analog FXO voice ports. This feature applies to analog FXO voice ports with loop-start signaling connected to PSTNs, PBXs, or key systems.

The FXO Answer and Disconnect Supervision feature enables analog FXO ports to monitor call-progress tones and to monitor voice and fax transmissions returned from a PBX or from the PSTN.

Answer supervision can be accomplished in two ways: by detecting battery reversal, or by detecting voice, fax, or modem tones. If an FXO voice port is connected to the PSTN and battery reversal is supported, use the battery reversal method. Voice ports that do not support battery reversal must use the answer supervision method, in which answer supervision is triggered when the DSP detects voice, modem, or fax transmissions. Configuring answer supervision automatically enables disconnect supervision; however, you can configure disconnect supervision separately if answer supervision is not configured.

Disconnect supervision can be configured to detect call-progress tones sent by the PBX or PSTN (for example, busy, reorder, out-of-service, number-unavailable), or to detect any tone received (for example, busy tone or dial tone). When an incoming call ends, the DSP detects the associated call-progress tone, causing the analog FXO voice port to go on-hook.

This section provides solutions to problems that you might encounter when implementing the FXO Answer and Disconnect Supervision feature.

Typical problems with the answer supervision feature are as follows:

- Call-progress tones such as ringback are not heard by the calling party.

  If any call legs have IVR configured, ensure that the IVR version is 2.0.

- Ringback timer is not initiated or ringback is not detected.
  - The wrong call-progress tone (**cptone**) command is configured on the voice port.
  - The wrong DTMF detection parameters are configured.
  - Custom call-progress tones are assigned to the voice port but ringback tone has not been configured; in this case, the default behavior is not to detect any ringback tones.

- Answer supervision is not triggered.

  Answer supervision—either by battery-reversal detection or by call-progress tone detection—is not configured on the voice-port in use.

- Excessive delay before answer supervision is activated.

  The level on the **sensitivity** parameter in the **supervisory answer dualtone** command is set too low. Configure the sensitivity for **high**.

If incorrect disconnect cause codes are reported, check the following:

- The values configured for custom call-progress tones could be incorrect.
- Overlapping detection frequencies might have been incorrectly specified in the voice class created by the **voice class dualtone-detect-params** command. For example if the **freq-max-deviation** parameter is configured to be 20 Hz, and the **busy** and **reorder** parameters are set for frequencies 350 and 370 respectively, the voice port cannot detect the reorder tone, resulting in an incorrect disconnect cause code.

Note If the frequencies and cadences (including error deviations as defined in the **voice class dualtone-detect-params** command) are the same for multiple call-progress tones, the order of detection is as follows: **busy**, **reorder**, **number-unobtainable**, **out-of-service**, **disconnect**.

If calls are not billed correctly, it might be that answer supervision is not being triggered. For answer supervision to be triggered, voice, fax, or data tones originating at the called-party end must be detected.

To configure the FXO Supervisory Disconnect Tone, refer to the *Voice Port Configuration* document.

Monitoring and Maintaining FXO Answer and Disconnect Supervision

To monitor the status of the FXO Answer and Disconnect Supervision feature, use the **show voice port** command, which causes the FXO voice port to be monitored. The following table illustrates the use of the **show voice port** command for monitoring voice port 1/1/0.

Command	Purpose
`Router# show voice port 1/1/0`	Shows a detailed status of the voice port. Under the heading "Voice card specific Info Follows:", the status of the FXO Answer and Disconnect Supervision feature is indicated by one of the following messages: "Answer Supervision is active" or "Answer Supervision is inactive".

Unbreakable Dial Tone

A common problem encountered in a VoIP network is being unable to break dial tone. The router puts a seizure on the local PBX but when digits are dialed, the dial tone stays. The calling party is unable to pass the DTMF tones or digits to the terminating device, resulting in callers being unable to dial the desired extension or interact with the device that needs DTMF tones, such as a voice mail or IVR application. Here are some possible causes of the problem:

- DTMF tones not being sent.
- DTMF tones not being understood.
- DTMF tones too distorted to be understood.
- Other signaling and cabling issues.

Make sure the dial type is set as DTMF on both the router and the PBX. The FXS port does not pass on the digits, therefore this setting is not available on an FXS port. However, this setting can be changed on FXO and E&M ports:

```
Router(config-voiceport)# dial-type ?
  dtmf   touch-tone dialer
  mf     mf-tone dialer
  pulse  pulse dialer
```

For more information, refer to *Inability To Break Dialtone in a Voice over IP Network*, document ID 22376.

Troubleshooting Caller ID Problems

Several debugs can be used to troubleshoot the Caller ID feature on the routers. The voice port module (VPM) signaling debugs, such as the **debug vpm signal** command, track the standard debugs with Caller ID feature turned on. These debugs are analyzed from the perspective of the terminating router and its FXO port; the caller ID is sent from this end. The following example shows an FXO port receiving caller ID. In this example, the phone sends the caller ID to the FXO port.

```
Nov 20 10:40:15.861 EST: [1/0/0] htsp_start_caller_id_rx
Nov 20 10:40:15.861 EST: [1/0/0] htsp_set_caller_id_rx:BELLCORE
Nov 20 10:40:15.861 EST: htsp_timer - 10000 msec
Nov 20 10:40:17.757 EST: [1/0/0, FXOLS_RINGING, E_DSP_SIG_0100]
Nov 20 10:40:17.757 EST: fxols_ringing_not
Nov 20 10:40:17.761 EST: htsp_timer_stop
```

```
Nov 20 10:40:17.761 EST: htsp_timer - 10000 msec
Nov 20 10:40:18.925 EST: [1/0/0] htsp_stop_caller_id_rx
Nov 20 10:40:21.857 EST: [1/0/0, FXOLS_RINGING, E_DSP_SIG_0000]
Nov 20 10:40:23.857 EST: [1/0/0, FXOLS_RINGING, E_DSP_SIG_0100]
Nov 20 10:40:23.857 EST: fxols_ringing_not
Nov 20 10:40:23.861 EST: htsp_timer_stop htsp_setup_ind
Nov 20 10:40:23.861 EST: [1/0/0] get_fxo_caller_id:Caller ID
received. Message type=128 length=31 checksum=74
Nov 20 10:40:23.861 EST: [1/0/0] Caller ID String 80 1C 01 08
31 31 32 30 31 35 34 30 02 07 35 35 35 31 32 31 32 07 07 4F 75
74 73 69 64 65 74
Nov 20 10:40:23.865 EST: [1/0/0] get_fxo_caller_id calling
num=5551212 calling name=Outside calling time=11/20 15:40
Nov 20 10:40:23.869 EST: [1/0/0, FXOLS_WAIT_SETUP_ACK,
E_HTSP_SETUP_ACK]
Nov 20 10:40:23.873 EST: fxols_wait_setup_ack:
Nov 20 10:40:23.873 EST: [1/0/0] set signal state = 0xC
timestamp = 0
Nov 20 10:40:23.985 EST: [1/0/0, FXOLS_PROCEEDING,
E_DSP_SIG_0100] fxols_proceed_clear
Nov 20 10:40:23.985 EST: htsp_timer_stop2
Nov 20 10:40:24.097 EST: [1/0/0, FXOLS_PROCEEDING,
E_DSP_SIG_0110] fxols_rvs_battery
Nov 20 10:40:24.097 EST: htsp_timer_stop2
Nov 20 10:40:24.733 EST: [1/0/0, FXOLS_PROCEED_RVS_BT,
E_HTSP_PROCEEDING] fxols_offhook_proc
Nov 20 10:40:24.733 EST: htsp_timer - 120000 msec
Nov 20 10:40:24.745 EST: [1/0/0, FXOLS_PROCEED_RVS_BT,
E_HTSP_VOICE_CUT_THROUGH] fxols_proc_voice
```

In this example, everything was working fine and both Name and Number Display were properly delivered to the phone. In the two scenarios below, the calling number is missing in one case, and the name display is missing in the other.

Calling Number Lost, Name Delivered

In the following example, the calling number is lost, but the name is delivered:

```
Nov 17 17:39:34.164 EST: [1/1/0] htsp_set_caller_id_tx
calling num=display_info=Outside called num=9913050
Nov 17 17:39:34.164 EST: [1/1/0] Caller ID String 80 16
01 08 31 31 31 37 32 32 33 39 04 01 4F 07 07 4F 75 74
73 69 64 65 88
```

In the Caller ID String, looking at "**04 01 4F**" translates to:

04 : Reason for Absence of DN
01 : Length of message
4F : "Out of Area"

Calling Number Delivered, Name Lost

In the following example, the calling number is delivered, but the name is lost:

```
Nov 17 17:53:24.034 EST: [1/1/0] htsp_set_caller_id_tx
calling num=5550109display_info= called num=5550011
Nov 17 17:53:24.034 EST: [1/1/0] Caller ID String 80
16 01 08 31 31 31 37 32 32 35 33 02 07 35 35 35 31 32
31 32 08 01 4F 05
```

In the Caller ID String, looking at "**08 01 4F**" translates to:

08 : Reason for Absence of Display
01 : Length
4F : Out of Area

For more information, refer to *Caller ID Name Delivery Issues on Cisco IOS Gateways*, document ID 23444.

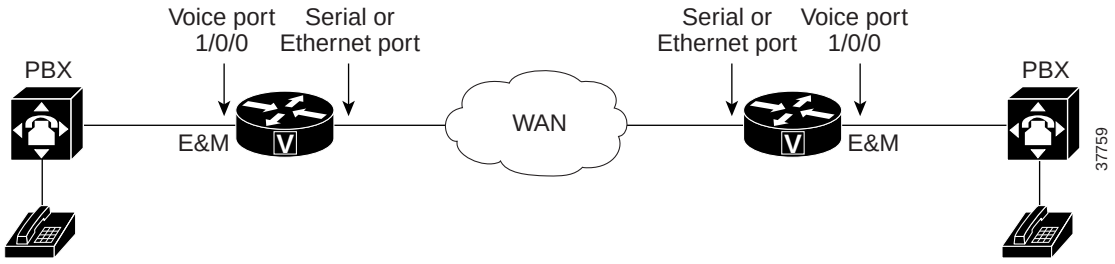
E&M Interfaces

The difference between a conventional two-wire telephone interface such as FXS or FXO and an E&M interface is that the E&M interface has wires that pass the audio signals plus wires to act as an input (to sense an incoming call) or an output (to indicate an outgoing call). These control leads are normally called the E lead (input) and the M lead (output). Depending on the type of E&M interface, the signaling leads could be controlled by connecting them to the ground, switching a –48-Vdc source, or completing a current loop between the two devices.

E&M interfaces can normally be two- or four-wire operation, which does not refer to the total number of physical connections on the port but rather to the way that audio is passed between the devices. Two-wire operation means the transmitting and receiving audio signals are passed through a single pair of wires (one pair equals two wires). Four-wire operation uses one pair for transmitting and another pair for receiving audio.

In Figure 18, two PBXs are connected across a WAN by E&M interfaces. This topology illustrates the path over a WAN between two geographically separated offices in the same company.

Figure 18 E&M Signaling Interfaces



This section contains the following topics:

- E&M Hardware Troubleshooting, page 14
- E&M Interface Types, page 17
- Troubleshooting E&M Interfaces at the Physical Level, page 26
- Confirming E&M Configuration, page 33
- Unbreakable Dial Tone, page 41

E&M Hardware Troubleshooting

The E&M interface typically connects remote calls from an IP network to a PBX. Troubleshoot Cisco E&M hardware by checking the following sections:

- Software Compatibility, page 15

Software Compatibility

For interface cards inserted into Cisco modular access routers, check the compatibility tables in the "Overview of Cisco Interface Cards" chapter in the *Cisco Interface Cards Installation Guide.*

Cabling

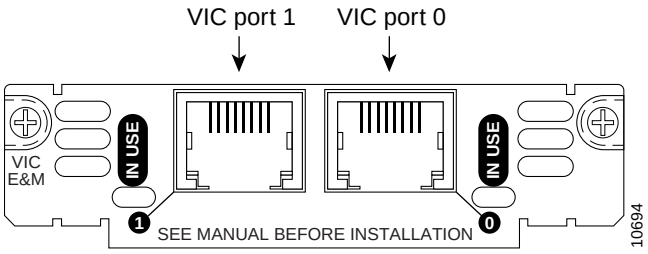
E&M is a signaling technique for two-wire and four-wire telephone and trunk interfaces. The E&M interface typically connects remote calls from an IP network to a PBX. The card is connected to the PSTN or PBX through a telephone wall outlet by a straight-through RJ-48C cable.

Note Refer to the appropriate platform product documentation for specific interface information about your E&M card.

The connector port for the E&M voice interface card is shown in Figure 19. Information about LEDs can be found in the "Connecting Voice Interface Cards to a Network" chapter of the *Cisco Interface Cards Installation Guide.*

Note Ports on the E&M voice interface card are color-coded brown.

Figure 19 Two-Port E&M Card Front Panel



To verify that the analog E&M hardware is being recognized by the Cisco IOS platform, use the following commands:

- **show version**—This command displays the configuration of the system hardware, the software version, the names of configuration files, and the boot images. See the following sample output.
- **show running-config**—This command shows the configuration of the Cisco platform. The voice ports should appear in the configuration automatically. See the following sample output.

show version Command on a Cisco 3640 Platform

```
Router# show version

Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3640-IS-M), Version 12.1(2), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2000 by cisco Systems, Inc.
Compiled Wed 10-May-00 07:20 by linda
Image text-base: 0x600088F0, data-base: 0x60E38000
```

```
ROM: System Bootstrap, Version 11.1(20)AA2, EARLY DEPLOYMENT RELEASE SOFTWARE(fc1)
Router uptime is 0 minutes
System returned to ROM by power-on at 11:16:21 cst Mon Mar 12 2001
System image file is "flash:c3640-is-mz.121-2.bin"
cisco 3640 (R4700) processor (revision 0x00) with 126976K/4096K bytes of memory.
Processor board ID 16187704
R4700 CPU at 100Mhz, Implementation 33, Rev 1.0
Bridging software.
X.25 software, Version 3.0.0.
SuperLAT software (copyright 1990 by Meridian Technology Corp).
2 Ethernet/IEEE 802.3 interface(s)
2 Voice FXS interface(s)
2 Voice E&M interface(s)
DRAM configuration is 64 bits wide with parity disabled.
125K bytes of non-volatile configuration memory.
32768K bytes of processor board System flash (Read/Write)
20480K bytes of processor board PCMCIA Slot0 flash (Read/Write)
Configuration register is 0x2102
```

show running-config Command on a Cisco 3640 Platform

```
Router# show running-config

Building configuration...
Current configuration:
!
!--- Some output omitted.
version 12.1
service timestamps debug uptime
service timestamps log uptime
!
hostname Router
!
voice-port 3/0/0
!
voice-port 3/0/1
!
voice-port 3/1/0
!
voice-port 3/1/1
!
end
```

Shutdown Port

Check to make sure the port is not shut down. Enter the **show voice port** command with the voice port number that you are troubleshooting. The output will tell you:

- If the voice port is up. If it is not, use the **no shutdown** command to make it active.
- What parameter values have been set for the voice port, including default values (default values do not appear in the output from **the show running-config command**). If these values do not match those of the telephony connection you are making, reconfigure the voice port.

E&M Interface Types

This section describes the standard analog E&M interface types I, II, III, and V (IV is not supported by Cisco platforms). The following topics are covered:

E&M Signaling Unit Side and Trunk Circuit Side Compatibility Issues

E&M signaling defines a trunk circuit side and a signaling unit side for each connection. Cisco's analog E&M interface functions as the signaling unit side, so it expects the other side to be a trunk circuit. When you use E&M interface model Type II or Type V, you can connect two signaling unit sides back to back by appropriate crossing of the signaling leads. When using the E&M Type I or Type III interface, you cannot connect two signaling unit sides back to back.

Many PBX brands have E&M analog trunk cards that can operate as either the trunk circuit side or the signaling unit side. Because the Cisco E&M interfaces are fixed as the signaling unit side of the interface, it may be necessary to change the E&M trunk settings on the PBX to operate as the trunk circuit side. If Type I or III E&M is being used, this is the only way the PBX can work with the Cisco E&M interface.

Some PBX products (and many key systems) can operate only as the signaling unit side of the E&M interface. They cannot interoperate with the Cisco E&M interface if Type I or Type III is chosen. If Type II or Type V E&M is being used, PBX products fixed as “signaling unit” side can still be used with the Cisco E&M interface via Type II or Type V.

Each E&M signaling type has a unique circuit model and connection diagram. The following sections describe the different types. Table 25 shows the E&M supervisory signal description.

Table 25 E&M Interface Supervision Signal Description

Signal	Meaning	Description
E	Ear or earth	Signal wire from trunking (CO) side to signaling side.
M	Mouth or magnet	Signal wire from signaling side to trunking (CO) side.
SG	Signal ground	Used on E&M Types II, III, and IV. (Type IV is not supported on Cisco gateways.)
SB	Signal battery	Used on E&M Types II, III, and IV. (Type IV is not supported on Cisco gateways.)
T/R	Tip/Ring	Tip and ring leads carry audio between the signaling unit and the trunking circuit. On a two-wire audio operation circuit, this pair carries the full-duplex audio path.
T1/R1	Tip-1/Ring-1	Used on four-wire audio operation circuits only. The four-wire implementation provides separate paths for receiving and sending audio signals.

E&M Type I Interface Model

E&M Type I is the original E&M lead signaling arrangement, and it is the most common interface type in North America. Table 26 shows the sent signal states for on- and off-hook signaling.

Table 26 E&M Type I Signal States

PBX to Cisco Gateway			Cisco Gateway to PBX		
Lead	**On-Hook**	**Off-Hook**	**Lead**	**On-Hook**	**Off-Hook**
M	Ground	Battery	E	Open	Ground

The gateway grounds its E-lead to signal a trunk seizure. The PBX applies battery to its M-lead to signal a seizure. Cisco gateways expect to see off-hook conditions on the M-lead, and they signal off-hook to a remote device on the E-lead. E&M Type I 2-wire operation is shown in Figure 20. E&M Type I 4-wire operation is shown in Figure 21.

Figure 20 E&M Type I 2-Wire Audio Operation

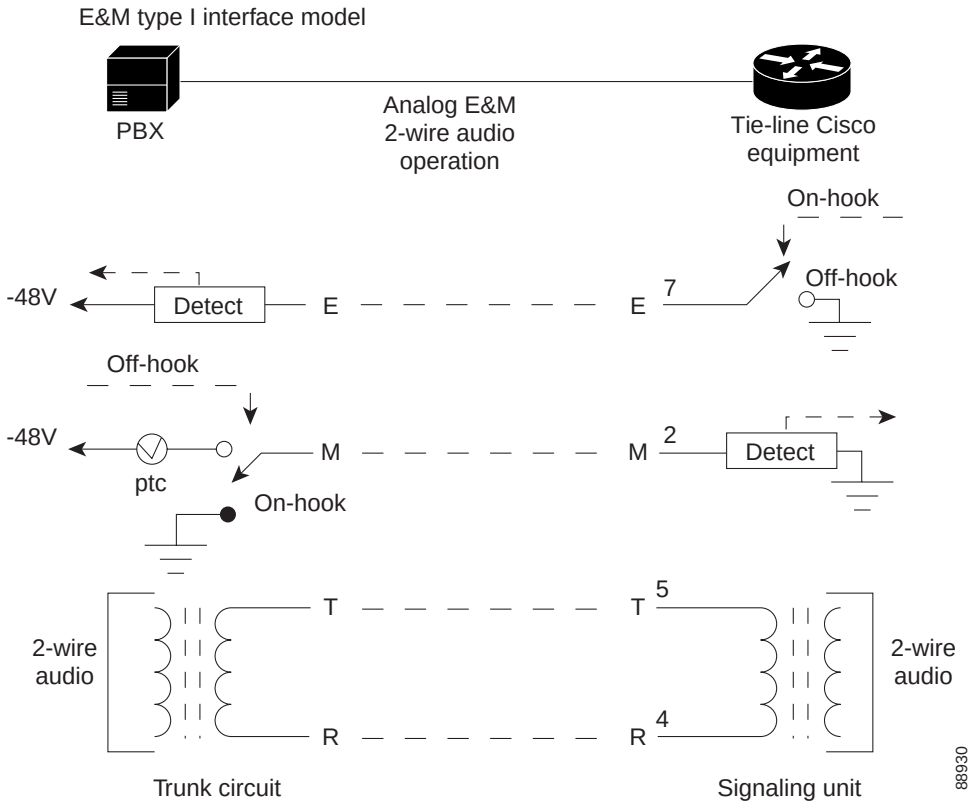
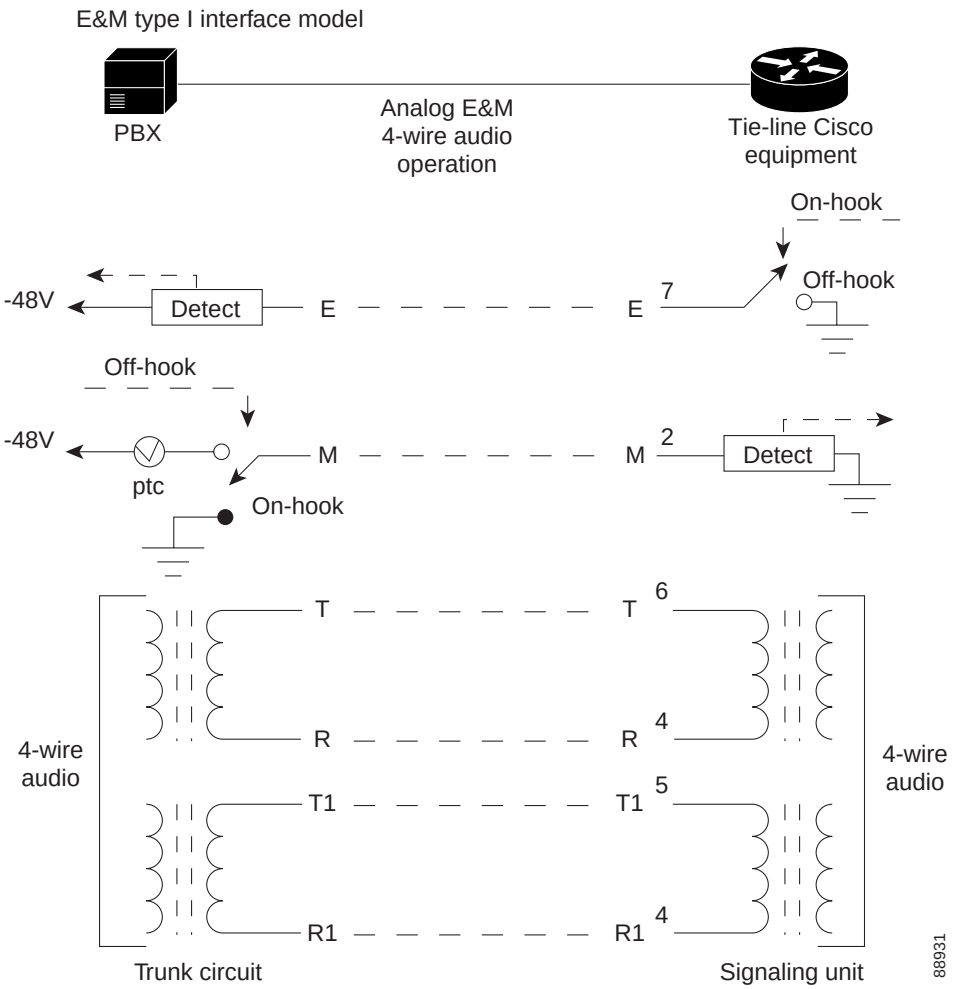
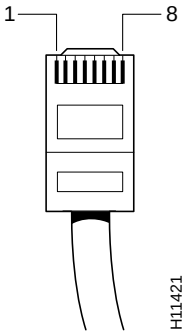


Figure 21 E&M Type I 4-Wire Audio Operation



Note For the four-wire audio setup, Pin 6 (Tip) and Pin 3 (Ring) on the router transport the audio path from the PBX to the router. Pin 5 (Tip1) and 4 (Ring1) on the router transport the audio path from the router to the PBX. Pins for the cable are shown in Figure 22.

Figure 22 E&M Cabling Pins



Considerations for Type I interfaces include:

- Two signaling units cannot be connected back to back.
- A Type I signaling unit and a trunk circuit share a common ground.
- Type I does not provide isolation between trunk circuits and signaling units, can produce noise in audio circuits, and might be susceptible to electrical transients.
- It is critical to provide and ground connection directly between the Cisco product and the PBX. Otherwise, E&M signaling might be intermittent.
- Four wires are used for Type I, two-wire audio operation.
- Six wires are used for Type I, four-wire audio operation.

E&M Type II Interface Model

E&M Type II provides a 4-wire fully looped arrangement that provides full isolation between the trunks and signaling units. Type II is usually used on Centrex lines and Nortel PBX systems. Table 27 shows the sent signal states for on- and off-hook signaling.

Table 27 E&M Type II Signal States

PBX to Cisco Gateway			Cisco Gateway to PBX		
Lead	**On-Hook**	**Off-Hook**	**Lead**	**On-Hook**	**Off-Hook**
M	Open	Battery	E	Open	Ground

The gateway grounds its E-lead to signal a trunk seizure. The PBX applies battery to its M-lead to signal a seizure. Cisco gateways expect to see off-hook conditions on the M-lead, and they signal off-hook to a remote device on the E-lead. E&M Type II 2-wire operation is shown in Figure 23. E&M Type II 4-wire operation is shown in Figure 24.

Figure 23 E&M Type II 2-Wire Audio Operation

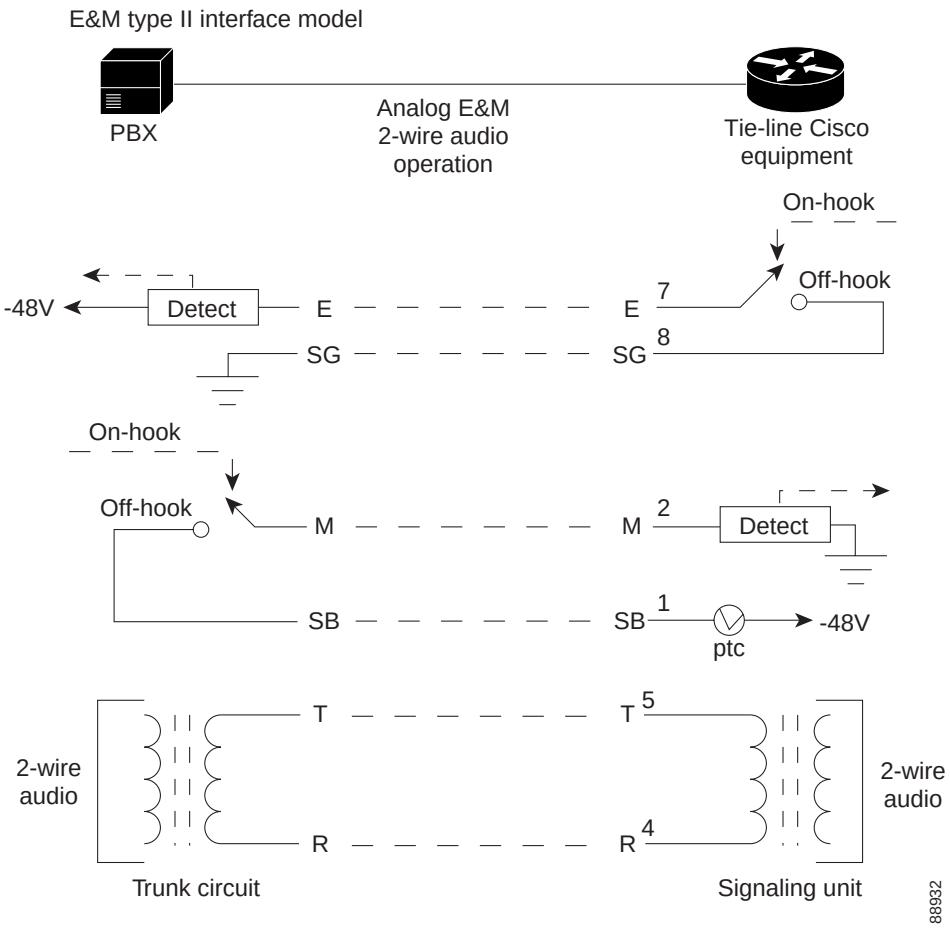
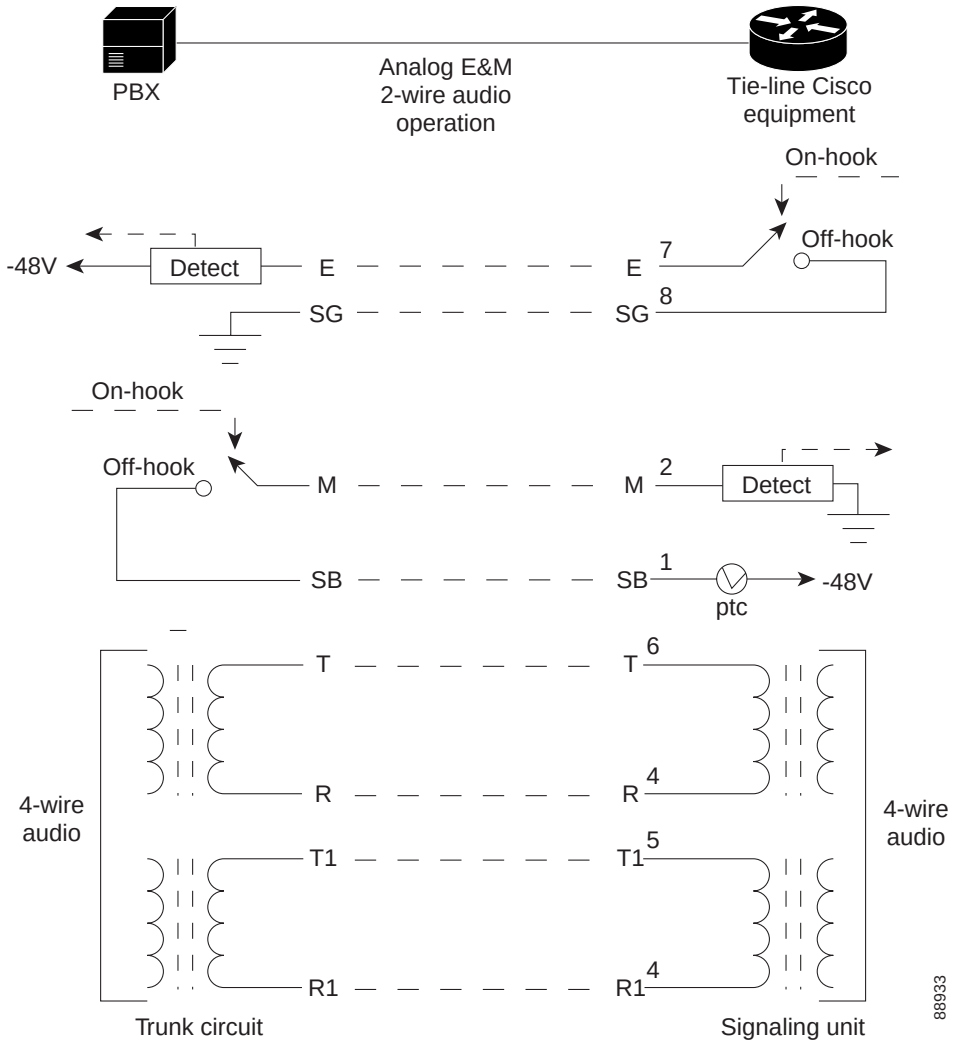


Figure 24 E&M Type II 4-Wire Audio Operation



Note For the four-wire audio setup, Pin 6 (Tip) and Pin 3 (Ring) on the router transport the audio path from the PBX to the router. Pin 5 (Tip1) and Pin 4 (Ring1) on the router transport the audio path from the router to the PBX.

Considerations for Type II interfaces include:

- Two signaling unit sides can be connected back-to-back if the appropriate signaling leads are swapped.
- Six wires are used for Type II, two-wire audio operation.
- Eight wires are used for Type II, four-wire audio operation.

E&M Type III Interface Model

E&M Type III is a partially looped four-wire E&M arrangement with ground isolation. The signaling unit provides both the battery and the ground. Table 28 shows the sent signal states for on- and off-hook signaling.

Table 28 E&M Type III Signal States

PBX to Cisco Gateway			Cisco Gateway to PBX		
Lead	**On-Hook**	**Off-Hook**	**Lead**	**On-Hook**	**Off-Hook**
M	Ground	Battery	E	Open	Ground

The router senses loop current on the M-lead for an inbound seizure and grounds its E-lead for an outbound seizure. Cisco routers/gateways expect to see off-hook conditions on the M-lead, and they signal off-hook to a remote device on the E-lead. E&M Type III 2-wire operation is shown in Figure 25. E&M Type III 4-wire operation is shown in Figure 26.

Figure 25 E&M Type III 2-Wire Audio Operation

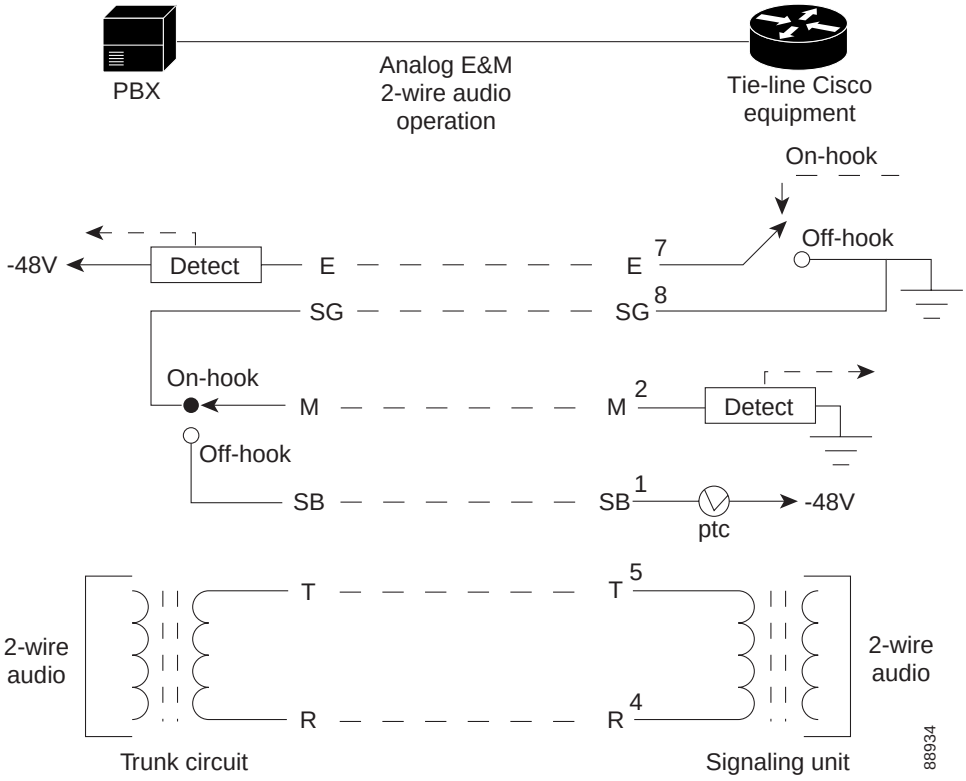
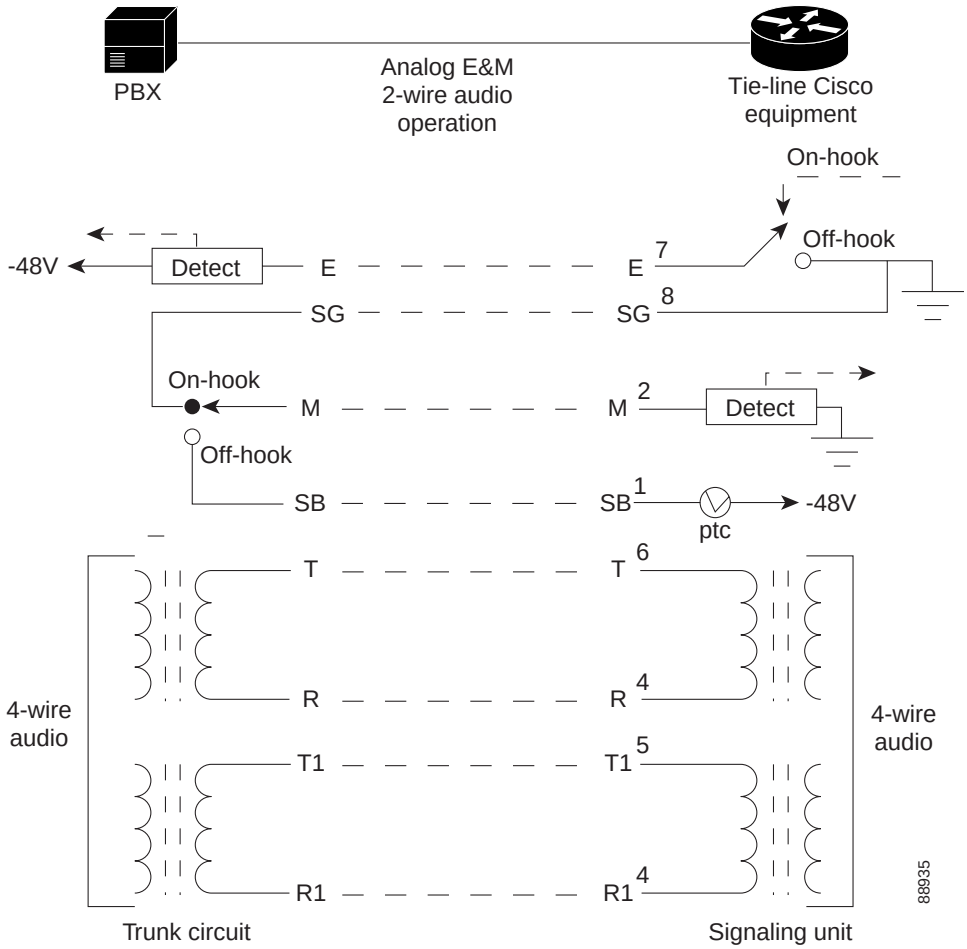


Figure 26 E&M Type III 4-Wire Audio Operation



Note For the four-wire audio setup, Pin 6 (Tip) and Pin 3 (Ring) on the router transport the audio path from the PBX to the router. Pin 5 (Tip1) and Pin 4 (Ring1) on the router transport the audio path from the router to the PBX.

Considerations for Type III interfaces include:

- Two signaling units cannot be connected back to back.
- Six wires are used for Type III, two-wire audio operation.
- Eight wires are used for Type III, four-wire audio operation.

E&M Type V Interface Model

E&M Type V is widely used outside North America (nearly a worldwide standard.) Type V is a symmetrical two-wire lead arrangement that signals in both directions (open for on-hook and ground for off-hook.) Table 29 shows the sent signal states for on- and off-hook signaling.

Table 29 E&M Type V Signal States

PBX to Cisco Gateway			Cisco Gateway to PBX		
Lead	**On-Hook**	**Off-Hook**	**Lead**	**On-Hook**	**Off-Hook**
M	Open	Ground	E	Open	Ground

The gateway grounds its E-lead to signal a trunk seizure. The PBX grounds its M-lead to signal a seizure. Cisco gateways expect to see off-hook conditions on the M-lead, and they signal off-hook to remote device on the E-lead. E&M Type V 2-wire operation is shown in Figure 27. E&M Type V 4-wire operation is shown in Figure 28.

Figure 27 E&M Type V 2-Wire Audio Operation

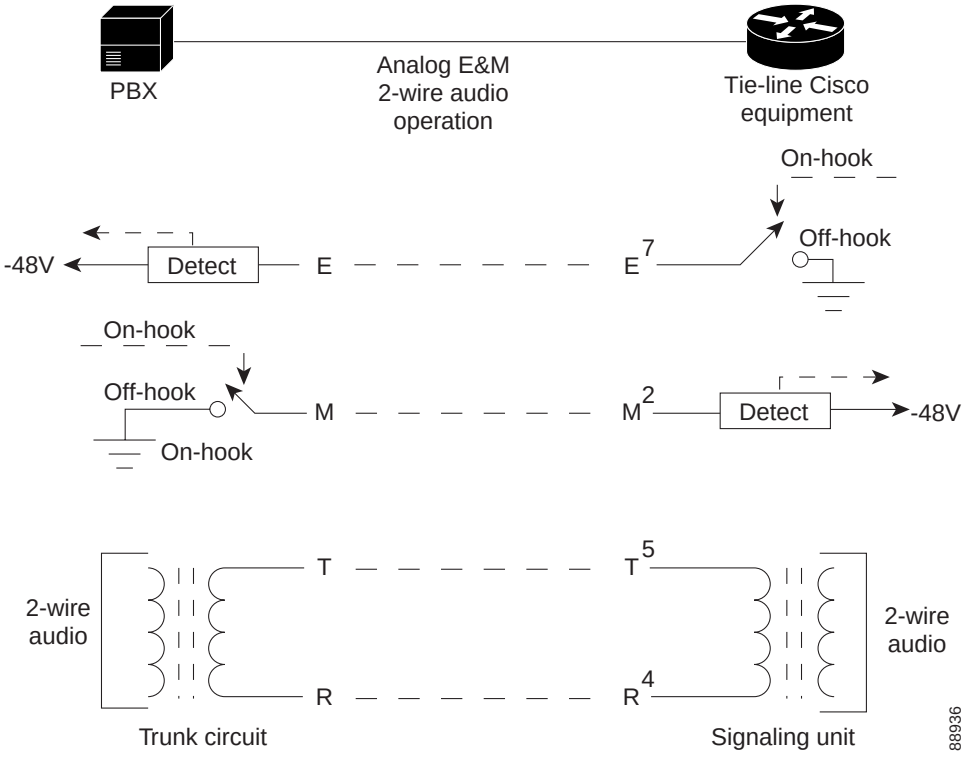
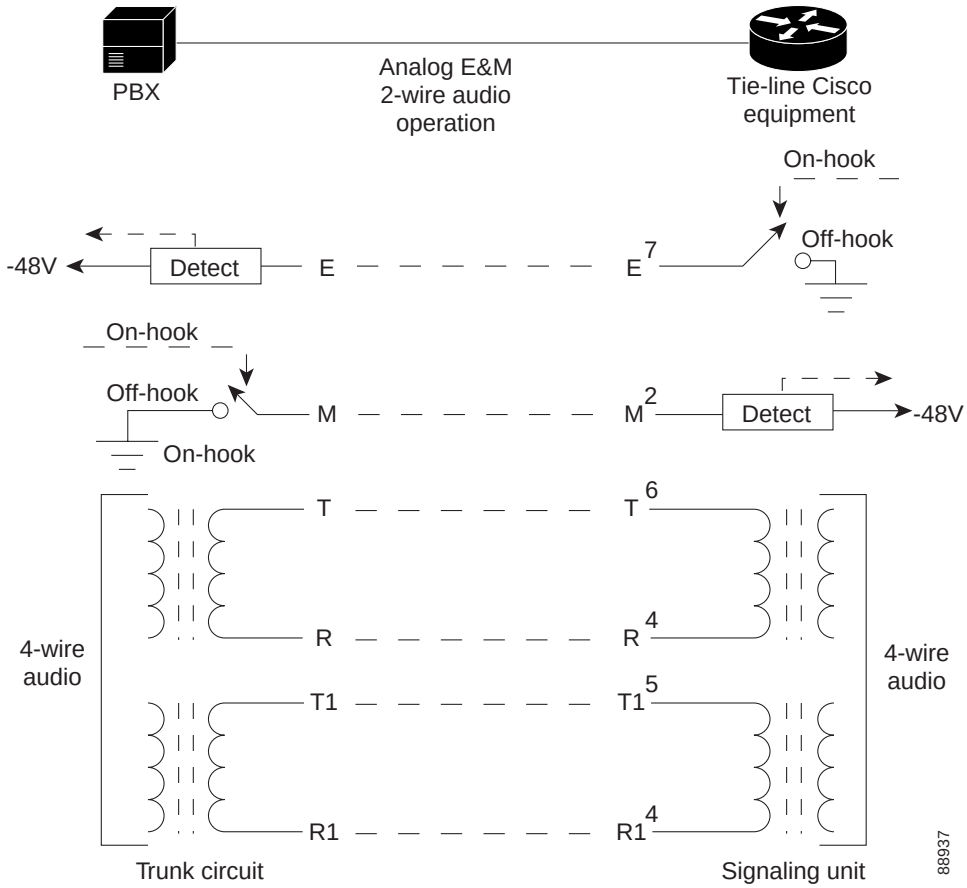


Figure 28 E&M Type V 4-Wire Audio Operation



Note For the four-wire audio setup, Pin 6 (Tip) and Pin 3 (Ring) on the router transport the audio path from the PBX to the router. Pin 5 (Tip1) and Pin 4 (Ring1) on the router transport the audio path from the router to the PBX.

Considerations for Type V interfaces include:

- Type V does not provide ground isolation.
- Two signaling unit sides can be connected back-to-back if the appropriate signaling leads are swapped.
- Four wires are used for Type V, two-wire audio operation.
- Six wires are used for Type V, four-wire audio operation.

Troubleshooting E&M Interfaces at the Physical Level

E&M provides the highest quality analog interface available, but it also is the most difficult to administer due to the number of leads, configurations, and protocol issues. Usually it is helpful to have the appropriate reference diagram available when verifying the connections.

Preparing to Troubleshoot E&M Physical Problems

Use the information in the following sections to prepare to troubleshoot E&M physical problems:

Hardware Troubleshooting Tools

Test equipment is not required for every installation, but sometimes you need to use it to isolate problems with analog E&M ports. The most useful equipment is a digital multimeter and a technician's line test set. These tools allow measurement of signaling states and voltages, and monitoring of audio signals. A digital multimeter is used to measure the DC loop voltage and AC ringing voltage on FXS ports, E- or M-lead signaling transitions, voltages on E or M leads, and DC resistance of E&M signaling leads.

In the terminating mode of operation, the technician's line test set acts like a normal telephone handset when connected to a loopstart trunk, allowing telephone numbers to be dialed on the built-in keypad. When switched to the monitoring mode (bridging mode), the unit presents a high impedance to the TX or RX audio pairs of the E&M port, allowing the audio signals and tones to be heard on the built-in loudspeaker. This mode helps you find problems with one-way audio, incorrect digits being sent or received, distortion and level problems, and possible sources of noise and echo.

For an effective troubleshooting kit, have the following items available:

- Digital volt ohm meter (VOM) with sharp-tipped probes. Those with the analog bar graph and a beeper with pitch proportional to the display are particularly useful.
- Lineman's test set.
- RJ-45 breakout adapter. This adapter has an RJ-45 socket on each end, with terminals for each of the lines distributed about each side.
- RJ-45 straight-through cable (verify that it is straight through).
- Alligator-clip patch cables.

Precautions

Warning **Equipment closets where telecommunication devices exist, while usually not hazardous, can have some potentially harmful situations, including, but not limited to:**

- **Lead acid battery stacks** able to supply large amounts of current, and possibly flammable hydrogen fumes. Ventilation and insulation are the keys to avoiding damage. Wear long-sleeved shirts, long pants, and steel-toed work boots. Keep electrically insulated work gloves and OSHA-approved eye protection available. Avoid wearing metal objects such as chains, bracelets, rings, and watches unless under cover and away from making any connection. Voltage does not injure; current does.
- **Many wires** for voice, data, power, and so on. Watch for potentially damaging outages caused by pulling a wire that is snagged on another wire. RJ plugs have a tendency to snag on other wires and loosen equipment.
- **Sharp edges**. Equipment deployed before there were safety requirements regarding snag or cut hazards often have protruding bolts and screws. Full clothing protection helps protect you in these cases.

- **Loose, heavy equipment**. Objects in the equipment room may be less than secure. These objects can fall and hurt the equipment, you, or others. If moving heavy objects is involved, leave it to the facility staff or other professional movers; otherwise, use a back protector belt and follow proper OSHA-approved lifting and moving guidelines.

PBX Interconnection

The majority of PBXs interface with peripheral equipment using cable distribution frames (DFs). Multipair cables are run from the PBX equipment cabinet to the distribution frame, where they are jumpered (cross-connected) to the external devices. These DFs have various names, but the most common terms for them are 110 block, 66 block, and Krone frame. The DF is generally the place where all connections are made between the router voice port and the PBX, so it is where most wiring errors are made and would obviously be the best place to perform testing and troubleshooting.

Use Rollover Cable for E&M Port-to-Port Testing

Past experience of Cisco Technical Assistance Center (TAC) engineers has shown that most E&M-related faults are due to incorrect wiring or PBX port programming. To assist in determining if the fault is external to the router, you can use the standard rollover console cable that is supplied with every Cisco router as an E&M cross-over cable. This cross-over cable connects the signaling output of one port to the input of the other port and maintains an audio path between the two ports. You can configure a dial peer so that a test call is sent out one port and looped back into the second port, proving the operation of the router.

The rollover console cable has the following RJ-45 connector wiring:

1-------8

2-------7

3-------6

4-------5

5-------4

6-------3

7-------2

8-------1

The signaling cross-over occurs as pins 2 (M-lead) and 7 (E-lead) on one port are connected to pins 7 (E-lead) and 2 (M-lead) on the other port. The two ports share a common internal ground. The cross-over on pins 4 and 5 (audio pair) has no effect on the audio signal. By setting both voice ports to 2-wire, type 5 operation, the E&M ports become symmetrical and an outward seizure on one port is seen as an incoming seizure on the second port. Any DTMF digits sent out immediately come back in and are then matched on another dial peer. If the test calls are successful, there is little doubt about the operation of the router voice ports. In the following example, the assumption is made that there are working devices on the IP network that can originate and accept VoIP calls.

The voice ports and dial peers are configured like this:

```
voice-port 1/0/0
!--- First port under test.
operation 2-wire
signal-type wink
type 5
!
voice-port 1/0/1
```

```
!--- Second port under test.
operation 2-wire
signal-type wink
type 5
Cisco - Analog E&M Troubleshooting Guidelines (Cisco IOS Platforms)
!
dial-peer voice 100 pots
!--- Send call out to port 1/0/0, strip the 100 and prefix with a called
!--- number 200.
destination-pattern 100
port 1/0/0
prefix 200
!
dial-peer voice 200 voip
!--- Incoming test call for 200 comes
!--- in on port 1/0/1 and is sent to 10.1.1.1 as VoIP call.
destination-pattern 200
session-target ipv4:10.1.1.1
!
```

When a VoIP call comes in to the router with a called number of 100, it is sent out port 1/0/0. By default, any explicitly matched digits on a POTS dial peer are assumed to be an access code and stripped off before the call is made. To route the call correctly, these digits need to be replaced. In this case the **prefix** command prepends the digits 200 as the called number. This call is immediately looped back in on port 1/0/1. The digits match on dial-peer 200 and make the new call to the designated IP address. The devices originating and accepting the VoIP calls should then have an audio connection that is across the IP network and goes out and back through the E&M ports. This connection proves the router is working properly and indicates that the fault is external to the router. The majority of faults are due to incorrect cabling or PBX port programming issues.

Troubleshooting Type I Interfaces

The four-wire Type I interface from the PBX (set up for the trunk circuit side) has the following characteristics:

- E detector “floats” at –48 V below ground.
- M contact has low ohms to ground on-hook, and is –48 V below ground when off-hook.
- Resistance is approximately 30 to 150 ohms between tip and ring, sometimes in series with 2.2 uF of capacitance.
- Resistance is approximately 30 to 150 ohms between tip-1 and ring-1, sometimes in series with 2.2 uF of capacitance.

Confirm the Cable Interface from the PBX

If you think the cable is bad, pull the suspect voice cable from the router and leave the other side connected to the PBX. Then do the following:

- With a VOM, measure DC voltage between pin 7 of the cable and the chassis ground. The meter should read between –24 V and –56 V. If it does not, pin 7 is likely not the E lead on the PBX.
- Measure the other pins, looking for –24 to –56 V to ground. Some devices, like an AT&T, Lucent or Avaya PBX, bias the tip/ring leads to –48 V to aid debugging. On pins that had no conclusive energy, measure the ohms to ground with a VOM. If one shows less than 500 ohms, it is likely the M lead. It should be pin 2 on the cable. If pin 2 shows between -–24 v and –48 V to ground, it is possible that the PBX is off hook; sometimes a PBX busies out what seems to be a bad port.

- With a VOM, measure the resistance (ohms) between tip and ring. It should read from 30 to 120 ohms if the PBX has no DC blocking capacitor. If there is a capacitor, you will see the meter jump to around 100 ohms, then climb to infinity as the capacitor charges. With either signature, there is an audio pair—you just need to figure out which direction it is.
- Do the same for tip-1/ring-1. It should behave like tip/ring.
- Attach a test to tip/ring. While listening, ground E (pin 7 on the cable). If the PBX is configured to provide a dial tone, you should hear it in the earpiece. If you hear nothing, try the other audio pair in case it is cross-wired. If you still hear nothing, the PBX might not give a dial tone on a trunk line.
- It is acceptable to cross tip with ring or tip-1 with ring-1.

Additional Troubleshooting Tips

- On either the router or the PBX, try a similar port that is known to work.
- Listen in on both sides of the audio path (one at a time) with the test set to hear the call progress.
- Try to spoof the signaling of one end or the other by clipping one of the active signals to see if the equipment reacts as expected. Grounding E should simulate an inbound call coming over the trunk to the PBX, and the PBX might respond with a dial tone (if provisioned to do so).
- Using an extension off of the PBX, try to seize the trunk and see if M connects to ground.

Troubleshooting Type II Interfaces

The four-wire Type-II interface from the PBX (setup for trunk circuit side) has the following characteristics:

- E lead detector “floats” at -48 v below ground.
- SG lead has a low ohms to ground.
- M lead contact between M and SB is open when on-hook and closed when off-hook.
- M lead floats.
- SB lead floats.
- Approximately 30 to 150 ohms between tip and ring, sometimes in series with 2.2 uF of capacitance.
- Approximately 30 to 150 ohms between tip-1 and ring-1, sometimes in series with 2.2 uF of capacitance.

Confirm the Cable Interface from the PBX

Pull the suspect voice cable from the router and leave the other side connected to the PBX. Then do the following:

- With a VOM, measure the DC voltage between E (pin 7 of the cable) and the chassis ground. The meter should read between –24 V and –56 V. If it does not, pin 7 on the cable is likely not the E lead.
- Measure the other pins, looking for –24 to –56 V to ground. Some devices, like an AT&T, Lucent, or Avaya PBX, bias the tip/ring leads to –48 V to aid debugging. On pins that have no conclusive energy, measure the ohms to ground with a VOM. If one shows less than 500 ohms, it is likely the SG lead. It should be pin 8 on the cable.
- With a VOM, measure the resistance (ohms) between tip and ring. It should read from 30 to 120 ohms if the PBX has no DC blocking capacitor. If there is a capacitor, you will see the meter jump to around 100 ohms, then climb to infinity as the capacitor charges. With either signature, there is an audio pair—you just need to figure out which direction it is.

- Do the same for tip-1/ring-1. It should behave like tip/ring.
- Attach a test set to tip/ring. While listening, ground E (pin 7 on the cable). If the PBX is configured to provide a dial tone, you should hear it in the earpiece. If you hear nothing, try the other audio pair in case it is cross-wired. If you still hear nothing, the PBX might not give a dial tone on a trunk line.
- It is acceptable to cross tip with ring or tip-1 with ring-1.
- In most cases, you can get M/SB backwards and E/SG backwards and still have no problems.

Additional Troubleshooting Tips

- On either the router or the PBX, try a similar port that is known to work.
- Listen in on both sides of the audio path (one at a time) with the test set to hear the call progress.
- Try to spoof the signaling of one end or the other by clipping one of the active signals to see if the equipment reacts as expected. Grounding E should simulate an inbound call coming over the trunk to the PBX, and the PBX might respond with a dial tone (if provisioned to do so).
- Using an extension off of the PBX, try to seize the trunk and see if M connects to ground.

Troubleshooting Type III Interfaces

The four-wire Type-III interface from the PBX has the following characteristics:

- E lead detector “floats” at –48 V below ground.
- M lead contact between M and SG when on-hook, and between M and SB when off-hook.
- SG lead floats.
- M lead floats.
- SB lead floats.
- Approximately 30 to 150 ohms between tip and ring, sometimes in series with 2.2 uF of capacitance.
- Approximately 30 to 150 ohms between tip-1 and ring-1, sometimes in series with 2.2 uF of capacitance.

Confirm the Cable Interface from the PBX

Pull the suspect voice cable from the router and leave the other side connected to the PBX. Then do the following:

- With a VOM, measure DC voltage between E (pin 7 of the cable) and the chassis ground. The meter should read somewhere between –24 V and –56 V. If it does not, pin 7 is likely not the E lead.
- Measure the other pins, looking for –24 to –56 V to ground. Some PBXs bias (apply a DC voltage to control the operation of a device) the tip/ring leads to –48 V to aid debugging. On pins that have no conclusive energy:
  - Look for a contact closure (low ohms) between M and SG (if the PBX is on-hook).
  - Look for a contact closure (low ohms) between M and SB (if the PBX is off-hook).
- With a VOM, measure the resistance (ohms) between tip and ring. It should read from 30 to 120 ohms if the PBX has no DC blocking capacitor. If there is a capacitor, you'll see the meter jump to around 100 ohms, then climb to infinity as the capacitor charges. With either signature, there is an audio pair--you just need to figure out which direction it is.
- Do the same for tip-1/ring-1. It should behave like tip/ring.

- Attach a test set to tip/ring. While listening, ground E (pin 7 on the cable). If the PBX is configured to provide a dial tone, you should hear it in the earpiece. If you hear nothing, try the other audio pair in case it is cross-wired. If you still hear nothing, the PBX might not give a dial tone on a trunk line.
- It is acceptable to cross tip with ring or tip-1 with ring-1.

Additional Troubleshooting Tips

- On either the router or the PBX, try a similar port that is known to work.
- Listen in on both sides of the audio path (one at a time) with the test set to hear the call progress.
- Try to spoof the signaling of one end or the other by clipping one of the active signals to see if the equipment reacts as expected. Grounding E should simulate an inbound call coming over the trunk to the PBX, and the PBX might respond with a dial tone (if provisioned to do so).
- Using an extension off of the PBX, try to seize the trunk and see if M (pin 2 on the cable) connects to SB (pin 1 on the cable).

Troubleshooting Type V Interfaces

The four-wire Type-V interface from the PBX has the following characteristics:

- E lead detector “floats” at –48 V below ground.
- M lead contact ground is open when on-hook, and closed when off-hook.
- Approximately 30 to 150 ohms between tip and ring, sometimes in series with 2.2 uF of capacitance.
- Approximately 30 to 150 ohms between tip-1 and ring-1, sometimes in series with 2.2 uF of capacitance.

Confirm the Cable Interface from the PBX

Pull the suspect voice cable from the router and leave the other side connected to the PBX. Then do the following:

- With a VOM, measure DC voltage between E (pin 7 of the cable) and the chassis ground. The meter should read between –24 V and –56 V. If it does not, pin 7 on the cable is likely not the E lead.
- With a VOM, measure the resistance (ohms) between tip and ring. It should read from 30 to 120 ohms if the PBX has no DC blocking capacitor. If there is a capacitor, you will see the meter jump to around 100 ohms, then climb to infinity as the capacitor charges. With either signature, there is an audio pair—you just need to figure out which direction it is.
- Do the same for tip-1/ring-1. It should behave like tip/ring.
- Attach a test set to tip/ring. While listening, ground E (pin 7 on the cable). If the PBX is configured to provide a dial tone, you should hear it in the earpiece. If you hear nothing, try the other audio pair in case it is cross-wired. If you still hear nothing, the PBX might not give a dial tone on a trunk line.
- It is acceptable to cross tip with ring or tip-1 with ring-1.

Additional Troubleshooting Tips

- On either the router or the PBX, try a similar port that is known to work.
- Listen in on both sides of the audio path (one at a time) with the test set to hear the call progress.

- Try to spoof the signaling of one end or the other by clipping one of the active signals to see if the equipment reacts as expected. Grounding E should simulate an inbound call coming over the trunk to the PBX, and the PBX might respond with a dial tone (if provisioned to do so).
- Using an extension off of the PBX, try to seize the trunk and see if M (pin 2 on the cable) connects to ground.

Confirming E&M Configuration

The following items should be checked to confirm the E&M configuration:

- Confirming the PBX E&M Configuration Parameters, page 33
- Confirming the Cisco IOS Gateway Configuration, page 33
- Verifying the Wiring Arrangement Between the PBX and the Cisco Gateway, page 34
- Verifying Supervision Signaling, page 35
- Verifying That the Cisco Equipment and PBX Are Sending and Receiving Digits, page 36
- Verifyinng That the Gateway Sends the Expected Digits to the PBX, page 39
- Verify That the Gateway Receives the Expected Digits from the PBX, page 40

Confirming the PBX E&M Configuration Parameters

The Cisco gateway needs to match the PBX configuration. One of the challenges of configuring and troubleshooting analog E&M circuits are the amount of configuration variables.

- E&M signaling type (I, II, III, V)
- Audio implementation (2-wire / 4-wire)
- Start dial supervision (wink-start, immediate, delay-dial)
- Dial method (DTMF, pulse)
- Call progress tones (standardized within geographic regions)
- PBX port impedance

For information about how specific PBX types interoperate with your gateway, go to the PBX interoperability portal, which is located here:

http://www.cisco.com/warp/public/779/largeent/avvid/inter_operability/

Note E&M Type IV is not supported by Cisco gateways. E&M Type V is the most common interface type used outside of North America, but the term Type V is not commonly used outside of North America. From the viewpoint of many PBX operators, there is only one E&M type, what is called Type V in North America.

Confirming the Cisco IOS Gateway Configuration

The Cisco gateway configuration should match the connected PBX configuration. Use the following commands to verify the Cisco IOS platform configuration:

- **show running-config—**This command displays the running configuration of the router/ gateway.

Note The default configuration on E&M voice ports is Type I, wink-start, 2-wire operation, DTMF dialing. Default E&M voice port parameters are not displayed with the **show running-config** command.

- **show voice-port—** For E&M voice ports, this command displays specific configuration data such as E&M voice port, interface type, impedance, dial-supervision signal, audio operation, and dial method. For detailed information see the sample output below.

Sample Output of show voice port Command

```
Router# show voice port 1/0/0
recEive And transMit 1/0/0 Slot is 1, Sub-unit is 0, Port is 0
Type of VoicePort is E&M
Operation State is DORMANT
Administrative State is UP
The Last Interface Down Failure Cause is Administrative Shutdown
Description is not set
Noise Regeneration is enabled
Non Linear Processing is enabled
Music On Hold Threshold is Set to -38 dBm
In Gain is Set to 0 dB
Out Attenuation is Set to 0 dB
Echo Cancellation is enabled
Echo Cancel Coverage is set to 8 ms
Connection Mode is normal
Connection Number is not set
Initial Time Out is set to 10 s
Interdigit Time Out is set to 10 s
Call-Disconnect Time Out is set to 60 s
Region Tone is set for US
Analog Info Follows:
Currently processing none
Maintenance Mode Set to None (not in mtc mode)
Number of signaling protocol errors are 0
Impedance is set to 600r Ohm
Voice card specific Info Follows:
Signal Type is immediate
Operation Type is 2-wire
E&M Type is 5
Dial Type is dtmf
In Seizure is inactive
Out Seizure is inactive
Digit Duration Timing is set to 100 ms
InterDigit Duration Timing is set to 100 ms
Pulse Rate Timing is set to 10 pulses/second
InterDigit Pulse Duration Timing is set to 500 ms
Clear Wait Duration Timing is set to 400 ms
Wink Wait Duration Timing is set to 200 ms
Wink Duration Timing is set to 200 ms
Delay Start Timing is set to 300 ms
Delay Duration Timing is set to 2000 ms
Dial Pulse Min. Delay is set to 140 ms
```

Verifying the Wiring Arrangement Between the PBX and the Cisco Gateway

Physical wiring is often the primary source for analog E&M problems. It is imperative that you verify that the cable/wiring you are using is appropriate for the E&M setup in place. A few things to consider:

- **E&M Type I and Type V use two leads for supervisory signaling (on/off hook signaling)**—E (ear, earth) and M (mouth, magnet). Cisco routers/gateways expect to see off-hook conditions on the M-lead and signal off-hook to the remote device on the E-lead.
- **E&M Type II and Type III use four leads for supervisory signaling (on/off hook signaling)**—E (ear, earth), M (mouth, magnet), SG (signal ground), SB (signal battery). Cisco routers/gateways expect to see off-hook conditions on the M-lead and signal off-hook to a remote device on the E-lead.
- **Audio operation**—The 2-wire/4-wire operation is independent of the signaling type. For example, a 4-wire audio operation E&M circuit has 6 physical wires if configured for Type I or Type V and 8 physical wires if configured for Type II or Type III.
- **Audio path wiring**—In 4-wire audio mode, some PBX es and key systems reverse the normal usage of the tip and ring and tip-1 and ring-1 pairs. To match up the audio pairs with the Cisco E&M audio pairs, connect tip and ring on the PBX side to tip-1 and ring-1 on the Cisco side, and tip-1 and ring-1 on the PBX side to tip and ring on the Cisco side.

See the "Troubleshooting E&M Interfaces at the Physical Level" section on page 26 for more information on the wiring arrangement.

Verifying Supervision Signaling

In this step, verify that on-hook/off-hook signals are being transmitted between the PBX and the gateway. If you are accessing the router through the console port, enter the command **terminal monitor**, otherwise no debug output is displayed.

Follow these steps to verify supervision signaling:

SUMMARY STEPS

1. **enable**
2. **debug vpm signal**
3. Place a call from the PBX to the gateway.

DETAILED STEPS

Step 1 At the Router> prompt, enter **enable** to enter privileged EXEC mode. Enter your password if prompted.

Step 2 Turn on the command **debug vpm signal** on the Cisco gateway. This command is used to collect debug information for signaling events (on-hook/ off-hook transitions).

Step 3 Place a call from the PBX to the gateway. The PBX should seize the E&M trunk and send the on-hook -> off-hook signal transition to the gateway. The following output displays a successful reception of these signals.

In this example, the PBX is seizing the router trunk. The router E&M voice port transitions from on-hook to off-hook. This shows that on-hook, off-hook signaling is being received from the PBX.

```
Router# debug vpm signal
Voice Port Module signaling debugging is enabled
*Mar 2 05:54:43.996: htsp_process_event: [1/0/0, 1.4 , 34]
em_onhook_offhookhtsp_setup_ind
*Mar 2 05:54:44.000: htsp_process_event: [1/0/0, 1.7 , 8]
*Mar 2 05:54:44.784: htsp_process_event: [1/0/0, 1.7 , 10]
*Mar 2 05:54:44.784: htsp_process_event: [1/1/0, 1.2 , 5]
fxsls_onhook_setuphtsp_alerthtsp_alert_notify
*Mar 2 05:54:44.788: htsp_process_event: [1/0/0, 1.7 , 11]
```

```
*Mar 2 05:54:44.788: htsp_process_event: [1/1/0, 1.5 , 11]
fxsls_waitoff_voice
```

If no output is displayed, there is probably a problem with the E&M supervision signaling.

Table 30 describes some possible problems and the corresponding solutions.

Table 30 E&M Supervisory Signaling Troubleshooting Table

Symptom	Problem	Solution
No dial tone from the Cisco port. No port seizure activity seen on the Cisco gateway.	The PBX is not configured to seize the E&M port connected to the Cisco equipment.	Configure the PBX to seize the trunk.
The port is seized but the call does not go through.	There is an E&M Type (I, II, III or V) mismatch between the PBX and the gateway.	Verify (and change if necessary) the E&M type configured on the Cisco equipment. See the "Confirming the Cisco IOS Gateway Configuration" section on page 33.
The port has unbreakable dial tone. The Cisco gateway is unable to send digits when a port is seized.	Incorrect wiring arrangement (cabling) for the supervisory signaling leads (E and M leads for Type I and V; E, M, SB, and SG leads for Types II and III).	Wiring issues are usually the primary source of analog E&M problems. Make sure the cable used corresponds to the required PBX and Cisco gateway pinout, interface type, and audio operation setup. For more information see the "Troubleshooting E&M Interfaces at the Physical Level" section on page 26.
The port on the Cisco gateway cannot be seized. The Cisco gateway is unable to send digits.	The Cisco gateway configuration changes are not enabled.	Issue the **shutdown/no shutdown** command sequence on the E&M voice port after the configuration changes.
Calls cannot be made in two directions.	On-hook or off-hook signals have been sent one way only.	This is probably an indication of a defective cable, where one path of the signaling leads is wired correctly and the other side is not.

Verifying That the Cisco Equipment and PBX Are Sending and Receiving Digits

After confirming successful supervisory (on-hook/off-hook) signaling between the PBX and the gateway, you need to verify that address information (DTMF digits or pulse dial) is being passed between both ends.

Note DTMF digits are sent on the audio path. Pulse-dial address information is sent by pulsing on the E or M lead.

There are three start dial supervision line protocols that analog E&M uses to define how the equipment passes address information:

- Immediate start
- Wink start
- Delay dial

Make sure both the Cisco gateway and the PBX are configured with the same start dial supervision protocol. Verify that information is being passed by performing the following steps:

SUMMARY STEPS

1. **enable**
2. **debug vpm signal, debug vtsp dsp**
3. Place a call from the PBX to the gateway.
4. Place a call from the gateway to the PBX.

DETAILED STEPS

Step 1 At the Router> prompt, enter **enable** to enable privileged EXEC mode. Enter your password if prompted.

Step 2 Turn on the commands **debug vpm signal** and **debug vtsp dsp** on the Cisco gateway. The command **debug vtsp dsp** is useful for displaying the digits received and sent by the voice DSPs.

Step 3 Place a call from the PBX to the gateway. The following output displays a successful reception of the expected digits. In this example, the router receives a call from the PBX to extension 2000.

```
Router# show debugging
Voice Port Module signaling debugging is on
Voice Telephony dsp debugging is on
Router#
*Mar 1 03:16:19.207: htsp_process_event: [1/0/0, 1.4 , 34]
em_onhook_offhookhtsp_setup_*Mar 1 03:16:19.207: htsp_process_event: [1/0/0, 1.7 , 8]
*Mar 1 03:16:19.339: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN:
digit=2,rtp_=0x9961CF03
*Mar 1 03:16:19.399: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF:
digit=2,duration=*Mar 1 03:16:19.539: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN:
digit=0,rtp_=0x9961CF03
2.
*Mar 1 03:16:19.599: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF:
digit=0,duration=*Mar 1 03:16:19.739: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN:
digit=0,rtp_=0x9961CF03
*Mar 1 03:16:19.799: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF:
digit=0,duration=*Mar 1 03:16:19.939: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN:
digit=0,=rtp_=0x9961CF03
*Mar 1 03:16:19.999: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF:
digit=0,duration=*Mar 1 03:16:19.999: htsp_process_event: [1/0/0, 1.7 , 10]
*Mar 1 03:16:19.999: htsp_process_event: [1/1/0, 1.2 , 5]
fxsls_onhook_setuphtsp_alerthtsp_*Mar 1 03:16:20.003: htsp_process_event: [1/0/0, 1.7 ,
11]
*Mar 1 03:16:20.003: htsp_process_event: [1/1/0, 1.5 , 11] fxsls_waitoff_voice
*Mar 1 03:16:27.527: htsp_process_event: [1/1/0, 1.5 , 34] fxsls_waitoff_offhook
*Mar 1 03:16:27.531: htsp_process_event: [1/0/0, 1.7 , 6] em_offhook_connectem_stop_
```

Step 4 Place a call from the gateway to the PBX. The following output displays the digits the Cisco equipment is sending. In this example, the PBX receives a call from the router to extension 1000. If digits are not parsed properly, the wink start timers being triggered.

```
Log Buffer (1000000 bytes):
*Mar 1 03:45:31.287: htsp_process_event: [1/1/1, 1.2 , 34] fxsls_onhook_offhook
htsp_setup_ind
*Mar 1 03:45:31.291: htsp_process_event: [1/1/1, 1.3 , 8]
*Mar 1 03:45:33.123: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN: digit=1
, rtp_timestamp=0xCD4365D8
*Mar 1 03:45:33.283: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF: digit=1,
duration=205
*Mar 1 03:45:33.463: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN: digit=0
, rtp_timestamp=0xCD4365D8
*Mar 1 03:45:33.643: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=225
*Mar 1 03:45:33.823: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN: digit=0
, rtp_timestamp=0xCD4365F0
*Mar 1 03:45:34.003: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=222
*Mar 1 03:45:34.203: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_BEGIN: digit=0
, rtp_timestamp=0xCD4365F0
*Mar 1 03:45:34.411: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=252
*Mar 1 03:45:34.415: htsp_process_event: [1/1/1, 1.3 , 10]
*Mar 1 03:45:34.415: htsp_process_event: [1/0/0, 1.4 , 5] em_onhook_setup em_of
fhook
*Mar 1 03:45:34.415: htsp_process_event: [1/0/0, 1.13 , 43] em_start_timer: 120
0 ms
*Mar 1 03:45:34.715: htsp_process_event: [1/0/0, 1.10 , 34] em_wink_offhookem_s
top_timers em_start_timer: 1200 ms
*Mar 1 03:45:34.923: htsp_process_event: [1/0/0, 1.11 , 22] em_wink_onhook em_s
top_timers em_send_digit htsp_dial
*Mar 1 03:45:34.923: digit=1, components=2, freq_of_first=697, freq_of_second
=1209, amp_of_first=16384, amp_of_second=16384
*Mar 1 03:45:34.923: digit=0, components=2, freq_of_first=941, freq_of_second
=1336, amp_of_first=16384, amp_of_second=16384
*Mar 1 03:45:34.923: digit=0, components=2, freq_of_first=941, freq_of_second
=1336, amp_of_first=16384, amp_of_second=16384
*Mar 1 03:45:34.923: digit=0, components=2, freq_of_first=941, freq_of_second
3.
=1336, amp_of_first=16384, amp_of_second=16384
*Mar 1 03:45:35.727: vtsp_process_dsp_message: MSG_TX_DIALING_DONE
*Mar 1 03:45:35.727: htsp_process_event: [1/0/0, 1.7 , 19] em_offhook_digit_don
ehtsp_alerthtsp_alert_notify
```

Table 31 shows digit send and receive problems and the corresponding solutions. These problems can be diagnosed if you notice that the wink timers are being triggered.

Table 31 Digit Send and Receive Troubleshooting Table

Problem	Solution
Start dial supervision mismatch or timing issues between the PBX and gateway.	Make sure both end systems are configured with the same start dial protocol.
Audio operation mismatch (for example, one side configured for 2-wire, the other for 4-wire) or wiring problems on the audio path.	Verify the gateway configuration and PBX configuration and the wiring arrangement. For more information see the "Troubleshooting E&M Interfaces at the Physical Level" section on page 26. **Note** DTMF digits are passed on the audio path. Even if the line supervision signaling is operating correctly, DTMF digits are not passed if the audio path is broken.
Wiring problems in the audio path.	Verify the wiring arrangement. See the "Troubleshooting E&M Interfaces at the Physical Level" section on page 26.

In the 4-wire audio mode, some PBX and key system products reverse the normal usage of the tip and ring and tip-1 and ring-1 pairs. In that case, to match up the audio pairs with the Cisco E&M audio pairs, you might need to connect tip and ring on the PBX side to tip-1 and ring-1 on the Cisco side, and tip-1 and ring-1 on the PBX side to tip and ring on the Cisco side. If the audio pairs are not correctly matched up in 4-wire mode, there is no end-to-end audio path in either direction. If the E&M interface is configured to send dial strings as dial pulse (which works by pulsing on the E or M lead), it is possible to establish a call even with the 4-wire audio pairs reversed, but there will be little or no audio path in either direction after the call is established (there might be low-level transmission of audio, but the sound levels will be far too low for comfort). If you are using DTMF to send dial strings, the E&M interface goes off hook at the start of the call, but the call does not complete, because one end sends the DTMF tones on the wrong audio pair, and the other end does not receive these DTMF tones.

Verifyinng That the Gateway Sends the Expected Digits to the PBX

Once the two end devices are able to successfully send supervision and address signaling (on-hook, off-hook, digits), we can assume that the troubleshooting process is complete for analog E&M signaling, and it is now in the dial plan domain. For more information about dial plan design, refer to the *Voice Design and Implementation Guide*, document ID 5756.

If incomplete or incorrect digits are sent by the Cisco equipment, then the Telco switch (CO or PBX), cannot ring the correct station.

On POTS dial peers, the only digits that are sent to the other end are the ones specified with the command **destination-pattern** and the wild card character (".”). The POTS dial peer command **prefix** can be used to include a dial-out prefix that the system enters automatically instead of people dialing it. Refer to the following output example for a sample configuration.

```
!
!--- Some output ommited.
!
!--- E&M Voice Port
!
voice-port 1/0/0
type 2
signal immediate
```

```
!
!--- FXS Voice Port
voice-port 1/1/0
!
dial-peer voice 1 pots
destination-pattern 2000
port 1/1/0
!
!--- Dial peer 2 is in charge of forwarding calls to the E&M voiceport 1/0/0.
!--- In this case the digit "1" in the destination pattern will be dropped and the syste
!--- will transmit the 3 digits matched by the "." wildcard.
!--- Notice that since the PBX is expecting the "1000" string, the prefix command is used.
!
dial-peer voice 2 pots
destination-pattern 1...
port 1/0/0
prefix 1
!
```

Verify That the Gateway Receives the Expected Digits from the PBX

Verify that the digits received from the PBX match a dial peer in the gateway. If incomplete or incorrect digits are sent by the PBX, a dial peer cannot be matched. Use the command **debug vtsp dsp** to view the digits received by the analog E&M voice port.

To verify which dial peers match a specific string use the command **show dialplan number**. Refer to the following sample output example.

```
Router# show dialplan number 1000
Macro Exp.: 1000
VoiceEncapPeer2
information type = voice,
tag = 2, destination-pattern = `1...',
answer-address = `', preference=0,
group = 2, Admin state is up, Operation state is up,
incoming called-number = `', connections/maximum = 0/unlimited,
application associated:
type = pots, prefix = `1',
session-target = `', voice-port = `1/0/0',
direct-inward-dial = disabled,
register E.164 number with GK = TRUE
Connect Time = 19644, Charged Units = 0,
Successful Calls = 63, Failed Calls = 2,
Accepted Calls = 65, Refused Calls = 0,
Last Disconnect Cause is "10 ",
Last Disconnect Text is "normal call clearing.",
Last Setup Time = 28424467.
Matched: 1000 Digits: 1
Target:
Router# show dialplan number 2000
Macro Exp.: 2000
VoiceEncapPeer1
information type = voice,
tag = 1, destination-pattern = `2000',
answer-address = `', preference=0,
group = 1, Admin state is up, Operation state is up,
incoming called-number = `', connections/maximum = 0/unlimited,
application associated:
type = pots, prefix = `',
session-target = `', voice-port = `1/1/1',
direct-inward-dial = disabled,
```

```
register E.164 number with GK = TRUE
Connect Time = 19357, Charged Units = 0,
Successful Calls = 68, Failed Calls = 8,
Accepted Calls = 76, Refused Calls = 0,
Last Disconnect Cause is "10 ",
Last Disconnect Text is "normal call clearing.",
Last Setup Time = 28424186.
Matched: 2000 Digits: 4
Target:
```

Unbreakable Dial Tone

A common problem occurs when the router seizes the local PBX but as digits are dialed, the dial tone stays. The calling party is unable to pass the DTMF tones or digits to the terminating device, resulting in callers being unable to dial the desired extension or interact with the device that needs DTMF tones, such as a voice mail or interactive voice response (IVR) application. This problem can result from a number of reasons such as:

- DTMF tones not sent
- DTMF tones not understood
- DTMF tones too distorted to be understood
- Other signaling and cabling issues

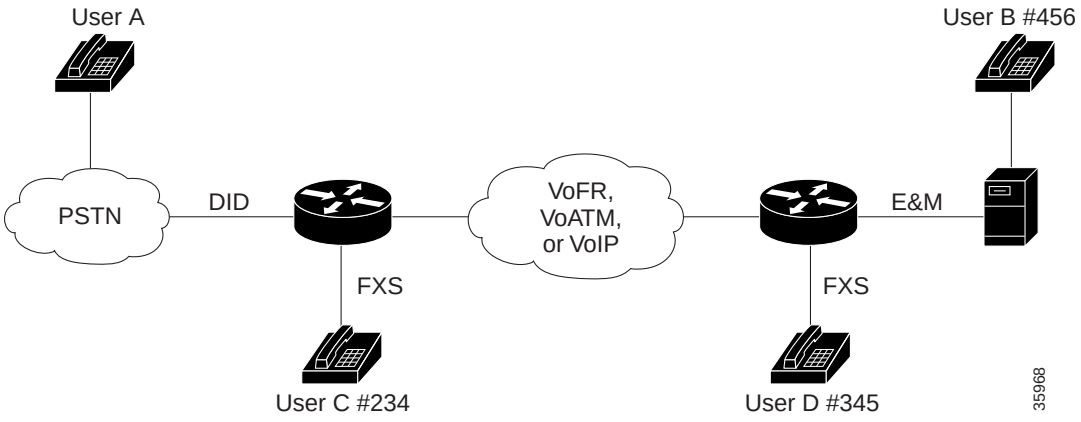
For more information, refer to *Inability To Break Dialtone in a Voice over IP Network* , document ID 22376.

Analog DID Interfaces

Direct inward dialing (DID) is a service offered by telephone companies that enables callers to dial directly to an extension on a PBX without the assistance of an operator or automated call attendant. This service makes use of DID trunks, which forward only the last three to five digits of a phone number to the PBX. The DID state machine is identical to the E&M state machine.

Figure 29 shows a hypothetical topology in which a user connected to the PSTN (User A) dials various numbers and is connected to the appropriate extensions on a PBX.

Figure 29 DID Support for Cisco 2600 and Cisco 3600 Series Routers



Number Dialed by User A	Number Received by Router	Extension Receiving Call
555-1234	234	User C
555-1345	345	User D
555-1456	456	User B
555-1678	678	No dial-peer match found; fast busy tone is played

DID Hardware Troubleshooting

A DID voice interface connects directly to a standard telephone, fax machine, or similar device and supplies ring, voltage, and dial tone.

Troubleshoot DID hardware by checking the following sections:

- Software Compatibility, page 42
- Cabling, page 42
- Shutdown Port, page 43

Software Compatibility

For interface cards inserted into Cisco 1600 series, Cisco 1700 series, Cisco 2600 series, Cisco 3600 series, Cisco 3700 series, and Cisco ICS 7750 platforms, refer to the compatibility tables in the "Overview of Cisco Interface Cards" chapter in the *Cisco Interface Cards Installation Guide.*

Cabling

The two-port and four-port DID interface cards support the RJ-11 connector. Illustrations of the connector ports are shown in Figure 30 and Figure 31. Information about LEDs can be found in the "Connecting Voice Interface Cards to a Network" chapter of the *Cisco Interface Card Hardware Installation Guide*.

Figure 30 Two-Port Analog DID Voice Interface Card

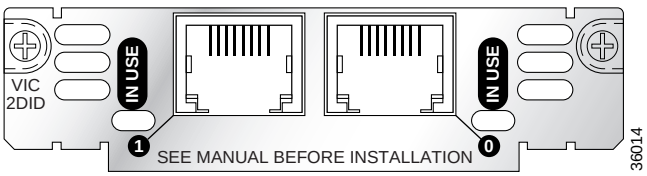
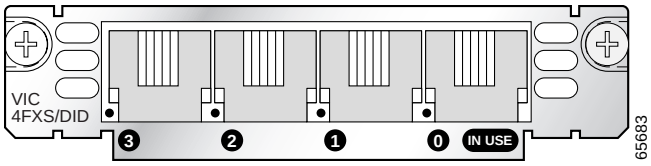


Figure 31 Four-Port Analog FXS/DID Voice Interface Card



For more information about the VIC-2DID interface card, refer to *Understanding 2 Port Direct Inward Dial (2 DID) Voice Interface Cards*, document ID 15268.

Shutdown Port

Check to make sure that the port is not shut down. Enter the **show voice port** command with the voice port number that you are troubleshooting, which will tell you:

- If the voice port is up. If it is not, use the **no shutdown** command to make it active.
- What parameter values have been set for the voice port, including default values (these do not appear in the output from **the show running-config command**). If these values do not match those of the telephony connection you are making, reconfigure the voice port.

Verifying Direct Inward Dialing Voice-Port Configuration

To verify voice-port configuration, enter the **show voice port** command. You can specify a voice port or view the status of all configured voice ports. In the following example, the specified port is configured for DID.

```
Router# show voice port 1/1/0
Foreign Exchange Station with Direct Inward Dialing (FXS-DID) 1/1/0 Slot is
1, Sub-unit is 1, Port is 0
  Type of VoicePort is DID-IN
  Operation State is DORMANT
  Administrative State is UP
  No Interface Down Failure
  Description is not set
  Noise Regeneration is enabled
  Non Linear Processing is enabled
  Music On Hold Threshold is Set to -38 dBm
  In Gain is Set to 0 dB
  Out Attenuation is Set to 0 dB
  Echo Cancellation is enabled
  Echo Cancel Coverage is set to 8 ms
  Playout-delay Mode is set to default
  Playout-delay Nominal is set to 60 ms
  Playout-delay Maximum is set to 200 ms
  Playout-delay Minimum mode is set to default, value 4 ms
  Playout-delay Fax is set to 300 ms
  Connection Mode is normal
  Connection Number is not set
  Initial Time Out is set to 10 s
  Interdigit Time Out is set to 10 s
  Call Disconnect Time Out is set to 3 s
  Ringing Time Out is set to 180 s
  Wait Release Time Out is set to 3 s
  Companding Type is u-law
  Region Tone is set for US
```

```
Analog Info Follows:
Currently processing none
Maintenance Mode Set to None (not in mtc mode)
Number of signaling protocol errors are 0
Impedance is set to 600r Ohm
Station name Chalil Mohanan, Station number 1234567

Voice card specific Info Follows:
Signal Type is wink-start
Dial Type is dtmf
In Seizure is inactive
Out Seizure is inactive
Digit Duration Timing is set to 100 ms
InterDigit Duration Timing is set to 100 ms
Pulse Rate Timing is set to 10 pulses/second
InterDigit Pulse Duration Timing is set to 750 ms
Clear Wait Duration Timing is set to 400 ms
Wink Wait Duration Timing is set to 200 ms
Wait Wink Duration Timing is set to 550 ms
Wink Duration Timing is set to 200 ms
Delay Start Timing is set to 300 ms
Delay Duration Timing is set to 2000 ms
Dial Pulse Min. Delay is set to 140 ms
Percent Break of Pulse is 60 percent
Auto Cut-through is disabled
Dialout Delay for immediate start is 300 ms
```

Voice Port Testing Commands

Voice port testing commands allow you to force voice ports into specific states for testing. The following types of voice-port tests are covered:

- Detector-Related Function Tests, page 44
- Loopback Function Tests, page 45
- Tone Injection Tests, page 46
- Relay-Related Function Tests, page 46
- Fax/Voice Mode Tests, page 47

Detector-Related Function Tests

Using the **test voice port detector** command, you are able to force a particular detector into an on or off state, perform tests on the detector, and then return the detector to its original state.

To configure this feature, enter these commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/subunit/port detector {m-lead | battery-reversal | loop-current | ring | tip-ground | ring-ground | ring-trip} {on | off}`	Identifies the voice port you want to test. • Enter a keyword for the detector under test and specify whether to force it to the on or off state. **Note** For each signaling type (E&M, FXO, FXS), only the applicable keywords are displayed. The **disable** keyword is displayed only when a detector is in the forced state.
Step 2	`Router# test voice port slot/subunit/port detector {m-lead | battery-reversal | loop-current | ring | tip-ground | ring-ground | ring-trip} disable`	Identifies the voice port on which you want to end the test. • Enter a keyword for the detector under test and the keyword **disable** to end the forced state. **Note** For each signaling type (E&M, FXO, FXS), only the applicable keywords are displayed. The **disable** keyword is displayed only when a detector is in the forced state.

Loopback Function Tests

To establish loopbacks on a voice port, enter the following commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/subunit/port loopback {local | network}`	Identifies the voice port you want to test and enters a keyword for the loopback direction. **Note** A call must be established on the voice port under test.
Step 2	`Router# test voice port slot/subunit/port loopback disable`	Identifies the voice port on which you want to end the test and enters the keyword disable to end the loopback.

Tone Injection Tests

To inject a test tone into a voice port, enter the following commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/subunit/port inject-tone {local | network} {1000hz | 2000hz | 200hz | 3000hz | 300hz | 3200hz | 3400hz | 500hz | quiet}`	Identifies the voice port you want to test and enter keywords for the direction to send the test tone and for the frequency of the test tone. **Note** A call must be established on the voice port under test.
Step 2	`Router# test voice port slot/subunit/port inject-tone disable`	Identifies the voice port on which you want to end the test and enter the keyword **disable** to end the test tone. **Note** The **disable** keyword is available only if a test condition is already activated.

Relay-Related Function Tests

To test relay-related functions on a voice port, enter the following commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/subunit/port relay {e-lead | loop | ring-ground | battery-reversal | power-denial | ring | tip-ground} {on | off}`	Identifies the voice port you want to test. • Enter a keyword for the relay under test and specify whether to force it to the on or off state. **Note** For each signaling type (E&M, FXO, FXS), only the applicable keywords are displayed. The **disable** keyword is displayed only when a relay is in the forced state.
Step 2	`Router# test voice port slot/subunit/port relay {e-lead | loop | ring-ground | battery-reversal | power-denial | ring | tip-ground} disable`	Identifies the voice port on which you want to end the test. • Enter a keyword for the relay under test, and the keyword **disable** to end the forced state. **Note** For each signaling type (E&M, FXO, FXS), only the applicable keywords are displayed. The **disable** keyword is displayed only when a relay is in the forced state.

Fax/Voice Mode Tests

The **test voice port switch fax** command forces a voice port into fax mode for testing. After you enter this command, you can use the **show voice call** or **show voice call summary** command to check whether the voice port is able to operate in fax mode. If no fax data is detected by the voice port, the voice port remains in fax mode for 30 seconds and then reverts automatically to voice mode.

The **disable** keyword ends the forced mode switch; however, the fax mode ends automatically after 30 seconds. The **disable** keyword is available only while the voice port is in fax mode.

To force a voice port into fax mode and return it to voice mode, enter the following commands, beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/subunit/port switch fax`	Identifies the voice port you want to test. • Enter the keyword **fax** to force the voice port into fax mode.
Step 2	`Router# test voice port slot/subunit/port switch disable`	Identifies the voice port on which you want to end the test. • Enter the keyword **disable** to return the voice port to voice mode.

Troubleshooting Digital Voice Interfaces to the IP Network

Digital voice ports are found at the intersection of a packet voice network and a digital, circuit-switched telephone network. Digital voice telephony interfaces include T1 or E1 channel-associated signaling (CAS), ISDN primary-rate interface (PRI) or basic rate interface (BRI), and E1 R2 signaling.

To troubleshoot digital voice interfaces, see the following sections:

If you are troubleshooting a connection to a PBX, you might the PBX interoperability notes useful. These notes contain configuration information for Cisco gateways and several types of PBXs. To access these notes, use the following website:

http://www.cisco.com/warp/public/779/largeent/avvid/inter_operability/

Checking the Hardware

Digital voice interface hardware connects a router or access server to a line from a circuit-switched telephony device in a PBX or the public switched telephone network (PSTN).

Troubleshoot digital voice hardware by checking the following:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Software Compatibility

To ensure that your card is compatible with your software, check the following:

- For network modules inserted into Cisco modular access routers, refer to the compatibility tables in the "Overview of Cisco Network Modules" chapter in the *Cisco Network Modules Hardware Installation Guide.*
- For interface cards inserted into Cisco modular access routers, refer to the compatibility tables in the "Overview of Cisco Interface Cards" chapter in the *Cisco Interface Cards Installation Guide.*

Cabling

Cabling for the digital ports varies by platform:

- Cisco 1600 series, Cisco 1700 series, Cisco 2600 series, Cisco 3600 series, Cisco 3700 series, and Cisco ICS 7750 platforms that use the Multiflex Trunk Interface card use an RJ-48C cable. Refer to the *Cisco Interface Card Hardware Installation Guide* for information about digital Cisco interface cards.
- Cisco 7200 VXR platforms use RJ-48C cables for the port adapter. See the *MIX-Multichannel T1/E1 Port Adapter Installation and Configuration Guide* for more information.
- Cisco AS5300 universal access servers use RJ-45 cables for the T1 or E1 interface. A VoIP feature is also required for voice traffic. See the *Cisco AS5300 Module Installation Guide*.
- Cisco AS5350 and 5400 universal gateways use RJ-45 cables for the four-port card and a 36-pin cable to RJ-45 interface for the eight-port card. For more information about cabling these platforms, see the "Cabling Specifications" chapter of the *Cisco AS5350 and AS5400 Universal Gateway Card Installation Guide.*

T1/E1 Trunk and Digital Voice Port Pinouts (RJ-48)

Figure 32 shows the RJ-48 connector wiring for the T1/E1 trunk cable and the digital voice port cable; Table 32 lists the pinouts.

Figure 32 RJ-48-to-RJ-48 T1/E1 Cable Wiring

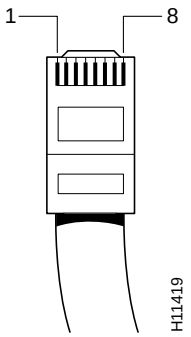


Table 32 Pinouts for T1/E1 Trunk and Digital Voice Port (RJ-48)

Pin[1]	Signal
1	RX (input)
2	RX (input)
3	–
4	TX (output)
5	TX (output)
6	–
7	–
8	–

1. Any pin not referenced on a connector is not connected.

T1/E1 Trunk and Digital Voice Port Pinouts (RJ-45)

Table 33 T1 or E1 Port Pinouts (RJ-45)

RJ-45 Pin	Description
1	RX tip
2	RX ring
3	RX shield
4	TX tip
5	TX ring
6	TX shield
7	-
8	-

Shutdown Port

If the port is not operational, check to make sure the port is not shut down. Enter the **show voice port** command with the voice port number that you are troubleshooting, which tells you:

- If the voice port is up. If it is not, use the **no shutdown** command to make it active.
- What parameter values have been set for the voice port, including default values. (these do not appear in the output from **the show running-config command**.) If these values do not match those of the telephony connection you are making, reconfigure the voice port.

Checking the Digital Signal Processors

Digital signal processors (DSPs) enable Cisco platforms to efficiently process digital voice traffic. The following symptoms can be attributed to DSP hardware or software issues:

- No audio heard by either party or one-way audio on the voice path after the call is connected.

- Call setup failure, such as the inability to detect or transmit proper Channel Associated Signaling (CAS) state transitions.
- Channels are stuck in the PARK state and cannot be used.
- Error messages on the console or in the router log complain of DSP timeouts.

To check your DSPs, use the following sections:

- Voice DSP Control Message Logger, page 4
- Voice Call Tuning, page 9
- Voice DSP Crash Dump File Analysis, page 9
- Troubleshooting Universal Port SPEs, page 13
- DSP Troubleshooting Links, page 29

Voice DSP Control Message Logger

This section contains the following information:

- Message Logger Overview, page 4
- Configuration Tasks, page 5
- Configuration Examples, page 8

Caution Using the logger feature in a production network environment increases CPU and memory usage on the gateway.

Note We recommend that you work closely with your Cisco representative to use this feature. If you are experiencing problems with certain voice calls, the engineering team at Cisco might ask you to capture the control messages using the voice DSP logger. You can capture these messages by turning on the logger, repeating the problematic calls, and capturing the logs. Only Cisco engineers can determine if you should send the logs in for further review.

Message Logger Overview

The Voice DSP Control Message Logger feature provides improved debugging capabilities through Cisco IOS software and allows loggin of control messages that pass through the voice DSP firmware on the host port interface (HPI). The logged messages can later be examined for diagnosis of voice problems.

There are two main types of HPI messages that flow through the HPI interface: control messages and data messages. Control messages carry control information between Cisco IOS software and the DSP. Data messages carry voice data.

The Voice DSP Contol Message Logger feature captures control messages sent between the platform-independent portions of Cisco IOS software and the DSP. The HPI subsystem that is in Cisco IOS software contains the platform-independent portion of Cisco IOS software. This feature addresses the sequence and contents of the control messages. The logged messages can be checked for parameters that might cause undesirable DSP behavior including the following:

- Incorrect parameters

- Out-of-sequence function calls
- Interactions between parameters of different HPI calls

In many cases, DSP problems have been the result of bad control messages. By logging all of these messages for offline analysis, you can better integrate and debug at-speed issues for analysis.

Message Capture

Message capture occurs when voice control messages are captured and passed between the Cisco IOS software and the DSP to a ring buffer. Some of these messages are sent in fast-path routines that run at a high priority, so the capture of the message must be done as quickly as possible. After the fast-path routine messages have been sent, a normal priority process sends the messages that are waiting in the ring buffer to off-router data storage through the Cisco IOS File System (IFS).

The size of the ring buffer is configurable through the use of the **voice hpi capture** command. If the ring buffer fills up faster than the normal priority process can move the messages off the router, some of the control messages are dropped.

Counters keep track of the number of messages that are waiting in the ring buffer, the number of messages that are sent, and the number of messages that are dropped. When message capture is enabled and a message arrives for which there is no buffer space, a missed-message count is started. The next time there is room for a message on the ring, the dropped-message count is included with the message data. This alerts the software that processes the messages to the missed messages, and it provides data capture feedback that helps you configure the ring buffer size to your specifications.

If messages are dropped during the capture, the ability to check the messages becomes limited. A complete capture is required for analysis.

Benefits

Improved DSP Reliability

This feature improves the reliability of DSPs by improving debug capabilities. Unexpected sequences of calls or parameters that cause DSP problems are difficult to debug because many calls can be made to the DSP before any ill effects are noticed. Systems that are running under load are more likely to encounter subtle timing-related issues that occur infrequently and are very hard to reproduce and debug. These parameters are marked as bad in HPI calls to the DSP, and they can cause undesirable DSP behavior. There fore, the logger intercepts those parameters that pass between Cisco IOS software and the DSP that can later be checked for errors.

Robust Firmware

This feature makes the T1-based DSP firmware more robust, adding debug capabilities and enabling better field support.

Restrictions

The Voice DSP Contol Message Logger feature is supported only on systems that use the HPI interface.

Configuration Tasks

See the following sections for configuration tasks for the Voice DSP Control Message Logger feature. Each task in the list is identified as either required or optional.

- Configuring the Voice DSP Control Message Logger (required)
- Verifying the Voice DSP Control Message Logger (optional)

Configuring the Voice DSP Control Message Logger

You can start the message logger by choosing the amount of memory (greater than 324 bytes) that the buffer-queueing system can allocate to the free message pool. HPI messages are captured until buffer space runs out. Once the buffer-queueing system is running, the transport process attempts to connect to a new or existing capture destination URL. A version message is written to the URL, and if the version message is accepted, any messages placed into the message queue are written to the URL. If a new URL is entered using command-line interface (CLI), an open URL is closed, and the system tries to write to the new URL. If the new URL fails, the transport process exits. The transport process is restarted when another URL is entered or the system is restarted.

To configure the message logger, use the following commands beginning in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **show voice hpi capture**
3. **debug hpi capture**
4. **configure terminal**
5. voice hpi capture buffer *size*
6. voice hpi capture destination *url*
7. exit
8. show voice hpi capture
9. configure terminal
10. no voice hpi capture buffer 0
11. exit
12. show voice hpi capture

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice hpi capture** **Example:** `Router# show voice hpi capture`	(Optional) Displays the capture status and statistics. • Use this command to confirm logger status and examine the logger status output when the logger is running.
Step 3	**debug hpi capture** **Example:** `Router# debug hpi capture`	(Optional) Turns on the debug output for the logger. • It is recommended that you enable the debug output for the logger when you are interacting with it using CLI.

	Command	Purpose
Step 4	**configure** {**terminal** | **memory** | **network**} **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 5	**voice hpi capture buffer** *size* **Example:** `Router(config)# voice hpi capture buffer 122`	(Optional if you already have a nonzero buffer) Allocates the buffer for storing captured messages. • Starts the logger by giving it a nonzero buffer size. • The **no** form of the command turns the logger off by setting the buffer size to zero. Valid range is from 0 to 9000000. • If the buffer overflows so that messages are dropped during the capture, the buffer size needs to be increased and the capture needs to be restarted. **Note** To change buffer size, first configure buffer size to zero, and then set the buffer size to your specifications.
Step 6	**voice hpi capture destination** *url* **Example:** `Router(config)# voice hpi capture destination 172.14.33.255`	(Optional if you already have a destination or do not want to change the currently assigned destination) Sets up an FTP destination file to which the logged data is sent. • The *url* argument is the destination address.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 8	**show voice hpi capture** **Example:** `Router# show voice hpi capture`	(Optional) Displays the capture status and statistics. • Use this command to confirm logger status and examine the logger status output when the logger is running. **Note** At this point, you can execute voice calls and capture the control messages. You can capture these messages by turning on the logger, repeating problematic calls, and capturing the logs. Cisco engineers can determine if you should send the logs in for further review.
Step 9	**configure** {**terminal** | **memory** | **network**} **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 10	**no voice hpi capture buffer 0** **Example:** `Router(config)# no voice hpi capture buffer 0`	(Optional if you already have a nonzero buffer) Turns the logger off by setting the buffer size to zero and stops message capture.

	Command	Purpose

Step 11	`exit`
	
	Example:
	`Router(config)# exit`
		Exits global configuration mode and returns to privileged EXEC mode.

Step 12	`show voice hpi capture`
	
	Example:
	`Router# show voice hpi capture`
		Verifies that no messages were dropped during the capture.
		Note At this point, gather the captured messages in the destination files on your PC or UNIX station and send the information to Cisco TAC for analysis.

Verifying the Voice DSP Control Message Logger

To verify and print capture status and statistics, use the **show voice hpi capture** privileged EXEC command. This command displays the capture status and statistics and checks that the message counter is incrementing. If messages are being dropped consistently, try increasing the buffer size.

Note If you want to stop the logger or change the buffer to another size, first set the buffer size to zero.

Troubleshooting Tips

Use the **debug hpi capture** command in privileged EXEC mode to turn on the debug output for the logger. Enable the debug output for the logger by using the CLI.

Configuration Examples

This section provides configuration examples for the Voice DSP Control Message Logger feature. This section contains the following examples:

- Starting the Logger Feature Example
- Setting up an FTP Destination Example
- Verifying Configuration Example

Starting the Logger Feature Example

In the following example, the **voice hpi capture buffer** command is used in global configuration mode to start the logger by giving it a buffer size of 700000:

```
Router(config)# voice hpi capture buffer 700000

*Mar  1 00:24:47.090:caplog:caplog_cli_interface:hpi capture buffer size set to 700000
 bytes
*Mar  1 00:24:47.090:caplog:caplog_logger_init:TRUE, Started task HPI Logger (PID 140
*Mar  1 00:24:47.150:caplog:caplog_cache_init:TRUE, malloc_named(699952), 2134 elements
(each 328 bytes big)
*Mar  1 00:24:47.154:caplog:caplog_logger_proc:Terminating...
```

In the following example, the **show voice hpi capture** command is used in privileged EXEC mode to examine the logger status output now that the logger is enabled:

```
Router# show voice hpi capture
```

```
HPI Capture is on and is logging to URL <www.company.com>0 messages sent to URL, 0
messages droppedMessage Buffer (total:inuse:free)  2134:0000:2134Buffer Memory:699952
bytes, Message size:328 bytes
```

Setting up an FTP Destination Example

In the following example, the **voice hpi capture destination** command is used in global configuration mode to set up an FTP destination file where the logged data can be sent:

```
Router(config)# voice hpi capture destination ftp://100.00.100.200/d:\test_data.dat

*Mar  1 00:26:54.617:caplog:caplog_cli_interface:hpi capture
destination:ftp://100.00.100.200/d:\test_data.dat
*Mar  1 00:26:54.621:caplog:caplog_logger_init:TRUE, Started task HPI Logger (PID 140)
*Mar  1 00:26:54.621:caplog:caplog_logger_proc:Attempting to open
ftp://100.00.100.200/d:\test_data.dat
*Mar  1 00:26:57.091:caplog:caplog_logger_proc:Logging to
ftp://100.00.100.200/d:\test_data.dat
```

In the following example, the **show voice hpi capture** command is used in privileged EXEC mode to examine the logger status output:

```
Router# show voice hpi capture

HPI Capture is on and is logging to URL ftp://172.23.184.216/d:\test_data.dat1 messages
sent to URL, 0 messages droppedMessage Buffer (total:inuse:free)  2134:0000:2134Buffer
Memory:699952 bytes, Message size:328 bytes
```

Verifying Configuration Example

In the following example, the **show voice hpi capture** command is used in privileged EXEC mode to check on the status of the logger before, during, and after configuration:

```
Router# show voice hpi capture

HPI Capture is off and is logging to URL <no URL>
0 messages sent to URL, 0 messages dropped
Message Buffer (total:inuse:free)  0000:0000:0000
Buffer Memory:0 bytes, Message size:328 bytes
```

Voice Call Tuning

The Voice Call Tuning feature monitors the interface between Cisco IOS software and a system's digital signaling processors (DSPs) in real time and reports status on the following: packet flow, DSP state, echo-cancellation state, and jitter state. The feature also allows you to manipulate echo-cancellation and jitter-buffer parameters in real time. For details on this feature, see the "Voice Call Tuning" section on page 309 in the "Troubleshooting Quality of Service for VoIP" chapter.

Voice DSP Crash Dump File Analysis

The Voice Crash Dump File Analysis feature allows Cisco IOS voice platforms using Texas Instruments DSPs the ability to capture the contents of the DSP memory into a file in the event of a DSP crash. By making this crash dump file available for offline analysis, engineers can better and more quickly determine and fix the cause of the crash.

DSP crash dump analysis allows to you do the following:

- Detect when control messages have been lost between Cisco IOS software and the DSP
- Detect when the DSP has crashed
- Collect an image of the DSP memory after a DSP crash and put it into a file for analysis later by an engineer

When these events have been detected, they are announced by console alarms. You can enable and disable this feature and specify where the crash dump is to be written using Cisco IOS command-line interface (CLI). The active part of the stack is written to the console, while the entire contents of the DSP memory is written to the crash dump file. You can request that a dump file be written into a "smart" slot 0 or slot 1 flash card, or sent to a server using TFTP or FTP, or it may be written directly to Flash.

How to Configure Voice DSP Crash Dump File Analysis

To configure Voice DSP crash dump file analysis, use the following steps:

SUMMARY STEPS

1. **enable**
2. **configure {terminal** | **memory** | **network}**
3. **voice dsp crash-dump destination** *url*
4. **voice dsp crash-dump file-limit** *limit-number*
5. **exit**
6. **show voice dsp crash-dump**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables higher privilege levels, such as privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure {terminal | memory | network}`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

	Command or Action	Purpose
Step 3	**voice dsp crash-dump destination** *url* **Example:** `Router(config)# voice dsp crash-dump destination 175.101.122`	(Required) Designates a valid file system where crash dump analysis is stored. • The *url* argument must be set to a valid file system. • The destination URL can be one of the following – The file on a TFTP server with the following format: *tftp://x.x.x.x/subfolder/filename.* *The* x.x.x.x value is the IP address of the TFTP server – The file on the flashcard of the router, with the following format: *slot0:filename* **Note** The DSP crash dump feature is disabled when the crash-dump destination is not specified.
Step 4	**voice dsp crash-dump file-limit** *limit-number* **Example:** `Router(config)# voice dsp crash-dump file-limit 99`	(Required) Sets the number of files you would like to write. • The crash dump **file-limit** keyword must be set to a non-zero value. The default is that the crash dump capability is turned off, as the *url* argument is empty, and the *file-number* argument is zero. • The *limit-number* argument can range from 0 to 99. **Note** The DSP crash dump feature is disabled when the crash-dump file limit is set to 0.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits to privileged EXEC mode.
Step 6	**show voice dsp crash-dump** **Example:** `Router# show voice dsp crash-dump`	(Optional) Displays voice DSP crash dump information

Troubleshooting Voice DSP Crash Dump File Analysis

To troubleshoot the Voice DSP Crash Dump File Analysis feature, use the **debug voice dsp crash-dump** command in privileged EXEC mode. This command is intended only for troubleshooting purposes because the volume of output generated by the software can result in severe performance degradation on the router.

SUMMARY STEPS

1. **enable**
2. **debug voice dsp crash-dump keepalive**
3. **undebug all**
4. **debug voice dsp crash detail**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug voice dsp crash-dump keepalives** **Example:** `Router(config)# no logging console`	Displays debugging information for the crash dump feature keepalives. • Confirms that a crash dump file has been written to the specified destination.
Step 3	**undebug all** **Example:** `Router# un all`	Disables all the debug output on screen to stop the above output. • Alternately, you can use the **no debug all** command.
Step 4	**debug voice dsp crash detail** **Example:** `Router# debug voice dsp crash detail`	Displays debugging information for the crash dump feature details. • There is no debug output until there is one DSP crash. When the crash dump feature is turned on, the detailed debug messages are displayed.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits to privileged EXEC mode.

Verifying DSP Crash Dump File Analysis

To verify crash dump statistics, use the **show voice dsp crash-dump** command in privileged EXEC mode.

Configuration Examples for Voice DSP Crash Dump File Analysis

The following example shows that crash dump analysis is enabled:

```
Router(config)# voice dsp crash-dump destination 172.29.248.12
Router(config)# voice dsp crash-dump file-limit 10

voice dsp crash-dump destination tftp://172.29.248.12/tester/crash-152-t
voice dsp crash-dump file-limit 10
end
1w0d:%SYS-5-CONFIG_I:Configured from console by consoleoice dsp crash
```

Verifying Voice DSP Crash Dump File Analysis

The following example shows output information that verifies the status of the crash dump:

```
Router# show voice dsp crash-dump

Voice DSP Crash-dump status:
    Destination file url is
```

```
      tftp://172.29.248.12/tester/crash-152-t
File limit is 10
Last DSP dump file written was
      tftp://172.29.248.12/zongshan/crash/26-152-t2
Next DSP dump file written will be
      tftp://172.29.248.12/tester/crash-152-t1
```

Troubleshooting Universal Port SPEs

A universal port card is a hardware card that processes digital signals for the Cisco AS5350 and Cisco AS5400 universal gateways. The service-processing element (SPE) works as a DSP for the universal port card.

This section provides troubleshooting information that apply to modems regardless of service type mode. It describes how to perform diagnostic tests on installed ports or SPEs, configure automatic recovery of ports on an SPE, and configure a scheduled recovery of SPEs.

Configure SPE Diagnostic Tests

You can perform three types of diagnostic tests on the SPE modem:

- SPE Startup Test, page 13
- SPE Auto-Test, page 13
- SPE Back-to-Back Test, page 14

SPE Startup Test

To perform diagnostic testing on all the installed SPE ports during the system's initial startup or rebooting process, use the**port modem startup-test** command in global configuration mode.

The results of the SPE port startup test are displayed in the **show port modem test** command output. SPE ports that pass the diagnostic test are marked as *Pass*, *Fail*, and *Unkn*. Ports that fail the diagnostic test are marked as *Bad*. These ports cannot be used for call connections. Depending on how many ports are installed, this diagnostic test may take from 5 to 10 minutes to complete. Perform additional testing on an inoperative SPE port by executing the **test port modem back-to-back** command. The **no port modem startup-test** command disables startup testing.

SPE Auto-Test

To perform diagnostic testing on all the installed SPE ports during the system’s initial startup or rebooting process, or during service, use the **port modem autotest** command in global configuration mode.

The results of the SPE port auto-test are displayed in the **show port modem test** command's output. Ports that pass the diagnostic test are marked as *Idle*, *Busy*, *Downloading*, and *Reset*, and are put into service. Ports that fail the diagnostic test are marked as *Bad*, and are not put into service or tested again until they are no longer marked as *Bad*. If all the ports of an SPE are bad, the corresponding SPE is also marked bad. These ports cannot be used for call connections. Depending on how many ports are present and not marked *Bad*, this diagnostic test may take from 5 to 10 minutes to complete. You may perform additional testing on an inoperative port by executing the **test port modem back-to-back** command. The **no port modem autotest** command disables testing.

You may optionally configure the following commands:

- **port modem autotest minimum** *ports*—Define the minimum number of free ports available for autotest to begin.
- **port modem autotest time** *hh:mm {interval}*—Enable autotesting time and interval.A sample diagnostic autotest setting the time at 12:45 and at 8 hour intervals looks like the following:

```
AS5400(config)# port modem autotest time 12:45 8
AS5400(config)#
```

- **port modem autotest error** *threshold*—Define the maximum number of errors detected for autotest to begin.

SPE Back-to-Back Test

When an SPE port tests as *Bad*, perform additional testing by conducting a series of internal back-to-back connections and data transfers between two SPE ports. All port test connections occur inside the gateway. For example, if mobile users cannot dial into port 2/5 (the sixth port on the universal port card in the second chassis slot), attempt a back-to-back test with port 2/5 and a known-functioning port such as port 2/6.

Enter the following command in privileged EXEC mode (the prompt is displayed as `AS5350#` or `AS5400#`) to perform internal back-to-back port tests between two ports:

test port modem back-to-back *slot/port slot/port* {*num-packets*}—Perform internal back-to-back port tests between two ports, sending test packets of the specified size.

You might need to enable this command on several different combinations of ports to determine which one is not functioning properly. A pair of operable ports successfully connect and complete transmitting data in both directions. An operable port and an inoperable port do not successfully connect with each other.

A sample back-to-back test might look like the following:

```
AS5400# test port modem back-to-back 2/10 3/20

Repetitions (of 10-byte packets) [1]:
*Mar  02 12:13:51.743:%PM_MODEM_MAINT-5-B2BCONNECT:Modems (2/10) and (3/20) connected in
back-to-back test:CONNECT33600/V34/LAP
*Mar  02 12:13:52.783:%PM_MODEM_MAINT-5-B2BMODEMS:Modems (3/20) and (2/10) completed
back-to-back test:success/packets = 2/2
```

A port that has been confirmed to have problems can often be fixed using the **clear spe** command.

The results of the **test port modem back-to-back** command are displayed in the **show port modem test** command's output:

```
AS5400# show port modem test

 Date Time             Modem  Test              Reason            State Result
 3/02 12:00:57 PM       2/01  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:00:57 PM       2/00  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:00:58 PM       2/02  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:00:58 PM       2/03  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:00:58 PM       2/04  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:00:58 PM       2/05  Back-To-Back     :STARTUP TEST      Idle  PASS
...
 3/02 12:01:14 PM       3/95  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:01:14 PM       3/94  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:01:15 PM       3/75  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:01:15 PM       3/74  Back-To-Back     :STARTUP TEST      Idle  PASS
 3/02 12:13:52 PM       3/20  Back-To-Back     :USER INITIATED    Idle  PASS
```

```
 3/02 12:13:52 PM       2/10  Back-To-Back     :USER INITIATED    Idle  PASS
...
 3/02 12:44:00 PM      3/102  No Test (Time)   :MIN IDLE MODEMS   Idle  NOTST
 3/02 12:44:00 PM      3/103  No Test (Time)   :MIN IDLE MODEMS   Idle  NOTST
 3/02 12:44:00 PM      3/104  No Test (Time)   :MIN IDLE MODEMS   Idle  NOTST
 3/02 12:44:00 PM      3/105  No Test (Time)   :MIN IDLE MODEMS   Idle  NOTST
 3/02 12:44:00 PM      3/106  No Test (Time)   :MIN IDLE MODEMS   Idle  NOTST
 3/02 12:44:00 PM      3/107  No Test (Time)   :MIN IDLE MODEMS   Idle  NOTST
 3/02 12:44:21 PM       2/73  Back-To-Back     :TIME INTERVAL     Idle  PASS
 3/02 12:44:21 PM       2/72  Back-To-Back     :TIME INTERVAL     Idle  PASS
 3/02 12:44:21 PM       2/33  Back-To-Back     :TIME INTERVAL     Idle  PASS
 3/02 12:44:21 PM       2/32  Back-To-Back     :TIME INTERVAL     Idle  PASS
 3/02 12:44:21 PM       3/37  Back-To-Back     :TIME INTERVAL     Idle  PASS
```

Note The *Reason* column indicates why the test was started. The *TIME INTERVAL* is one of the triggers under autotest; the other is the error threshold.

SPE Disconnect Reason Codes

This section describes how to interpret the call disconnect reason codes reported by Cisco universal port card SPEs. Whenever a call using the universal port SPEs is cleared or disconnected, the SPE records the reason for the disconnect. This disconnect reason code can be used to determine whether the disconnect was normal or an error occurred. This reason code can be used to track down possible sources of failure. Modems can be disconnected due to a variety of factors such as client disconnects, telco errors, and call drops at the network access server (NAS). A “good” disconnect reason is that the DTE (client modem or NAS) at one end or the other wanted to terminate the call. Such “normal” disconnects indicate that the disconnect was not a result of modem or transmission level errors.

Note The disconnect reason is managed in a first-come-first-serve fashion. This means that the first disconnect reason generated is the only disconnect reason recorded. If the modem and the NAS attempt to terminate the session simultaneously and the modem happens to save the disconnect reason before the LINK_TERMINATE message from the NAS is processed, then the NAS disconnect reason is ignored.

Determining the disconnect Reason

When evaluating whether you are experiencing “good” or “bad” disconnects, it is important to obtain the history of disconnects that a particular port has experienced. In most environments, the disconnect reason is obtained using modem call records or call tracker syslog messages. This disconnect code can then be interpreted using the table provided in this document. Use the following commands to determine the disconnect reason:

- The **show spe modem disconnect-reason** command does not display the disconnect reason code as a hexadecimal value. However, it does indicate the disconnect reason as a name. The name and class of the disconnect reason can be found inTable 35 and Table 36 respectively.
- The **show port modem log** command displays the disconnect reason code as a hexadecimal value. Refer to Table 34 for the hexadecimal values.

Table 34 Disconnect Reason Code Hexadecimal Values

0x0..		0x1..			0x2..			0x3..	0x4..	0x5..
	0x010	0x100	0x1F00	-	-	0x210	0x220	0x3xx	-	-
0x001	0x011	0x101	0x1F01	-	0x201	0x211	0x221	-	-	0x501
0x002	0x012	0x102	0x1F02	-	0x202	0x212	0x222	-	-	0x502
0x003	-	0x103	0x1F03	-	0x203	-	-	-	0x403	0x503
0x004	-	0x104	0x1F04	-	0x204	-	0x224	-	0x404	0x504
0x005	-	0x105	0x1F05	-	0x205	-	0x225	-	-	0x505
0x006	-	0x106	0x1F06	-	0x206	-	-	-	-	0x506
0x007	-	0x107	0x1F07	-	-	-	-	-	-	0x507
0x008	-	0x108	0x1F08	-	-	-	-	-	0x408	-
0x009	-	0x109	-	-	-	-	-	-	-	-
0x00C	-	-	-	-	-	-	-	-	-	-
0x00D	-	-	-	-	-	-	-	-	-	-
0x00E	-	-	-	-	-	-	-	-	-	-
0x00F	-	-	-	0x1FFF	-	-	-	-	-	-

Using the show port modem log command

Use the **show port modem log** *slot/port* command to obtain the disconnect cause code (in Hex) for a particular call on a specific port. This disconnect code is identical to the cause code obtained from modem call-record and call-tracker syslog outputs. An example is shown:

```
*Jan  1 00:53:56.867: Modem State event: State: Terminate
*Jan  1 00:53:56.879: Modem End Connect event:
  Call Timer                              :   195  secs
  Disconnect Reason Info                  :   0x220
      Type (=0  ):
     Class (=2  ):  EC condition - locally detected
    Reason (=32 ):  received DISC frame -- normal LAPM termination
```

From the example above, note that the disconnect code is **0x220**.

Using the show spe modem disconnect-reason Command

Use the **show spe modem disconnect-reason** {**summary** | *slot* | *slot/spe*} command to determine the distribution of disconnect reasons that the particular port has experienced. A sample summary output of all the ports is shown below:

```
Router# show spe modem disconnect-reason summary
     ===CLASS OTHER====  =====CLASS DSP====  ===CLASS EC LCL===  ==CLASS EC FRMR===
     Software Rst     0  No Carrier     341  No LR            0  Frmr Bad Cmd     0
     EC Termntd       0  No ABT dtctd     0  LR Param1        0  Frmr Data        0
     Bad MNP5 Rx      0  Trainup flr    328  LR Incmpt        0  Frmr Length      0
     Bad V42B       110  Retrain Lt       0  Retrns Lt      226  Frmr Bad NR      0
     Bad COP stat     0  ABT end flr      0  Inactivity       0
     ATH              0                      Protocol Err     1  ===CLASS EC LD====
     Aborted          0  ====CLASS HOST====  Fallbck Term    74  LD No LR         0
     Connect Tout   198  Hst NonSpec      0  No XID          67  LD LR Param1     0
     Reset DSP        0  HST Busy         0  XID Incmpt       0  LD LR Incmpt     0
```

```
                      HST No answr     0  Disc         21448  LD Retrns Lt     0
 ===CLASS EC Cmd===   HST DTR       3615  DM               5  LD Inactivty     0
 Bad Cmd          0   HST ATH          0  Bad NR           0  LD Protocol      0
                      HST NoDialTn     0  SABME Online     0  LD User          0
 =====N O N E======   HST No Carr   5276  XID Online       0
    None          39  HST Ack          0  LR Online        0  TOTAL       31728
HST NoDialTn     0  SABME Online     0  LD User          0=====N O N E======  HST No
Carr   5276  XID Online       0  None            39  HST Ack          0  LR Online
0  TOTAL        31728
```

From the example above, let us say that we are interested in the disconnect category Disc within **CLASS EC LCL**. To determine what the disconnect reason Disc means, go to the entry corresponding to the class (CLASS EC LCL) and the disconnect reason name (Disc) which shows a hex code of 0x220 and is a normal disconnect. The codes for the classes are shown in the following tables:

- CLASS OTHER Code Summary Table
- CLASS DSP Reason Codes
- CLASS EC LCL: EC Condition, Locally Detected Reason Code Table
- CLASS EC Cmd: EC Detected Bad Command Code Reason Code Table
- CLASS EC FRMR: EC Detected FRMR From Peer Reason Code Table
- CLASS EC LD: Error Correction (EC) Detected Link Disconnect (LD) From Peer Reason Code Table
- CLASS HOST: Requested by Host Reason Code Table

Reason Code Summary Tables

The following tables contain the detailed reason codes for universal port disconnects.

Table 35 CLASS OTHER Code Summary Table

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
2	Software Rst	0x001	Cisco IOS software disconnected the call for some indeterminate reason (SOFTWARE_RESET).
2	EC Termntd	0x002	Error Correction (EC) layer termination
2	Bad MNP5 Rx	0x003	The Microcom Network Protocol 5 (MNP5) decompression task received an illegal token in the data stream. There is probably a logic error in the implementation of compression, decompression or error correction by the modem or partner. Can also be caused by a transient line or RAM memory error.
2	Bad V42B	0x004	The V.42bis or V.44 decompression task received an illegal token in the data stream. There is probably a logic error in either the modem's or partner's implementation of compression, decompression or error correction. Can also be caused by a transient line or RAM memory error.
2	Bad COP stat	0x005	Reserved

Table 35 CLASS OTHER Code Summary Table (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
6,7	ATH	0x006	ATH command detected by local modem. The ATH (Hangup) AT command is detected by the local modem (universal port card). For example, following a dialout from Cisco IOS, the DTE interface clears the call by transmitting an inband ATH AT command after the call is connected.
3	Aborted	0x007	AT mode any-key abort of dial command The AT dial command was aborted by the any-key abort command. For example, the host modem originates a call. During connection establishment, pressing any-key causes the AT dial command to be aborted.
3	Connect Tout	0x008	The call took too long to complete the connection. Notice that the S7 timer (wait for carrier after dial) expired for this disconnect. The causes include: • Difficulty negotiating a Layer I standard • Layer I and Layer II establishment taking too long. For example, error correction negotiation takes an extended amount of time because of bit-errors introduced when the client modem receiver tries to connect at a rate it can't sustain. This disconnect could also happen if the answer modem heard no tone from the channel because, for example, the originator was not a modem.
2	Reset DSP	0x009	The DSP was reset (command/internal/spontaneous). The DSP within the host modem was reset by the Control Processor (CP) or Signal Processor (SP). The CP resets the DSP if mail messages from the CP to SP are not being acknowledged. The SP resets itself if it gets an internal inconsistency error.
4,6	—	0x00C	V.42bis or V.44 codeword size exceeded negotiated maximum.
4,6	—	0x00D	V.42bis or V.44 received codeword equal to next empty dictionary entry.
4,6	—	0x00E	V.42bis or V.44 received codeword greater than the next empty dictionary entry.
4,6	—	0x00F	V.42bis or V.44 received reserved command code.
4,6	—	0x010	V.42bis or V.44 ordinal size exceeded eight.
4,6	—	0x011	V.42bis or V.44 negotiation error.
4,6	—	0x012	V.42bis or V.44 compression error.

Table 36 CLASS DSP Reason Codes

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
—	—	0x1xx	DSP conditions reported by SPE
4,5	No Carrier	0x100	The SPE carrier signal is lost. The universal port card detected a client modem carrier drop. The universal port SPE stopped hearing carrier for a period greater than the value specified in Register S10 (hang-up delay after carrier loss). This could mean that the talk path went away or that the client stopped transmitting. If a layer II protocol (V.42 and/or V.42bis) is in effect, it is abnormal to see such a disconnect. Common causes are users hanging up the call before a connection takes place can occur because of incidental dialing, aborted starts, and client applications timing out when calls take too long to connect due to multiple retrains during Layer 1 negotiation. The condition can also occur during normal data mode when the client abruptly drops the carrier. This can occur if the link is abruptly dropped (network error), or power is shut off to the client modem disconnecting the call. This can also occur with less sophisticated client modems that do not implement the Layer I and/or Layer II clear-down protocols on a DTR drop. For a large number of client modems, this is considered a normal disconnection.
3	No ABT dtctd	0x101	No answer-back tone detected, caller is probably not a modem
3	Trainup flrv	0x102	Call failure while modem training up due to incompatible modulation or bad line. This may be indicative of attempts to negotiate an unsupported modulation such as a legacy Rockwell proprietary modulation (K56Plus, V.FC, and so on). Other possible causes are DSP failures to train up due to severe line impairments, impulse noises, interrupting training, incompatible modulation parameters, and perhaps the inability to properly select a Layer I standard.

Table 36 CLASS DSP Reason Codes (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4,5	Retrain Lt	0x103	Too many consecutive retrains or speed-shifts. The retrain limit is specified with Register S40. During the progress of a call, too many retrains occurred and rendered the call ineffective—the data rate would be so poor as to be useless. Sometimes the client modem does not complete the clear-down protocol, for example, when the Telco tore down the call in the middle of the connection; NextPort (NP) attempts to recover the call by issuing retrains. Once the retrain limit is reached, NP drops the call and report this disconnect reason.
3	ABT end flr	0x104	Problem detecting end of Answer-Back Tone(ABT). Negotiation failure or excessive noise during V.34 training. Host modems answer and send out V.8bis and modulated 2100Hz answer-back tones (ABTs) with phase reversals but encounter excessive noise during the trainup sequence. Look for errors on the path from the calling modem to the answering modem in either one or both directions. Similar behavior occurs when there is latency in the Public Switched Telephone Network (PSTN)that exceeds one second for dial up and causes modems to be unable to train up the echo cancellers. Other possible causes are: • TX power levels are incorrect and the tones are then not handled by the remote side. • Excessive noise in Phase III and IV during V.34 training. • Operator error. • Network interference during V.34 training (someone picks up the extension).
3	—	0x105	SS7/COT (continuity test) operation completed successfully.
3	—	0x106	SS7/COT (continuity test) operation failed: T8/T24 timeout waiting for tone on.
3	—	0x107	SS7/COT (continuity test) operation failed: T8/T24 timeout waiting for tone off.

Table 36 CLASS DSP Reason Codes (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4	—	0x108	Modem on hold (MOH) cleardown by the universal port card. V.92 specifies that the cleardown reason can be: • Cleardown due to incoming call • Cleardown due to outgoing call • Cleardown due to other reason
4	—	0x109	MOH timeout value reached. This value can be adjusted using Register S62 (V.92 maximum MOH time).

Table 37 CLASS EC LCL: EC Condition, Locally Detected Reason Code Table

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
—	—	0x2xx	Local error correction (EC) conditions.
3	No LR	0x201	During negotiation a link request (LR) frame was not received. Peer may not support MNP.
3	LR Param1	0x202	The received MNP LR frame had a bad/unexpected PARAM1. For more information on PARAM1 refer to the V.42 specification.
3	LR Incmpt	0x203	The received MNP LR frame is incompatible with the host modem's settings for EC.

Table 37 CLASS EC LCL: EC Condition, Locally Detected Reason Code Table (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4,5	Retrns Lt	0x204	Too many consecutive retransmissions in EC. This disconnect reason can be caused by noise on the line that spurs retransmissions. For instance, the host modem transmits data to the client modem, but noise on the line causes the data to be received incorrectly (or not at all) by the client side. The client modem could also have disconnected without the host modem realizing this. So the host modem continuously retransmits, without knowing that the client modem is no longer present. Sometimes, when the call connects in LAPM or MNP, the universal port card is unable to transmit a frame to the client modem. The client modem fails to acknowledge the universal port card's initial transmission, then fails to respond to Register S19 (error correction retransmission limit) polls (the default is 12), so NP disconnects the call. One cause could be that the carrier in the transmit path degraded substantially while the client failed to downshift. Another cause could be a problem with the client's EC engine (as happens on a Winmodem system if Windows stops responding).
6,7	Inactivity	0x205	Inactivity timeout, MNP Link Disconnect (LD) sent. The host modem sends the client modem a LD frame, indicating that an inactivity timeout has occurred.
4,5	Protocol Err	0x206	EC protocol error. This is a general catch-all protocol error. It indicates that a LAPM or MNP EC protocol error has occurred.
3	Fallbck Term	0x210	No EC fallback protocol available. Error correction negotiation has not been successful. The call is terminated because there is no error correction fallback protocol available. S-register S25 (link protocol fallback) determines the available fallback protocol. The options are asynchronous framing, synchronous framing, or disconnect (hang up).

Table 37 CLASS EC LCL: EC Condition, Locally Detected Reason Code Table (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
3	No XID	0x211	Never received eXchange IDentification (XID) frame during negotiation. Peer may not support MNP.
3	XID Incmpt	0x212	The received XID frame is incompatible with local settings. The client modem may not support LAPM within V.42.
3,4,5	Disc	0x220	Received Disconnect (DISC) frame. This is the normal LAP-M disconnect. The call terminated normally with a proper clear down from the client side. For example, a V.42 disconnect packet was sent from the client modem to the host modem. The client modem dropped DTR and cleanly negotiated a clear-down protocol.
3,4,5	DM	0x221	Received DM frame. Peer might be disconnecting. The client modem indicates that it is disconnecting. During call setup, this reason indicates that the client modem is giving up on negotiating error correction.
4,5	Bad NR	0x222	Bad receive sequence number or ACK number was received. An MNP LD or LAP-M FRMR is sent. The host modem received a LAPM or MNP error correction frame with a bad sequence number or acknowledgment number. An LD or Frame Reject (FRMR) frame is sent to the client modem, indicating that the host modem is disconnecting.
4,5	SABME Online	0x224	Received MNP XID frame in steady-state. This is interpreted as a LAPM error correction protocol error in steady state. It means that the client modem may have reset due to receiving a FRMR.
4,5	XID Online	0x225	Received MNP LR frame while in steady-state. This is interpreted as an MNP error correction protocol error in steady state. It means that the client modem has reset.

Table 38 CLASS EC Cmd: EC Detected Bad Command Code Reason Code Table

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4,5	Bad Cmd	0x3xx	EC detected bad command code. The received unknown command is in the last 2 digits. An MNP LD or LAP-M FRMR frame is sent in response.

Table 39 CLASS EC FRMR: EC Detected FRMR From Peer Reason Code Table

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4,5	—	0x4xx	EC conditions indicated by client in LAP-M FRMR frame. The bit-mapped reason is in the last two digits.
4,5	Frmr Bad Cmd	0x401	LAPM: peer reports bad command. The host modem received a FRMR frame from the client modem. The received FRMR frame indicates that the client modem received an error correction frame from the host modem that contained a bad command.
4,5	Frmr Data	0x403	LAPM: peer reports that data field is not permitted or is incorrect length (U frames). The host modem received a FRMR frame from the client modem. The received FRMR frame indicates that the client modem received an error correction frame from the host modem that contained a data field that is not permitted or contained a data field with an incorrect length (that is, U frame).

Table 39 CLASS EC FRMR: EC Detected FRMR From Peer Reason Code Table (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4,5	Frmr Length	0x404	LAPM: peer reports data field length is greater than N401 (the maximum information field length specified in V.42), but has good Frame Check Sequence (FCS). The modem received a FRMR frame from the client modem. The received FRMR frame indicates that the client modem received an error correction frame from the modem that contained a data field length that is greater than the maximum number of octets that can be carried in the information field (N401) of an I frame, an SREJ frame, an XID frame, a UI frame, or a TEST frame. The frame check sequence is good.
4,5	Frmr Bad NR	0x408	LAPM: peer reports bad receive sequence number or N(R). The host modem received a FRMR frame from the client modem. The received FRMR frame indicates that the client modem received an error correction frame from the host modem that contained a bad receive sequence number.

Table 40 CLASS EC LD: Error Correction (EC) Detected Link Disconnect (LD) From Peer Reason Code Table

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
4,5	—	0x5xx	EC conditions indicated by client in MNP link disconnect (LD) frame. Reason field is in the last 2 digits.
3	LD No LR	0x501	MNP: peer never received LR frame. The host modem received a LD frame from the client modem. The received LD frame indicates that the client modem never received a link request from the host modem.
3	LD LR Param1	0x502	MNP: peer reports link request (LR) frame has bad parameter #1 The host modem received an LD frame from the client modem. The received LD frame indicates that the client modem received a link request frame from the host modem that contained an unexpected PARAM1. For more information on PARAM1 refer to the V.42 specification.

Table 40 CLASS EC LD: Error Correction (EC) Detected Link Disconnect (LD) From Peer Reason Code Table (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
3	LD LR Incmpt	0x503	MNP: peer reports LR frame is incompatible with its configuration The host modem received an LD frame from the client modem. The received LD frame indicates that the client modem received an LR frame from the host modem that is incompatible with the configuration of the client modem.
4,5	LD Retrns Lt	0x504	MNP: peer reports too many consecutive EC retransmissions The host modem received a LD frame from the client modem. The received LD frame indicates that the client modem received too many consecutive retransmissions.
4,5	LD Inactivty	0x505	MNP: peer reports inactivity timer expired The host modem received a Link Disconnect (LD) frame from the client modem. The received LD frame indicates that the client modem's host (DTE) has not passed data to the client modem within a period of time.
3	LD Protocol	0x506	MNP: peer reports error The host modem received an LD frame from the client modem. The received LD frame indicates that the client modem received a MNP protocol error.
3	LD User	0x507	Normal MNP disconnect The host modem received a LD frame from the client modem. The received LD frame indicates a normal MNP termination.

Table 41 CLASS HOST: Requested by Host Reason Code Table

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
6,7	—	0x1Fxx	Host initiated disconnect. Value is a sum of 0x1F00 and SessionStopCommand value. This is the other host terminate reason. The host reason is indicated in the low-order bytes “xx”.
3,6,7	HST NonSpec	0x1F00	Non-specific host-initiated disconnect. Value is a sum of 0x1F00 and SessionStopCommand value. This is the catch all Cisco IOS-initiated disconnect reason. It is used for all non-standard disconnects. For example, this could be a result of modem management software deciding to terminate the call. One possible explanation is a higher-level authentication failure RADIUS, TACACS, or another application issuing a DTR drop to the host modem. This type of disconnect does not count towards CSR when the host modem is in data mode.
3	HST Busy	0x1F01	Dialed number was busy. Disconnection has occurred because the host is indicating that the dialed number is busy.
3	HST No answr	0x1F02	Dialed number did not answer. Disconnection has occurred because the host is indicating that the dialed number didn't answer.
3,6,7	HST DTR	0x1F03	Virtual DTR dropped. This status is reflected by the I/O port redirector that is currently using the modem. Disconnection has occurred because the host dropped the virtual DTR line. This generic disconnect cause is initiated by the Cisco IOS software. Example causes are idle timeout, PPP LCP TERMREQ received, authentication failure, Telnet hangup, and so on. To determine the reason for the hang up, examine the Radius disconnect reason from the modem call-record terse command or from Authentication, Authorization, and Accounting (AAA).
6,7	HST ATH	0x1F04	ATH (hangup) command was detected by local host.
3	HST NoDialTn	0x1F05	No access to telco network. Disconnection has occurred because the host could not access the network.

Table 41 CLASS HOST: Requested by Host Reason Code Table (continued)

Disconnect Reason Type	Disconnect Reason: Name	Disconnect Reason Code (Hex)	Description
3,4,5	HST NoCarr	0x1F06	Network indicated disconnect. This is a client-side triggered disconnect that is not a graceful call termination. It can occur during call set-up. A common cause is when users of Windows 95 or Windows 98 Dial Up Networking (DUN) cancel the call before the call reaches steady state. Another common reason is any client-instigated DTR drop before steady state. During data mode, this is also a client side triggered disconnection that is not a graceful call termination. One very common cause is authentication failures.
3	—	0x1F07	NAS terminated SS7/continuity test (COT) operation. Disconnection has occurred because the NAS has terminated the SS7/COT operation.
3	—	0x1F08	The SS7/COT operation was terminated by the router because of a T8/T24 timeout.
-	—	0x1FFF	Unsolicited TERMINATING. The host sends this disconnect reason when it receives a unsolicited terminating message.

Table 42 Disconnect Reason Types

Disconnect Type	Description
0	(unused)
1 - 0x2...	(unused)
2 - 0x4...	Other situations
3 - 0x6...	Condition occurred during call setup
4 - 0x8...	In data mode. Rx (line to host) data flushing OK
5 - 0xA...	In data mode. Rx (line to host) data flushing not OK (at present, applications should not be concerned about the “not OK”)
6 - 0xC...	In data mode. Tx (host to line) data flushing OK
7 - 0xE...	In data mode. Tx (host to line) data flushing not OK (at present, applications should not be concerned about the “not OK”)

For more information about troubleshooting SPEs, refer to *Interpreting NextPort Disconnect Reason Codes,* document ID 9502.

DSP Troubleshooting Links

DSP troubleshooting procedures vary from platform to platform. To troubleshoot DSPs on your Cisco product, see the following links:

- For troubleshooting the DSP on NM-HDV for Cisco 2600 series, Cisco 3600 series, and VG200 series routers, refer to *Troubleshooting the DSP on NM-HDV for Cisco 2600/3600/VG200 Series Routers,* document ID 19066.
- For troubleshooting DSPs on the PA-VXA/PA-VXB/PA-VXC voice port adapters for Cisco 7200 series and Cisco 7500 series routers, refer to *Troubleshooting DSPs on the PA-VXA/PA-VXB/PA-VXC Voice Port Adaptors for Cisco 7200/7500 Series Routers,* document ID 26367.
- For troubleshooting the VTSP-3-DSP_TIMEOUT error on AS5300 platforms, refer to *Troubleshooting VTSP-3-DSP_TIMEOUT Error on Cisco AS5300 Access Server Platforms,* document ID 18680.
- If you have problems with unrecognized voice cards on Cisco 1750, Cisco 1751, and Cisco 1760 routers, it could be a problem with the packet voice data module (PVDM), which houses the DSPs. Refer to *Troubleshooting Unrecognized Voice Interface Cards on Cisco 1750, 1751, and 1760 Routers,* document ID 5711.

Verifying Codec Complexity

Codec complexity refers to the amount of processing power that a codec compression technique requires: some require more processing power than others. Codec complexity affects call density, which is the number of calls that can take place on the DSP interfaces. The DSP interfaces can be HCMs, port adapter DSP farms, or voice cards, depending on the type of router. The greater the codec complexity, the fewer the calls that can be handled.

Codec complexity is either medium or high. The difference between medium- and high-complexity codecs is the amount of CPU power necessary to process the algorithm and, therefore, the number of voice channels that can be supported by a single DSP. All medium-complexity codecs can also be run in high-complexity mode, but in that mode fewer (usually half as many) channels are available per DSP.

For details on the number of calls that can be handled simultaneously through the use of each of the codec standards, refer to the entries for the **codec** and **codec complexity** commands in the *Cisco IOS Voice Command Reference*.

Codec Complexity Mismatch

You might encounter a situation in which the router cannot set up a call and a message similar to the following appears in the output:

```
21:12:54: %DSPRM-5-SETCODEC: Configured codec 10 is not supported with this dsp image.
```

This condition indicates that the codec complexity and the voice card complexity configuration are mismatched. This problem can appear on Cisco modular access routers with HDV modules. This problem can affect Cisco IOS software Releases 12.0(7)T and later.

To see if you have this problem, you need to check the following conditions:

- Check if the codec you are using is a high-complexity codec. For more information about codecs, refer to *Understanding Codecs: Complexity, Hardware Support, MOS, and Negotiation*, document ID 14069.
- If you are going to use high-complexity codecs, check the voice card configuration. It should also be configured as high complexity.

The default configuration for voice cards on routers with HDV modules is medium complexity. To allow usage of high-complexity codecs, use the **voice-card 1** and **codec complexity high** configuration commands.

Note To change the voice card codec complexity, remove all voice ports bound to the card and remove the configuration from the E1/T1 controller.

Checking the Interface

To troubleshoot the T1 or E1 interface, perform the following steps:

SUMMARY STEPS

1. **show controller** {t1 | e1}
2. Check if the line is down.
3. Check for reported alarms.
4. Check for error events.
5. Check if the interface is T1 or E1 CAS, E1 R2, or PRI.

DETAILED STEPS

Step 1 Enter the **show controller t1** or **show controller e1** command with the controller number for the voice port you are troubleshooting.

```
Router# show controller {t1 | e1} controller-number
```

Step 2 Check if the line is down. If so, see the "Troubleshooting T1 and E1 Layer 1 Problems" section on page 31.

Step 3 Check if there are any reported alarms. If so, refer to *T1 Alarm Troubleshooting*, document ID 14172 to troubleshoot alarm indications.

Step 4 Check if there are any error events. If you encounter framing, line coding, or clock timing errors, see the "Checking T1/E1 Controller Configuration" section on page 34.

Refer to *T1 Error Event Troubleshooting*, document ID 14174 for a flowchart to troubleshoot error events.

Step 5 Check if the interface is T1 or E1 CAS, E1 R2, or PRI. See the following sections for more information:

- Checking T1/E1 Controller Configuration, page 34
- T1/E1 Channel-Associated Signaling, page 38
- E1 R2 Interfaces, page 39

Troubleshooting T1 and E1 Layer 1 Problems

Most T1 and E1 errors are caused by incorrectly configured lines. Ensure that line coding, framing, and clock source are configured according to the recommendations of your Service Provider.

Use the **show controllers t1** and **show controllers e1** commands in privileged EXEC mode to display information about the T1 or E1 links or to display the hardware and software driver information for the controller. These commands show what state the T1 or E1 controller is in. The controller can be in one of three states:

- Administratively down—If the controller is administratively down, you can manually bring it up using the procedure in the "Controller Is Administratively Down" section.
- Down—If the controller is down, then the cause is one of the following:
  - Loss of frame—See the "Controller Has Loss of Frame" section to resolve this issue.
  - Loss of signal—See the "Controller Has Loss of Signal" section to resolve this issue.
- Up—The controller is functioning properly on Layer 1.

Solutions for bringing the controller up follow.

Controller Is Administratively Down

The controller is administratively down when it has been manually shut down. Follow these steps to restart the controller to correct this error.

SUMMARY STEPS

1. **enable**
2. **configure** {**terminal** | **memory** | **network**}
3. **controller t1** *number*
4. **no shutdown**
5. **end**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
    enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router(config)# configure terminal
```

Step 3 Enter controller configuration mode:

```
controller t1 number
```

The *number* syntax is platform-specific. For more information about the syntax of this command, see the *Cisco IOS Interface and Hardware Component Command Reference*, Release 12.3.

```
Example: Router(config)# controller t1 0
```

Step 4 Restart the controller:

```
no shutdown
```

```
Example: Router(config-controller)# no shutdown
```

Step 5 Exit to privileged EXEC mode:

```
end
```

```
Example:Router(config)# end
```

What to Do Next

The controller should be running and normal configuration can continue.

Controller Has Loss of Frame

Complete the following steps if the receiver has loss of frame:

- Ensure that the framing format configured on the port matches the framing format of the line. Check the framing format of the controller from the running configuration or the s**how controller t1**or **show controller e1** command output. To change the framing format, use the **framing** command in controller configuration mode.
  - For T1, the options are **sf** and **esf**. For example:

    ```
    Router# configure terminal
    Enter configuration commands, one per line. End with CNTL/Z.
    Router(config)# controller t1 0
    Router(config-controlle)# framing esf
    ```

  - For E1, the options are **crc4** and **no-crc4**. For example:

    ```
    Router# configure terminal
    Enter configuration commands, one per line. End with CNTL/Z.
    Router(config)# controller e1 0
    Router(config-controlle)# framing crc4
    ```

  If the first framing format does not work, try the other framing format to see if the alarm clears. For more information on framing formats, see the "Framing Formats on Digital T1/E1 Voice Ports" section on page 35.

- For T1 lines, change the line build-out setting using the **cablelength long** or **cablelength short** command.

  Line build-out (LBO) or cable length compensates for the loss in decibels based on the distance from the device to the first repeater in the circuit. A longer distance from the device to the repeater requires that the signal strength on the circuit be boosted to compensate for loss over that distance.

To configure transmit and receive levels for a cable length (line build-out) longer than 655 feet for a T1 trunk with a channel service unit (CSU) interface, use the **cablelength long** controller configuration command. To configure transmit attenuation for a cable length (line build-out) of 655 feet or shorter for a T1 trunk with a DSX-1 interface, use the **cablelength shor**t controller configuration command.

Contact your service provider and refer to the *Cisco IOS Interface Configuration Guide* for details on build-out settings.

What to Do Next

If the preceding steps do not fix the problem, see the "Controller Has Loss of Signal" section.

Controller Has Loss of Signal

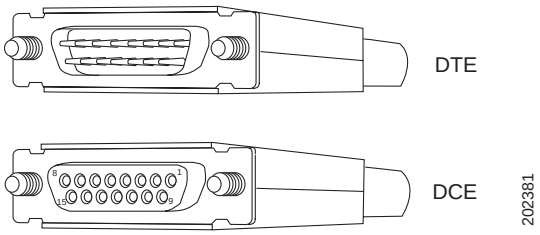
If your controller is experiencing loss of signal, check the following.

Note Use the **show controller t1** EXEC command after each step to see if the controller exhibits any errors.

- Ensure that the cable between the interface port and the T1 service provider's equipment or T1 terminal equipment is connected correctly. Ensure that the cable is hooked up to the correct ports. Correct the cable connections if necessary.
- Check the cable integrity by looking for breaks or other physical abnormalities in the cable. Ensure that the pinouts are set correctly. Replace the cable if necessary.
- Check the cable connectors. A reversal of the transmit and receive pairs or an open receive pair can cause errors. Depending on the type of module used, the cable terminates on a male DB-15 or RJ-45/48 connector.

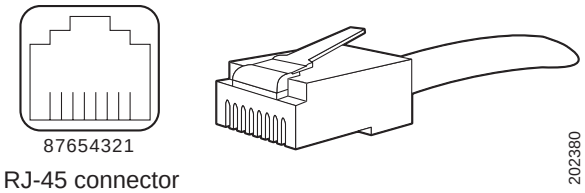
  On a DB-15 connector, the receive pair should be on pins 2 and 9, and the transmit pair on pins 8 and 15. A DB-15 connector is shown in Figure 33.

Figure 33 DB-15 Connector



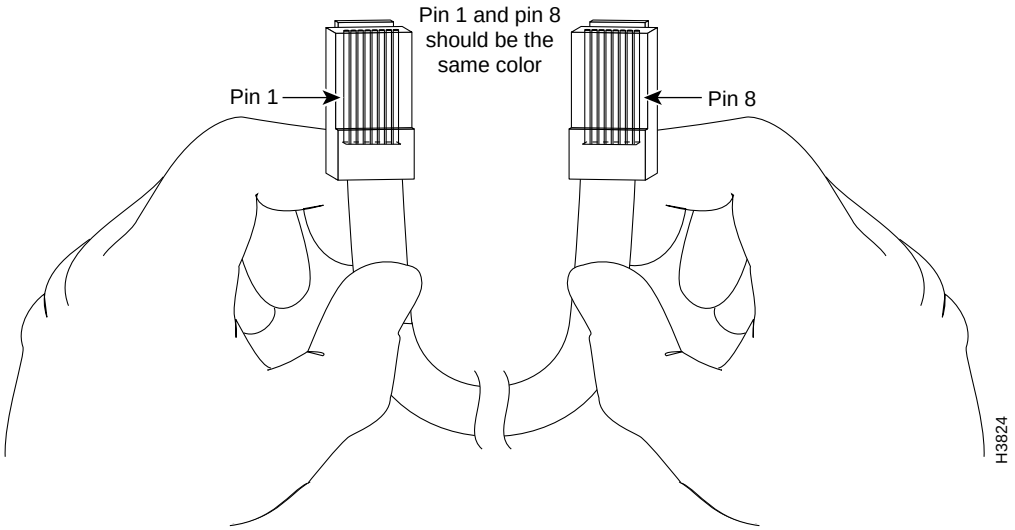
The pins on a RJ-45/48 jack are numbered from 1 through 8. With the metal pins facing toward you, pin 1 is the left-most pin. Figure 34 shows the pin numbering on an RJ-45 jack. The receive pair should be on lines 1 and 2, and the transmit pair should be on lines 4 and 5.

Figure 34 RJ-45 Pin Numbering



- If you have completed all of the steps above and you are still experiencing problems, try using a rollover cable. A rollover cable is shown in Figure 35.

Figure 35 Rollover Cable



Checking T1/E1 Controller Configuration

Digital T1/E1 voice ports must have controller configurations that match the configuration of the router to the line characteristics of the telephony network connection being made so that voice and signaling can be transferred between them. This configuration also affects allows the logical voice ports, or DS0 groups, to be established. Make sure to check the following:

- Framing Formats on Digital T1/E1 Voice Ports, page 35
- Clock Sources on Digital T1/E1 Voice Ports, page 35
- Line Coding on Digital T1/E1 Voice Ports, page 37

Contact your service provider for framing and line coding settings. Common settings are as follows:

- For T1 lines, it is common to use binary 8-zero substitution (B8ZS) line coding with extended superframe (ESF), and alternate mark inversion (AMI) line coding with superframe (SF).
- For E1 lines, both HDB3 and AMI line coding are available, but CRC4 framing is most widely used.

Framing Formats on Digital T1/E1 Voice Ports

The framing format parameter describes the way that bits are robbed from specific frames to be used for signaling purposes. The controller must be configured to use the same framing format as the line from the PBX or CO that connects to the voice port you are configuring.

Digital T1 lines use the SF or ESF framing format. SF provides two-state, continuous supervision signaling, in which a 0 bit value is used to represent on-hook and a 1 bit value is used to represent off-hook. ESF robs four bits instead of two, yet has little impact on voice quality. ESF is required for 64-kbps operation on DS0 and is recommended for Primary Rate Interface (PRI) configurations.

E1 lines can be configured for cyclic redundancy check (CRC4) or no cyclic redundancy check, with an optional argument for E1 lines in Australia.

To change the framing format, use the **framing** command in controller configuration mode. Refer to the *Voice Port Configuration* document for configuration information.

Path Code Violations Increasing

Ensure that the framing format configured on the port matches the framing format of the line. Path code violations and line code violations are typically present simultaneously. Always verify that your line coding is correct. Look for the following in the **show controller** command output:

- For E1 lines, look for "Framing is {crc4|no-crc4}"
- For T1 lines, check the line coding, as described in the "Line Coding on Digital T1/E1 Voice Ports" section on page 37. For T1 lines, path code violation error event is a frame synchronization bit error in the D4 (SF) format, or a CRC error in the ESF format.

If path code violations keep increasing, contact your service provider to check the line, because path code violations can also be caused by physical line problems.

Clock Sources on Digital T1/E1 Voice Ports

Digital T1/E1 interfaces use clock timers to ensure that voice packets are delivered and assembled properly. All interfaces handling the same packets must be configured to use the same source of timing so that packets are not lost or delivered late. The timing source that is configured can be external (from the **line**) or **internal** to the router's digital interface.

If the timing source is **internal**, timing derives from the onboard phase-lock loop (PLL) chip in the digital voice interface. If the timing source is **line** (external), timing derives from the PBX or PSTN CO to which the voice port is connected. It is generally preferable to derive timing from the PSTN because PSTN clocks are maintained at an extremely accurate level. This is the default setting for the clock source. When two or more controllers are configured, one should be designated as the primary clock source; it drives the other controllers.

To change the clock source, use the **clock source** command in controller configuration mode. Refer to the *Voice Port Configuration* document for configuration information.

Slip Seconds Error Counter Increasing

Use the **show controller** command to see if there are alarms or errors displayed by the controller. To see if the framing, line coding, and slip seconds error counters are increasing, use the **show controller e1** command repeatedly. Note the values of the counters for the current interval.

If slips are present on the line, there is a clocking problem. The customer premises equipment (CPE) needs to synchronize to the clocking from the T1/E1 service provider. Complete the following steps to correct this problem:

SUMMARY STEPS

1. **enable**
2. **show controller**
3. **configure terminal**
4. **controller** {**t1** | **e1**}
5. **clock source** {**line** [**primary** | **secondary**} | **internal**}

DETAILED STEPS

Step 1 Enter **enable** to enter privileged EXEC mode. Enter a password, if necessary.

Step 2 Ensure that the clock source is derived from the network. In the **show controller** EXEC command output, look for "Clock Source is Line Primary."

Note If there are multiple lines into an access server, only one can be the primary source. The other lines derive the clock from the primary source. If there are multiple lines, ensure that the line designated as the primary clock source is configured correctly. You can also configure a second line to provide clocking in case the primary source goes down. To do this, use the **clock source line secondary** command from controller configuration mode.

Step 3 Enter **configure terminal** to enter global configuration mode.

Step 4 Enter **controller t1** or **controller e1** to enter controller configuration mode.

Step 5 Set both the primary and secondary clock sources from controller configuration mode. For example:

```
Router(config-controlle)#clock source line primary
```

and

```
Router(config-controlle)#clock source line secondary 1
```

Ensure that the lines that you specify as the primary and secondary are both active and stable.

Note On Cisco universal gateways and access servers, the clock source is specified using the **dial-tdm-clock** command. Refer to the "Managing Dial Shelves" chapter in the *Cisco IOS Interface Configuration Guide*.

Framing Loss Seconds Increasing

Follow these instructions when dealing with a framing loss seconds increase:

SUMMARY STEPS

1. **enable**
2. **show controller**
3. **configure terminal**
4. **controller** {**t1** | **e1**}

5. **framing** {**sf** | **esf**} or **framing** {**crc4**|**no-crc4**}
6. **cablelength** {**long**|**short**}

DETAILED STEPS

Step 1 Enter **enable** to enter privileged EXEC mode. Enter a password, if necessary.

Step 2 Ensure that the framing format configured on the port matches the framing format of the line. Look for the following in the **show controller** output

- For T1 lines, look for "Framing is {ESF|SF}"
- For E1 lines, look for "Framing is {crc4|no-crc4}"

Step 3 Enter **configure terminal** to enter global configuration mode.

Step 4 Enter **controller t1** or **controller e1** to enter controller configuration mode.

Step 5 To change the framing format, use the following commands in controller configuration mode:

- For T1 lines, use **framing {sf | esf}**. For example:

```
Router(config-controlle)#framing esf
```

- For E1 lines, use **framing {crc4|no-crc4}**. For example:

```
Router>(config-controller)#framing crc4
```

Step 6 For T1 lines, change the line build-out using the **cablelength long** or **cablelength short** command.

Contact your service provider and consult the *Voice Port Configuration* document, Release 12.3 for details on settings.

Line Coding on Digital T1/E1 Voice Ports

Digital T1/E1 interfaces require that line encoding be configured to match that of the PBX or CO that is being connected to the voice port. Line encoding defines the type of framing used on the line.

T1 line encoding methods include alternate mark inversion (AMI) and binary 8 zero substitution (B8ZS). AMI is used on older T1 circuits and references signal transitions with a binary 1, or "mark." B8ZS, a more reliable method, is more popular and is recommended for PRI configurations as well. B8ZS encodes a sequence of eight zeros in a unique binary sequence to detect line-coding violations.

Supported E1 line encoding methods are AMI and high-density bipolar 3 (HDB3), which is a form of zero-suppression line coding.

Use the **show controller** command to see if there are alarms or errors displayed by the controller. To see if the framing, line coding, and slip seconds error counters are increasing, use the **show controller** command repeatedly. Note the values of the counters for the current interval.

Line Code Violations Increasing

Ensure that the line coding configured on the port matches the line coding of the line. Change the line code in controller configuration mode if necessary.

- For E1 lines, look for "Line Code is HDB3" in the **show controller e1** output.
- For T1 lines, look for "Line Code is {B8ZS|AMI}" in the **show controller t1** output.

For T1 lines, change the line build-out using the **cablelength long** or **cablelength short** command.

If line code violations keep increasing, contact your service provider to check the line, because line code violations can also be caused by physical line problems.

Path Code Violations Increasing

Ensure the framing format configured on the port matches the framing format of the line. Path code violations and line code violations are typically present simultaneously. Always verify that your line coding is correct. Look for the following in the **show controller** output:

- For T1 lines, look for "Line Code is {B8ZS|AMI}"

  For T1 lines, path code violation error event is a frame synchronization bit error in the D4 (SF) format, or a CRC error in the ESF format.

- For E1 lines, check the framing as described in the "Framing Formats on Digital T1/E1 Voice Ports" section on page 35.

For T1 lines, change the line build-out using the **cablelength long** or **cablelength short** command.

If path code violations keep increasing, contact your service provider to check the line. Path code violations can also be caused by physical line problems.

T1/E1 Channel-Associated Signaling

CAS exists in many networks today. CAS systems carry signaling information in the same channels in which voice and data are carried. Current telecommunication networks require more efficient means of signaling. CAS exists in many varieties that operate over analog and digital facilities. The analog facilities are either two- or four-wire, and the digital facilities are either North American T1 or European E1. Each CAS system uses either supervision signaling or address signaling over analog and digital facilities.

Three groups of signals are present in these facilities:

- Supervision signals represent events occurring on a trunk and can be specific to CAS. Signal types include seizure, wink, and answer.
- Address signals represent the digits dialed or called party number and, in some instances, other information. Address signals are based on multiflex signaling.
- Tone and announcement signals include ringing and busy tones and announcements specific to an event. Service circuits are used in most exchanges to send and receive address signals and tones as well as to play announcements.

This section describes CAS signaling, which is sometimes called *robbed-bit signaling* because user bandwidth is *robbed* by the network for signaling. A bit is taken from every sixth frame of voice data to communicate on- or off-hook status, wink, ground start, dialed digits, and other information about the call.

In addition to setting up and tearing down calls, CAS provides for the receipt and capture of dialed number identification (DNIS) and automatic number identification (ANI) information, which are used to support authentication and other functions. The main disadvantage of CAS signaling is its use of user bandwidth to perform these signaling functions.

For more information about troubleshooting CAS, refer to *Configuring and Troubleshooting T1 CAS Signaling*, document ID 24642.

If your CAS-configured router gets stuck in the EM_PARK state, refer to *Troubleshooting EM_PARK Issues for E&M Digital CAS Signaling*, document ID 18959.

Troubleshooting Commands

Certain **show** commands are supported by the Output Interpreter tool, which allows you to view an analysis of **show** command output.

- **debug voip ccapi inout**—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.
- **debug vpm all**—Enables all of the **debug vpm** commands: **debug vpm spi**, **debug vpm signal**, and **debug vpm dsp**.

Note **Note:** This debug command generates lots of output.

- **show call active voice**—Displays the contents of the active call table, which shows all of the calls currently connected through the router.
- **show call history voice**—Displays the call history table. The call history table contains a listing of all calls connected through this router in descending time order since VoIP was enabled. You can display subsets of the call history table by using specific keywords.
- **show voice port**—Displays configuration information about a specific voice port.
- **debug vtsp all**—Enables the following **debug vtsp** commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**.

E1 R2 Interfaces

R2 signaling is a CAS system developed in the 1960s and still in use today in Europe, Latin America, Australia, and Asia. R2 signaling exists in several country versions or variants in an international version called Consultative Committee for International Telegraph and Telephone-R2 (CCITT-R2). The R2 signaling specifications are contained in International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Recommendations Q.400 through Q.490. E1 R2 signaling is an international signaling standard that is common to channelized E1 networks.

E1 R2 signaling support allows the Cisco gateways to communicate with a central office (CO) or PBX trunk and act as a tie-line replacement. Although R2 signaling has been defined in ITU-T Q.400-Q.490 recommendations, R2 is implemented in many different ways. (Various countries implement R2 differently.) Cisco's implementation of R2 signaling on routers can accommodate most of the variations.

Troubleshooting E1 R2 Failures

Follow the instructions below to troubleshoot your configuration.

SUMMARY STEPS

1. **show controller e1**
2. **show vfc** *slot number* **interface**
3. Configure DID on the POTS peer.
4. **cptone**
5. Match line and register signaling provisions to the switch configuration.

6. Turn on appropriate debugs.
7. Check for communication between the router and PBX or switch.

DETAILED STEPS

Step 1 Verify that controller E1 0 is up with the **show controller e1 0** command.

If it is down, check framing, line coding, clock source, and alarms. Replace the cable and reseat the card. To run these tests, see the following sections:

- Checking the Hardware, page 1
- Checking the Digital Signal Processors, page 3
- Verifying Codec Complexity, page 29
- Troubleshooting T1 and E1 Layer 1 Problems, page 31
- Checking T1/E1 Controller Configuration, page 34

Step 2 If you are using an AS5300, check that the DSPs are correctly installed with the **show vfc** *slot number* **interface** command.

Step 3 Configure direct inward dial (DID) on the plain old telephone service (POTS) peer, so that the received digits are used to choose an outgoing peer.

Step 4 Specify **cptone** (**cptone** is specific for your country) on the voice-ports.

A **cptone** *country* must be configured to match **cas-custom** *country*. The **cptone** parameter sets the call progress tones for a particular country, and more importantly sets the encoding to a-law or u-law, depending on the country. For u-law, use the **us** keyword. For a-law, use the **gb** keyword. To configure **cptone**, see the *Voice Port Configuration* document, Release 12.3.

Step 5 Match line and register signaling provisions to the switch configuration.

Step 6 Turn on some of the debugs shown in the following section and study the outputs.

Step 7 Check for communication between the router and PBX or switch:

- Is the line seized?
- Does the router receive/send digits?
- Find out which side is clearing the call.

If possible, use the latest Cisco IOS software release.

debug and show Commands

Certain **show** commands are supported by the Output Interpreter Tool, which allows you to view an analysis of **show** command output.

For Cisco IOS Software Release 12.0 and newer, use the following debugs:

- **debug cas**—For line signaling
- **debug csm voice**—For interregister signaling
- **debug vtsp all** —Exchanges output of all messages (digits) between the PBX and the router

For Cisco IOS Software Release IOS 11.3, use the following commands:

- **modem-mgmt csm debug-rbs** —For line signaling (specify **service internal** in global configuration mode.)
- **debug csm voice** — For interregister signaling
- **debug vtsp all** —Exchanges output of all messages (digits) between the PBX and the router

For the AS5400 and AS5350 platforms, use the following debugs:

- **debug sigsm r2**—For interregister signaling
- **debug vtsp all** —Exchanges output of all messages (digits) between the PBX and the router

ISDN Interfaces

An ISDN network can consist of T1, T3, E1, and E3 and has two types of subscriber access: Basic Rate Interface (BRI) and Primary Rate Interface (PRI). Each access comprises B and D channels.

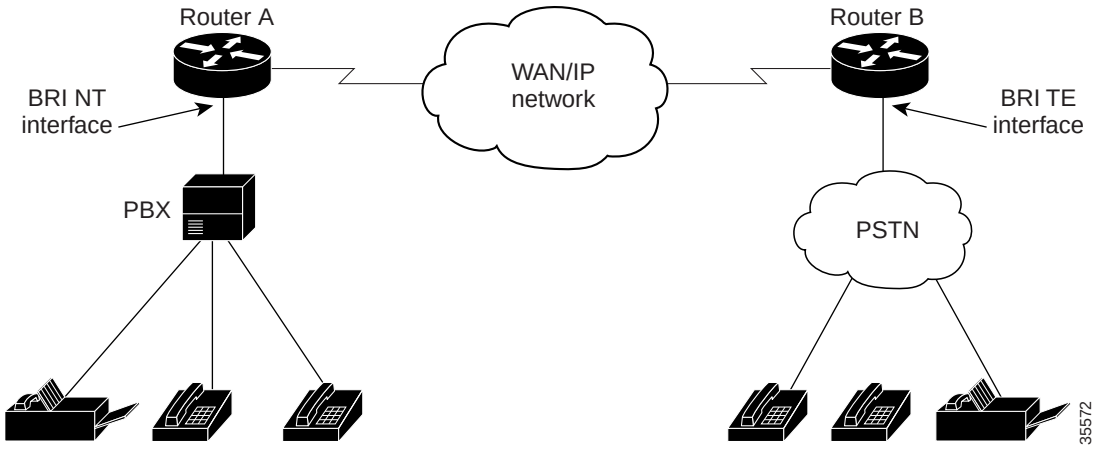
ISDN BRI provides two B channels, each capable of transferring voice or data at 64 kbps, and one 16-kbps D channel that carries signaling traffic. The D channel is used by the telephone network to carry instructions about how to handle each of the B channels. ISDN BRI (also referred to as “2 B + D”) provides a maximum transmission speed of 128 kbps.

ISDN PRI provides 23 B channels plus a D channel (in North America and Japan) or 30 B channels plus a D channel (in the rest of the world). Similar to the ISDN BRI D channel, the ISDN PRI D channel carries signaling traffic. ISDN PRI is often referred to as “23 B + D” (in North America and Japan) or “30 B + D” (in the rest of the world). The D channel notifies the central office switch to send the incoming call to particular time slots on the Cisco access server or router. Each one of the B channels carries data or voice. The D channel carries signaling for the B channels. The D channel indicates whether the call is a circuit-switched digital call or an analog modem call. Analog modem calls are decoded and then sent to the onboard modems. Circuit-switched digital calls are relayed directly to the ISDN processor in the router.

The ISDN interfaces for Cisco gateways enable Cisco IOS software to replicate the PSTN interface to a PBX that is compatible with European Telecommunications Standards Institute (ETSI) NET3 and QSIG switch types.

The application shown in Figure 36 allows enterprise customers with a large installed base of legacy telephony equipment to bypass the PSTN.

Figure 36 Typical Application Using ISDN BRI NT/TE VICs or ISDN BVMs



Topics for ISDN voice troubleshooting are as follows:

- ISDN PRI Troubleshooting Tips, page 42
- Verifying the ISDN Switch Type and PRI Group Timeslot Configuration, page 43
- QSIG Protocol Support, page 45

ISDN PRI Troubleshooting Tips

If you are having trouble connecting a call and you suspect that the problem is associated with voice port configuration, you can try to resolve the problem by performing the following tasks:

- Ping the associated IP address to confirm connectivity. If you cannot successfully ping your destination, refer to the chapter "Configuring IP" in the *Cisco IOS IP Configuration Guide.*
- Determine if the voice feature card (VFC) has been correctly installed. For more information, refer to *Installing Voice-over-IP Feature Cards in Cisco AS5300 Universal Access Servers*, which came with your voice network module (VNM).
- To learn if the VFC is operational, use the **show vfc** *slot_number* command.
- To view layer status information, use the **show isdn status** command. If you receive a status message stating that Layer 1 is deactivated, make sure the cable connection is not loose or disconnected.
- With T1 lines, determine if your a-law setting is correct. With E1 lines, determine if your u-law setting is correct. To configure both a-law and u-law values, use the **cptone** command. For more information about the **cptone** command, refer to the *Cisco IOS Voice, Video, and Fax Command Reference.*
- If dialing cannot occur, use the **debug isdn q931** command to check the ISDN configuration.

For more information about troubleshooting T1 PRI, refer to *T1 PRI Troubleshooting*, document ID 9344.

Verifying the ISDN Switch Type and PRI Group Timeslot Configuration

Use the **show running-config** command to ensure that **isdn switch-type** and **pri-group timeslots** are configured correctly. To specify the central office switch type on the ISDN interface, use the **isdn switch-type** global configuration command. Options for this command include **primary-net5**. Contact your service provider for the correct values to use.

Note If you have defined ISDN PRI groups and channel groups on the same controller, ensure that you do not overlap time slots or use the ISDN D-channel timeslot in a channel group. When configuring a Primary Rate Interface (PRI), use the **isdn switch-type** global configuration command to configure the switch type.

To configure the **isdn switch-type** and **pri-group**:

```
Router# configure terminal
Router(config)# isdn switch-type primary-net5
Router(config)# controller e1 0
Router(config-controlle)# pri-group timeslots 1-31
```

Note In some countries, service providers offer fractional PRI lines. This means that fewer than 30 B-channels may be used for ISDN connections. For fractional PRI lines, the time slot range must include the operational B channels, plus the D channel (this is fixed on time slot 16). For example:

- **pri-group timeslots 1–10, 16** for the first ten B-channels.
- **timeslots 1–21** for the first 20 B-channels.

Verifying the Signaling Channel

If the error counters do not increase, but the problem persists, complete the following steps to verify that the signaling channel is up and configured correctly

SUMMARY STEPS

1. **show interfaces serial** *number* **:15**
2. Ensure that the interface is up.
3. Ensure that encapsulation is PPP.
4. Ensure that the interface is not in loopback mode.
5. Power cycle the router.
6. Turn on appropriate debugs.
7. Contact TAC.

DETAILED STEPS

Step 1 Run the **show interfaces serial** *number***:15** command, where the number is the interface *number.*

Step 2 Ensure that the interface is up. If the interface is not up, use the **no shutdown** command to bring the interface up. For example:

```
Router# config terminal
```

```
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# interface serial 0:15
Router(config-if)# no shutdown
```

Step 3 Ensure that encapsulation is PPP. If not, use the **encapsulation ppp** command to set encapsulation. For example:

```
Router(config-if)# encapsulation ppp
```

Step 4 Ensure that the interface is not in loopback mode. Loopback should be set only for testing purposes. Use the **no loopback** command to remove loopbacks. For example:

```
Router(config-if)# no loopback
```

Step 5 Power cycle the router.

Step 6 If the problem persists, contact your service provider or the Cisco Technical Assistance Center (TAC). Refer to "Obtaining Technical Assistance" section on page xxvi for information about contacting TAC.

Ringback

In some situations, a gateway might intermittently fail to provide ringback tone (RBT) to an incoming ISDN caller. This problem has been seen on local, long-distance, and international calls.

The gateway generates ringback towards the network side (PSTN or PBX) if the setup contains Progress IE = 3, meaning the originating address (calling party) is NON-ISDN.

The gateway does *not* generate a ringback towards the network side (PSTN or PBX) if the setup contains no Progress IE (Progress IE =0), meaning the originating address (calling party) is ISDN.

The following figure is an example of when this might occur. International calls are arriving on ISDN.

PSTN (ISDN) --------- Cisco IOS gateway ------------ Cisco CallManager ------------ IP phone

Example of a Call That Gets Ringback Tone

You receive a call from a non-ISDN terminal. The setup contains a Progress IE = 3. The gateway generates ringback when it receives an alert from Cisco CallManager.

The following debugs were captured with the Cisco IOS command **debug isdn q931**:

```
01:34:48: ISDN Se0:15: RX <- SETUP pd = 8 callref = 0x002B
01:34:48: Sending Complete
01:34:48: Bearer Capability i = 0x9090A3
01:34:48: Channel ID i = 0xA9838D
01:34:48: Progress Ind i = 0x8583 - Origination address is non-ISDN
01:34:48: Calling Party Number i = 0x2183, '27045000', Plan:ISDN, Type:National
01:34:48: Called Party Number i = 0xA1, '27182145', Plan:ISDN, Type:National
01:34:48: ISDN Se0:15: TX -> CALL_PROC pd = 8 callref = 0x802B
01:34:48: Channel ID i = 0xA9838D
01:34:48: act_alert: Tone Ring Back generated in direction Network One
01:34:48: act_gen_tone: Tone Ring Back generated in direction Network
01:34:48: ISDN Se0:15: TX -> ALERTING pd = 8 callref = 0x802B
```

Example of a Call That Does Not Get Ringback Tone

You receive a call from an ISDN terminal. There is no Progress IE in the setup. (Progress IE = 0). The gateway is not generating ringback when it receives the alert from Cisco CallManager.

```
01:37:01: ISDN Se0:15: RX <- SETUP pd = 8 callref = 0x002E
01:37:01: Sending Complete
01:37:01: Bearer Capability i = 0x8090A3
01:37:01: Channel ID i = 0xA98391
01:37:01: Calling Party Number i = 0x2183, '478681058', Plan:ISDN, Type:International
01:37:01: Called Party Number i = 0xA1, '27182145', Plan:ISDN, Type:International
01:37:01: High Layer Compat i = 0x9181
01:37:01: ISDN Se0:15: TX -> CALL_PROC pd = 8 callref = 0x802E
01:37:01: Channel ID i = 0xA98391
01:37:01: ISDN Se0:15: TX -> ALERTING pd = 8 callref = 0x802E
```

In the case above, the gateway is expecting the ISDN to generate the ringback (due to no PI of 3). The ISDN, however, is not generating a ringback tone. This results in the caller hearing only silence until the called party answers the call. This might be due to an ISDN interworking issue caused because the call originated internationally (normally ring is generated at the terminating device for international calls).

Forcing Ringback

You can force the gateway to generate a ringback with the **progress_ind setup enable 3** command. Configure the forced ringback on the VoIP dial peer that points to Cisco CallManager.

```
!
dial-peer voice 500 voip
destination-pattern 5...
progress_ind setup enable 3
!--forces ring back tone for this peer
session target ipv4:10.200.73.15
codec g711ulaw
!
```

For more information, refer to *PSTN Callers not Hearing any Ring Back When they Call IP Phones*, document ID 8331.

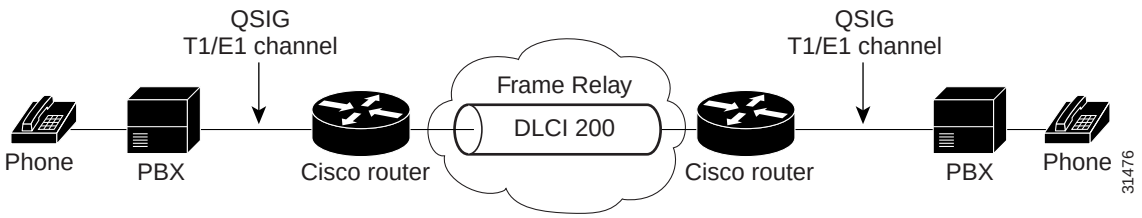
QSIG Protocol Support

QSIG is a peer-to-peer signaling system used in corporate voice networking. Internationally, QSIG is known as Private Signaling System No. 1 (PSS1). This open standard is a variant of ISDN D-channel voice signaling that is based on the ISDN Q.921 and Q.931 standards. Therefore, as well as providing inter-PBX communications, QSIG is compatible with public and private ISDN systems.

QSIG also has one important mechanism known as Generic Functional Procedures (QSIG GF). This mechanism provides a standard method for transporting features transparently across a network.

Integration of QSIG protocol support with Cisco voice switching services allows Cisco devices to connect PBXs, key systems, and central office switches (COs) that communicate by using the QSIG protocol. With QSIG, Cisco networks emulate the functionality of the PSTN, and QSIG signaling messages allow the dynamic establishment of voice connections across a Cisco WAN to a peer router, which can then transport the signaling and voice packets to a second PBX, as shown in Figure 37.

Figure 37 QSIG Signaling



The Cisco voice packet network appears to the traditional QSIG PBXs as a distributed transit PBX that can establish calls to any PBX, non-QSIG PBX, or other telephony endpoint served by a Cisco gateway, including non-QSIG endpoints.

QSIG Support Troubleshooting Tips

Table 43 lists **debug** and **show** commands that can help you analyze problems with your QSIG configuration.

Table 43 QSIG Troubleshooting Commands

Command	Purpose
`Router# show isdn status`	Displays the status of all ISDN interfaces, including active layers, timer information, and switch type settings.
`Router# show controller t1/e1`	Displays information about T1 and E1 controllers.
`Router# show voice port summary`	Displays summary information about voice port configuration.
`Router# show dial-peer voice`	Displays how voice dial peers are configured.
`Router# show cdapi`	Displays the Call Distributor Application Programming Interface (CDAPI) information.
`Router# show call history voice record`	Displays information about calls made to and from the router.
`Router# show rawmsg`	Displays information about any memory leaks.
`Router# debug isdn event`	Displays events occurring on the user side (on the router) of the ISDN interface. The ISDN events that can be displayed are Q.931 events (call setup and teardown of ISDN network connections).
`Router# debug tsp`	Displays information about the telephony service provider (TSP).
`Router# debug cdapi {events | detail}`	Displays information about CDAPI application events, registration, messages, and so on.

Troubleshooting Drop-and-Insert

When a router is configured for drop-and-insert, also called TDM cross connect, traffic is passed as a transparent bit stream between the configured ports. The router acts as a conduit between the ports, ensuring that the bit stream and clocking are preserved. Because of this, there are no commands to

monitor traffic or debug signaling bits. You can confirm the physical status of the T1 interfaces (carrier loss) and line quality (line errors, clock slips, framing errors) using the **show controller t1** *slot/port* command.

You can connect the PBX directly to the voice mail system to isolate signaling problems. If the system is still not working when the router has been bypassed, you might need to use T1 analyzers (for example, the Acterna Tberd T1 analyzer) to verify that the PBX or voice mail system is sending the correct information on the T1 trunk. You can also use the analyzer to verify that the drop-and-insert feature is working correctly from one port to the other.

For more information on troubleshooting drop-and-insert, refer to *Integrating PBXs into VoIP Networks Using the TDM Cross Connect Feature*, document ID 27789.

Troubleshooting Transparent Common Channel Signaling

Transparent Common Channel Signaling (T-CCS) allows the connection of two PBXs with digital interfaces that use a proprietary or unsupported CCS protocol without the need for interpretation of CCS signaling for call processing.

With T-CCS, the PBX voice channels can be “nailed-up” and compressed between sites, and the accompanying signaling channel(s) can be transparently tunneled across the IP/FR/ATM backbone between PBXs. Thus, calls from the PBXs are not routed by Cisco on a call-by-call basis but follow a preconfigured route to the destination.

For information about troubleshooting T-CCS, refer to *Configuring and Troubleshooting Transparent CCS*, document ID 19087.

Dial Tone Issues

This section discusses troubleshooting of the voice network when no dial tone is heard from a voice port that is in the off-hook condition. Some issues include:

A common problem in the voice network involves no dial tone being heard from a voice port in the off-hook condition. This might be related to configuration issues, a hardware problem, a DSP problem, or a bug in Cisco IOS software. A voice port configured with connection trunk does not provide dial tone. A faulty network module or Foreign Exchange Station (FXS) card might cause silence or no dial tone in a voice port.

For more information, refer to *Troubleshooting No Dial Tone Issues*, document ID 22372.

The following sections describe these various problems and related solutions.

Router Does Not Recognize Voice Port

When a router does not recognize a voice port, it might be because the router is not loaded with the Cisco IOS image required for voice support. In the case of a Cisco 1750 router, make sure that it does not have PVDM-256K-4 and PVDM-256K-8 DSPs. These are packet voice/data modules (PVDMs) for

Cisco routers 1751 and later. If the Cisco 1750 router does not have the correct PVDM, the voice ports might show up in **show version** and **show diag** command output, however, there is no dial tone. Also, no DSPs are seen in the **show voice dsp** command output. The Cisco 1750 router should carry the proper PVDM-4 and PVDM-8 DSP cards.

For Cisco modular access routers , another problem could be a bad network module. If there is an alarm light on the network module, remove the module, put it back in the slot and power cycle. If the alarm light is still lit, replace the network module. Also, try plugging an analog phone into the FXS port with a good cable. If there is no dial tone, replace the FXS card.

Note FXS-Direct Inward Dialing (DID) does not provide dial tone.

Voice Ports Are in the Shutdown State

When voice ports are in the shutdown state, they do not provide dial tone. To fix this, enable the voice port with the **no shut** command under the voice port.

Voice Ports Configured as Connection Trunk

If voice ports are configured as connection trunk, the ports are in the S_TRUNKED state. Voice ports do not provide dial tone if configured for connection private line auto ringdown (PLAR). Use the **show voice call summary** command to verify the VPM state. In the example below, the dial-tone is given by the remote PBX/PSTN.

```
Router# show voice call summary
PORT CODEC VAD VTSP STATE VPM STATE
==== ===== === ========== ==========
1/0/0 g729ar8 y S_CONNECT S_TRUNKED
1/0/1 g729ar8 y S_CONNECT S_TRUNKED
```

Remove the connection trunk/PLAR configuration to ensure that you are getting the dial-tone.

No Dial Tone on Digital Voice Port

Use the **show dial-peer** command to see if the dial-peer ports are configured with the **direct-inward-dial** command. This command disables dial tone from the voice port. For example:

```
dial-peer voice 1 pots
destination-pattern .T
direct-inward-dial
port 0:D
```

Removing the **direct-inward-dial** command from dial-peer pots causes the digital voice port to provide dial tone.

Debug Command Output Shows VTSP Timeout

The VTSP and DSP timeouts are known issues that appear in many forms. Use the **test dsps slot#** command to see if processors are alive. Cisco IOS Releases 12.2.6a [12.2.6cM1] and later include fixes for many of these issues, but possibly not all. The problem can be temporarily cleared by a power cycle. If you experience this problem, you should open a case with the Cisco Technical Assistance Center (TAC).

Verifying Digital Voice-Port Configurations

After configuring the voice ports on your router, follow these steps to verify proper operation:

SUMMARY STEPS

1. Check for dial tone.
2. Check for DTMF detection.
3. **show voice port summary**
4. **show voice port**
5. **show running-config**
6. show controller {t1 | e1} *controller-number*
7. **show voice dsp**
8. **show voice call summary**
9. **show call active voice**
10. **show call history voice** {**last** | *number* | **brief**}

DETAILED STEPS

Step 1 Pick up the handset of an attached telephony device and check for a dial tone.

Step 2 If you have dial tone, check for DTMF detection. If the dial tone stops when you dial a digit, then the voice port is most likely configured properly for DTMF detection.

Step 3 To identify port numbers of voice interfaces installed in your router, use the **show voice port summary** command.

Step 4 To verify voice-port parameter settings, enter the **show voice port** command with the appropriate syntax for your voice interfaces.

Step 5 For digital T1/E1 connections, to verify the codec complexity configuration, enter the **show running-config** command to display the current voice-card setting. If medium complexity is specified, the codec complexity setting is not displayed. If high complexity is specified, the setting codec complexity **high** is displayed. The following example shows an excerpt from the command output when high complexity has been specified:

```
Router# show running-config
.
.
.
hostname router-alpha

voice-card 0
 codec complexity high
.
.
.
```

Step 6 For digital T1/E1 connections, to verify that the controller is up and that no alarms have been reported, and to display information about clock sources and other controller settings, use the **show controller** command. For output examples, see the "show controller Samples" section on page 52.

```
Router# show controller {t1 | e1} controller-number
```

Step 7 To display voice-channel configuration information for all DSP channels, enter the **show voice dsp command**. For output examples, see the "show voice dsp Samples" section on page 52.

```
Router# show voice dsp
```

Step 8 To verify the call status for all voice ports, enter the **show voice call summary command**. For output examples, see the "show voice call summary Samples" section on page 53.

```
Router# show voice call summary
```

Step 9 To display the contents of the active call table, which shows all of the calls currently connected through the router or concentrator, enter **the show call active voice command**. For output examples, see the "show call active voice Samples" section on page 54.

```
Router# show call active voice
```

Step 10 To display the contents of the call history table, enter the **show call history voice command**. To limit the display to the last calls connected through this router, enter the keyword **last** and define the number of calls to be displayed with the argument *number*. To limit the display to a shortened version of the call history table, use the keyword **brief**. For output examples, see the "show call history voice Sample" section on page 55.

```
Router# show call history voice {last | number | brief}
```

show voice port Samples

In the following sections, output examples of the following types are shown:

- Cisco 3600 series digital E&M voice port
- Cisco AS5300 universal access server T1 CAS voice port
- Cisco 7200 series router digital E&M voice port

Cisco 3600 Digital E&M Voice Port

```
Router# show voice port 1/0:1

receEive and transMit Slot is 1, Sub-unit is 0, Port is 1
 Type of VoicePort is E&M
 Operation State is DORMANT
 Administrative State is UP
 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
 Music On Hold Threshold is Set to -38 dBm
 In Gain is Set to 0 dB
 Out Attenuation is Set to 0 dB
 Echo Cancellation is enabled
 Echo Cancel Coverage is set to 8 ms
 Connection Mode is normal
 Connection Number is not set
 Initial Time Out is set to 10 s
 Interdigit Time Out is set to 10 s
 Region Tone is set for US
```

Cisco AS5300 Universal Access Server T1 CAS Voice Port

```
Router# show voice port

DS0 Group 1:0 - 1:0
 Type of VoicePort is CAS
 Operation State is DORMANT
 Administrative State is UP
 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
 Music On Hold Threshold is Set to -38 dBm
 In Gain is Set to 0 dB
 Out Attenuation is Set to 0 dB
 Echo Cancellation is enabled
 Echo Cancel Coverage is set to 8 ms
 Playout-delay Mode is set to default
 Playout-delay Nominal is set to 60 ms
 Playout-delay Maximum is set to 200 ms
 Connection Mode is normal
 Connection Number is not set
 Initial Time Out is set to 10 s
 Interdigit Time Out is set to 10 s
 Call-Disconnect Time Out is set to 60 s
 Ringing Time Out is set to 180 s
 Companding Type is u-law
 Region Tone is set for US
 Wait Release Time Out is 30 s
 Station name None, Station number None

 Voice card specific Info Follows:

 DS0 channel specific status info:
                                IN      OUT
    PORT   CH SIG-TYPE   OPER STATUS   STATUS    TIP     RING
```

Cisco 7200 Series Router Digital E&M Voice Port

```
Router# show voice port 1/0:1

receEive and transMit Slot is 1, Sub-unit is 0, Port is 1  << voice-port 1/0:1

 Type of VoicePort is E&M

 Operation State is DORMANT

 Administrative State is UP

 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
 Music On Hold Threshold is Set to -38 dBm
 In Gain is Set to 0 dB

 Out Attenuation is Set to 0 dB

 Echo Cancellation is enabled

 Echo Cancel Coverage is set to 8 ms
 Connection Mode is normal
 Connection Number is not set
 Initial Time Out is set to 10 s
```

```
 Interdigit Time Out is set to 10 s

 Region Tone is set for US
```

show controller Samples

In the following sections, output examples of the following types are shown:

- Cisco 3600 series router T1 controller
- Cisco AS5800 universal access server T1 controller

Cisco 3600 Series Router T1 Controller

```
Router# show controller T1 1/1/0

T1 1/0/0 is up.
  Applique type is Channelized T1
  Cablelength is long gain36 0db
  No alarms detected.
  alarm-trigger is not set
  Framing is ESF, Line Code is B8ZS, Clock Source is Line.
  Data in current interval (180 seconds elapsed):
     0 Line Code Violations, 0 Path Code Violations
     0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
     0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
```

Cisco AS5800 Universal Access Server T1 Controller

```
Router# show controller t1 2
 T1 2 is up.
   No alarms detected.
   Version info of slot 0:  HW: 2, Firmware: 16, PLD Rev: 0

 Manufacture Cookie Info:
  EEPROM Type 0x0001, EEPROM Version 0x01, Board ID 0x42,
  Board Hardware Version 1.0, Item Number 73-2217-4,
  Board Revision A0, Serial Number 06467665,
  PLD/ISP Version 0.0, Manufacture Date 14-Nov-1997.

   Framing is ESF, Line Code is B8ZS, Clock Source is Internal.
   Data in current interval (269 seconds elapsed):
    0 Line Code Violations, 0 Path Code Violations
      0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs, 0 Degraded Mins
      0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs, 0 Unavail Secs
```

show voice dsp Samples

The following output is from a Cisco 3640 router when a digital voice port is configured.

```
Router# show voice dsp

TYPE DSP CH CODEC    VERS STATE STATE   RST AI PORT    TS ABORT   TX/RX-PAK-CNT
==== === == ======== ==== ===== ======= === == ======= == ===== ===============
C549 010 00 g729r8    3.3 busy  idle      0  0 1/015   1     0     67400/85384
         01 g729r8     .8 busy  idle      0  0 1/015   7     0     67566/83623
         02 g729r8        busy  idle      0  0 1/015  13     0     65675/81851
```

```
         03 g729r8        busy  idle      0  0 1/015  20     0     65530/83610
C549 011 00 g729r8    3.3 busy  idle      0  0 1/015   2     0     66820/84799
         01 g729r8     .8 busy  idle      0  0 1/015   8     0     59028/66946
         02 g729r8        busy  idle      0  0 1/015  14     0     65591/81084
         03 g729r8        busy  idle      0  0 1/015  21     0     66336/82739
C549 012 00 g729r8    3.3 busy  idle      0  0 1/015   3     0     59036/65245
         01 g729r8     .8 busy  idle      0  0 1/015   9     0     65826/81950
         02 g729r8        busy  idle      0  0 1/015  15     0     65606/80733
         03 g729r8        busy  idle      0  0 1/015  22     0     65577/83532
C549 013 00 g729r8    3.3 busy  idle      0  0 1/015   4     0     67655/82974
         01 g729r8     .8 busy  idle      0  0 1/015  10     0     65647/82088
         02 g729r8        busy  idle      0  0 1/015  17     0     66366/80894
         03 g729r8        busy  idle      0  0 1/015  23     0     66339/82628
C549 014 00 g729r8    3.3 busy  idle      0  0 1/015   5     0     68439/84677
         01 g729r8     .8 busy  idle      0  0 1/015  11     0     65664/81737
         02 g729r8        busy  idle      0  0 1/015  18     0     65607/81820
         03 g729r8        busy  idle      0  0 1/015  24     0     65589/83889
C549 015 00 g729r8    3.3 busy  idle      0  0 1/015   6     0     66889/83331
         01 g729r8     .8 busy  idle      0  0 1/015  12     0     65690/81700
         02 g729r8        busy  idle      0  0 1/015  19     0     66422/82099
         03 g729r8        busy  idle      0  0 1/015  25     0     65566/83852

Router# show voice dsp

TYPE DSP CH CODEC    VERS STATE STATE   RST AI PORT    TS ABORT   TX/RX-PAK-CNT
==== === == ======== ==== ===== ======= === == ======= == ===== ===============
C549 007 00 {medium}  3.3 IDLE  idle      0  0 1/0:1    4     0              0/0
                      .13
C549 008 00 {medium}  3.3 IDLE  idle      0  0 1/0:1    5     0              0/0
                      .13
C549 009 00 {medium}  3.3 IDLE  idle      0  0 1/0:1    6     0              0/0
                      .13
C549 010 00 {medium}  3.3 IDLE  idle      0  0 1/0:1    7     0              0/0
                      .13
C549 011 00 {medium}  3.3 IDLE  idle      0  0 1/0:1    8     0              0/0
                      .13
C549 012 00 {medium}  3.3 IDLE  idle      0  0 1/0:1    9     0              0/0
                      .13
C542 001 01 g711ulaw  3.3 IDLE  idle      0  0 2/0/0          0         512/519
                      .13
C542 002 01 g711ulaw  3.3 IDLE  idle      0  0 2/0/1          0         505/502
                      .13
C542 003 01 g711alaw  3.3 IDLE  idle      0  0 2/1/0          0     28756/28966
                      .13
C542 004 01 g711ulaw  3.3 IDLE  idle      0  0 2/1/1          0         834/838
                      .13
```

show voice call summary Samples

The following is an output example of show voice call summary on a Cisco 3600 series router digital voice port:

Cisco 3600 Series Router Digital Voice Port

```
Router# show voice call summary
PORT      CODEC    VAD VTSP STATE            VPM STATE
========= ======== === ==================== ========================
1/015.1   g729r8    y  S_CONNECT            S_TSP_CONNECT
1/015.2   g729r8    y  S_CONNECT            S_TSP_CONNECT
1/015.3   g729r8    y  S_CONNECT            S_TSP_CONNECT
```

```
1/015.4  g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.5  g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.6  g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.7  g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.8  g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.9  g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.10 g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.11 g729r8    y  S_CONNECT             S_TSP_CONNECT
1/015.12 g729r8    y  S_CONNECT             S_TSP_CONNECT
```

show call active voice Samples

The output from the Cisco 7200 series router is shown below.

```
Router# show call active voice
GENERIC:
SetupTime=94523746 ms
Index=448
PeerAddress=##73072

PeerSubAddress=
PeerId=70000

PeerIfIndex=37
LogicalIfIndex=0
ConnectTime=94524043
DisconectTime=94546241
CallOrigin=1

ChargedUnits=0
InfoType=2
TransmitPackets=6251
TransmitBytes=125020
ReceivePackets=3300
ReceiveBytes=66000
VOIP:
ConnectionId[0x142E62FB 0x5C6705AF 0x0 0x385722B0]
RemoteIPAddress=171.68.235.18

RemoteUDPPort=16580

RoundTripDelay=29 ms

SelectedQoS=best-effort
tx_DtmfRelay=inband-voice
SessionProtocol=cisco
SessionTarget=ipv4:171.68.235.18
OnTimeRvPlayout=63690
GapFillWithSilence=0 ms

GapFillWithPrediction=180 ms

GapFillWithInterpolation=0 ms
GapFillWithRedundancy=0 ms
HiWaterPlayoutDelay=70 ms
LoWaterPlayoutDelay=30 ms
ReceiveDelay=40 ms
LostPackets=0 ms

EarlyPackets=1 ms
```

```
LatePackets=18 ms

VAD = disabled

CoderTypeRate=g729r8

CodecBytes=20

cvVoIPCallHistoryIcpif=0
SignalingType=cas
```

show call history voice Sample

The following output is for the Cisco 7200 series router.

```
Router# show call history voice
GENERIC:
SetupTime=94893250 ms
Index=450
PeerAddress=##52258

PeerSubAddress=
PeerId=50000

PeerIfIndex=35
LogicalIfIndex=0
DisconnectCause=10

DisconnectText=normal call clearing.

ConnectTime=94893780
DisconectTime=95015500
CallOrigin=1

ChargedUnits=0
InfoType=2
TransmitPackets=32258
TransmitBytes=645160
ReceivePackets=20061
ReceiveBytes=401220
VOIP:
ConnectionId[0x142E62FB 0x5C6705B3 0x0 0x388F851C]
RemoteIPAddress=171.68.235.18

RemoteUDPPort=16552

RoundTripDelay=23 ms

SelectedQoS=best-effort
tx_DtmfRelay=inband-voice
SessionProtocol=cisco
SessionTarget=ipv4:171.68.235.18
OnTimeRvPlayout=398000
GapFillWithSilence=0 ms

GapFillWithPrediction=1440 ms

GapFillWithInterpolation=0 ms
GapFillWithRedundancy=0 ms
HiWaterPlayoutDelay=97 ms
```

```
LoWaterPlayoutDelay=30 ms
ReceiveDelay=49 ms
LostPackets=1 ms
EarlyPackets=1 ms

LatePackets=132 ms

VAD = disabled

CoderTypeRate=g729r8

CodecBytes=20
cvVoIPCallHistoryIcpif=0
```

Verifying Digits Received and Sent on the POTS Call Leg

Once the on-hook and off-hook signaling are verified to be working correctly, verify the correct digits are being received or sent on the digital voice port. A dial peer that does not matched or the switch (CO or PBX) cannot ring the correct station if incomplete or incorrect digits are being sent or received. Some commands that can be used to verify the digits received or sent are:

- **show dialplan number**—This command is used to show which dial peer is reached when a particular telephone number is dialed.
- **debug vtsp session**—This command displays information on how each network indication and application request is processed, signaling indications, and DSP control messages.
- **debug vtsp dsp**—This command displays the digits as they are received by the voice port.
- **debug vtsp all**—This command enables the following debug voice telephony service provider (VTSP) commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**.

show dialplan number

The **show dialplan number** command displays the dial peer that is matched by a string of digits. If multiple dial-peers can be matched, they are all shown in the order in which they are matched. The output of this command looks like this:

```
Router# show dialplan number 5000
Macro Exp.: 5000

VoiceOverIpPeer2
        information type = voice,
        tag = 2, destination-pattern = `5000',
        answer-address = `', preference=0,
        group = 2, Admin state is up, Operation
        state is up,
        incoming called-number = `',
        connections/maximum = 0/unlimited,
        application associated:
        type = voip, session-target =
        `ipv4:192.168.10.2',
        technology prefix:
        ip precedence = 5, UDP checksum =
        disabled, session-protocol = cisco,
        req-qos = best-effort,
        acc-qos = best-effort,
dtmf-relay = cisco-rtp,
        fax-rate = voice,
        payload size =  20 bytes
```

```
codec = g729r8,
payload size =  20 bytes,
Expect factor = 10, Icpif = 30,
signaling-type = cas,
VAD = enabled, Poor QOV Trap = disabled,
Connect Time = 25630, Charged Units = 0,
Successful Calls = 25, Failed Calls = 0,
Accepted Calls = 25, Refused Calls = 0,
Last Disconnect Cause is "10  ",
Last Disconnect Text is "normal call
clearing.",
Last Setup Time = 84427934.
Matched: 5000   Digits: 4
Target: ipv4:192.168.10.2
```

debug vtsp dsp

debug vtsp dsp shows the digits as they are received by the voice-port. The following output shows the collection of DTMF digits from the DSP:

```
Router# debug vtsp dsp
Voice telephony call control dsp debugging is on


!-- ACTION: Caller picked up handset and dialed
!-- digits 5000.
!-- The DSP detects DTMF digits. Digit 5 was
!-- detected with ON time of 130msec.


*Mar 10 17:57:08.505: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=5,
*Mar 10 17:57:08.585: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=5,
duration=130
*Mar 10 17:57:09.385: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:09.485: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=150
*Mar 10 17:57:10.697: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:10.825: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=180
*Mar 10 17:57:12.865: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:12.917: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=100

Router# debug vtsp session
Voice telephony call control session debugging is on


!--- <some output have been omitted>
!-- ACTION: Caller picked up handset.
!-- The DSP is allocated, jitter buffers, VAD
!-- thresholds, and signal levels are set.


*Mar 10 18:14:22.865: dsp_set_playout: [1/0/0 (69)]
```

```
packet_len=18 channel_id=1 packet_id=76 mode=1
initial=60 min=4 max=200 fax_nom=300
*Mar 10 18:14:22.865: dsp_echo_canceller_control:
[1/0/0 (69)] packet_len=10 channel_id=1 packet_id=66
flags=0x0
*Mar 10 18:14:22.865: dsp_set_gains: [1/0/0 (69)]
packet_len=12 channel_id=1 packet_id=91
in_gain=0 out_gain=65506
*Mar 10 18:14:22.865: dsp_vad_enable: [1/0/0 (69)]
packet_len=10 channel_id=1 packet_id=78
thresh=-38act_setup_ind_ack
*Mar 10 18:14:22.869: dsp_voice_mode: [1/0/0 (69)]
packet_len=24 channel_id=1 packet_id=73 coding_type=1
voice_field_size=80
VAD_flag=0 echo_length=64 comfort_noise=1
inband_detect=1 digit_relay=2
AGC_flag=0act_setup_ind_ack(): dsp_dtmf_mod
e()act_setup_ind_ack: passthru_mode = 0,
no_auto_switchover = 0dsp_dtmf_mode
(VTSP_TONE_DTMF_MODE)


!-- The DSP is put into "voice mode" and dial-tone is
!-- generated.


*Mar 10 18:14:22.873: dsp_cp_tone_on: [1/0/0 (69)]
packet_len=30 channel_id=1 packet_id=72 tone_id=4
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
```

If the digits are not being sent or received properly, then it might be necessary to use either a digit-grabber (test tool) or T1 tester to verify that the digits are being sent at the correct frequency and timing interval. If they are being sent “incorrectly” for the switch (CO or PBX), some values on the router or switch (CO or PBX) might need to be adjusted so that they match and can interoperate. These are usually digit duration and inter-digit duration values.

Another item to examine if the digits appear to be sent correctly are any number translation tables in the switch (CO or PBX) that may add or remove digits. Refer to your switch documentation to check the translation tables on your switch.

Voice Port Testing Commands

These commands allow you to force voice ports into specific states for testing. The following types of voice-port tests are covered:

Detector-Related Function Tests

Using the **test voice port detector** command, you are able to force a particular detector into an on or off state, perform tests on the detector, and then return the detector to its original state.

To configure this feature, enter these commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/port:ds0-group detector {m-lead | battery-reversal | loop-current | ring | tip-ground | ring-ground | ring-trip} {on | off}`	Identifies the voice port you want to test. Enter a keyword for the detector under test and specify whether to force it to the on or off state.
Step 2	`Router# test voice port slot/port:ds0-group detector {m-lead | battery-reversal | loop-current | ring | tip-ground | ring-ground | ring-trip} disable`	Identifies the voice port on which you want to end the test. Enter a keyword for the detector under test and the keyword **disable** to end the forced state.

Loopback Function Tests

To establish loopbacks on a voice port, enter the following commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/port:ds0-group loopback {local | network}`	Identifies the voice port you want to test and enters a keyword for the loopback direction. **Note** A call must be established on the voice port under test.
Step 2	`Router# test voice port slot/port:ds0-group loopback disable`	Identifies the voice port on which you want to end the test and enters the keyword disable to end the loopback.

Tone Injection Tests

To inject a test tone into a voice port, enter the following commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/port:ds0-group inject-tone {local | network} {1000hz | 2000hz | 200hz | 3000hz | 300hz | 3200hz | 3400hz | 500hz | quiet}`	Identifies the voice port you want to test and enter keywords for the direction to send the test tone and for the frequency of the test tone. **Note** A call must be established on the voice port under test.
Step 2	`Router# test voice port slot/port:ds0-group inject-tone disable`	Identifies the voice port on which you want to end the test and enter the keyword **disable** to end the test tone. **Note** The **disable** keyword is available only if a test condition is already activated.

Relay-Related Function Tests

To test relay-related functions on a voice port, enter the following commands beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/port:ds0-group relay {e-lead | loop | ring-ground | battery-reversal | power-denial | ring | tip-ground} {on | off}`	Identifies the voice port you want to test. • Enter a keyword for the relay under test and specify whether to force it to the on or off state.
Step 2	`Router# test voice port slot/port:ds0-group relay {e-lead | loop | ring-ground | battery-reversal | power-denial | ring | tip-ground} disable`	Identifies the voice port on which you want to end the test. • Enter a keyword for the relay under test, and the keyword **disable** to end the forced state.

Fax/Voice Mode Tests

The **test voice port switch fax** command forces a voice port into fax mode for testing. After you enter this command, you can use the **show voice call** or **show voice call summary** command to check whether the voice port is able to operate in fax mode. If no fax data is detected by the voice port, the voice port remains in fax mode for 30 seconds and then reverts automatically to voice mode.

The **disable** keyword ends the forced mode switch; however, the fax mode ends automatically after 30 seconds. The disable keyword is available only while the voice port is in fax mode.

To force a voice port into fax mode and return it to voice mode, enter the following commands, beginning in privileged EXEC mode:

	Command	Purpose
Step 1	`Router# test voice port slot/port:ds0-group switch fax`	Identifies the voice port you want to test. • Enter the keyword **fax** to force the voice port into fax mode.
Step 2	`Router# test voice port slot/port:ds0-group switch disable`	Identifies the voice port on which you want to end the test. • Enter the keyword **disable** to return the voice port to voice mode.

Troubleshooting Quality of Service for VoIP

The primary goal of Quality of Service (QoS) is to make network service better and more predictable by providing dedicated bandwidth, controlled jitter and latency, and improved loss characteristics. QoS achieves these goals by providing tools for managing network congestion, shaping network traffic, using expensive wide-area links more efficiently, and setting traffic policies across the network.

For information about configuring QoS for voice, refer to the *Quality of Service for Voice* document. For more information about QoS, refer to the *Cisco IOS Quality of Service Solutions Configuration Guide*.

This chapter contains the following topics:

Understanding VoIP QoS Issues

When VoIP calls are properly established, the next step is to verify that the voice quality is good. You should consider the following guidelines as you attempt to achieve good voice quality:

- Understand how much bandwidth a VoIP call consumes with each codec, including Layer 2 and IP/UDP/RTP headers. For more information about VoIP bandwidth consumption, refer to *Voice Over IP - Per Call Bandwidth Consumption*, document ID 7934.
- Understand the characteristics of the IP network over which the calls travel. For example, the bandwidth of a Frame Relay network at CIR is much different than that above-CIR (or burst), where packets could be dropped or queued in the Frame-Relay cloud. Ensure that delay and jitter are controlled and eliminated as much as possible. One-way transmit delay should not exceed 150 ms (per G.114 recommendation).
- Use a queuing technique that allows VoIP traffic to be identified and prioritized.
- When transmitting VoIP over low-speed links, consider using Layer 2 packet fragmentation techniques, such as Multilink Point-to-Point Protocol (MLPPP) with Link Fragmentation and Interleaving (LFI) on point-to-point links, or FRF.12 on Frame Relay links. Fragmentation of larger data packets allows less jitter and delay in transmitting VoIP traffic because the VoIP packets can be interleaved onto the link.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Try the call with a different codec and with the voice activity detector (VAD) enabled and disabled to possibly narrow down the issue to the digital signal processor (DSP), as opposed to the IP network.

With VoIP, when you are troubleshooting QoS issues, look especially for dropped packets and network bottlenecks that can cause delay and jitter.

Specifically, look for the following:

- Interface drops
- Buffer drops
- Policy-map drops
- Interface congestion
- Link congestion

Each interface in the path of the VoIP call should be examined and drops and congestion should be eliminated. Also, round-trip delay should be reduced as much as possible. Pings between the VoIP end points give an indication of the round trip delay of a link. The round trip delay should not exceed 300 ms whenever possible. If the delay does have to exceed this value, efforts should also be taken to ensure this delay is constant, so that you do not introduce jitter or variable delay.

When low latency queueing (LLQ) is set up, you can check policy-map drops by looking for packets that match the priority class by entering the **show policy interface serial** command. This shows how much traffic is matching the priority class and how much bandwidth is used.

Ensure that the Cisco IOS queuing mechanism is placing the VoIP packets within the proper queues. Cisco IOS commands, such as **show queue** and **show priority** can help you to verify queueing.

Measuring QoS

Cisco offers several options for monitoring QoS in networks using VoIP solutions. Cisco offers tools that provide information about the voice quality you are experiencing by measuring delay, jitter, and packet loss. Cisco solutions do not measure voice quality using Perceptual Speech Quality Measurement (PSQM) or some of the new proposed algorithms for voice quality measurement. Tools from outside vendors are available for this purpose.

When implementing service policies using the QoS command line interface (CLI), start with the Cisco Class-Based QoS Configuration and Statistics Management Information Base (MIB). This MIB provides read access to QoS configuration and statistics information for Cisco platforms that support the Modular QoS CLI. Statistics available through this MIB include summary counts and rates by traffic class before and after any configured QoS policies are enforced. In addition, detailed feature-specific statistics are available for select PolicyMap features. See Cisco MIBs for the object IDs.

In addition, Cisco offers the following software tools for monitoring QoS:

- Network monitoring using Cisco Service Assurance Agent (Cisco SSA)—The response time and availability monitoring capabilities of this tool include support for VoIP, QoS, and the World Wide Web. The Cisco SSA is an application-aware synthetic operation agent that monitors network performance by measuring key metrics such as response time, availability, jitter (interpacket delay variance), connect time, throughput, and packet loss. These metrics can be used for troubleshooting, for analysis before problems occur, and for designing future network topologies. This tool is designed more for trending, rather than real-time monitoring. Refer to *Using Cisco Service Assurance Agent and Internetwork Performance Monitor to Manage Quality of Service in Voice over IP Networks* for more information.

- Cisco Gateway Management Agent (CGMA)—The only real-time management Cisco IOS software agent and protocol for VoIP. The CGMA is a new gateway Cisco IOS agent that provides real-time call-state information for all VoIP calls. CGMA supports a push protocol, in which certain call-state changes result in a message being sent out of CGMA by the gateways. The interface from the CGMA is the Real Time Management Protocol (RTMP). RTMP is a lightweight XML-based protocol that uses TCP as the transport protocol. This solution allows Service Providers to monitor their calls (session initiation protocol (SIP) and H.323 networks), viewing call detail records (CDRs) and trunk utilization in real time. The validated gateways for the CGMA include the Cisco 2600 series, the Cisco 3600 series, and the Cisco 5000 series. The Cisco IOS that has been validated on all gateways is the 12.2(2)XB mainline release.
- CiscoWorks QoS Policy Manager (QPM)—QPM provides a scalable platform for defining, applying, and monitoring QoS policy on a system-wide basis for Cisco devices, including routers and switches.

  QPM enables you to baseline profile network traffic, create QoS policies at an abstract level, control the deployment of policies, and then monitor QoS to verify intended results. As a centralized tool QPM is used to monitor and provision QoS for groups of interfaces and devices.

  QPM provides a web-based intuitive user interface to define QoS policies, and translates those policies into the device's command line interface (CLI) commands.

  QPM runs on the CiscoWorks Common Services server, which can be installed as a standalone server, or as an add-on to CD One 5th Edition. CiscoWorks Common Services provides the infrastructure required by QPM to run from the CiscoWorks desktop environment, and also provides management of user roles and privileges, allowing you to control who gets access to specific tasks in QPM.

Common QoS Problems

Common voice quality issues include the following:

- Delay in Voice Networks
- Packet Loss
- Jitter Adjustment
- Echo Adjustment
- Voice Level Adjustment

Table 44 lists some QoS problems you might encounter after configuring voice ports and suggests some remedies. To configure QoS features on voice ports, see the "Fine-Tuning Analog and Digital Voice Ports" section in the *Voice Port Configuration* document.

Table 44 Troubleshooting Voice Quality on Voice Ports

Symptom	Suggested Action
Telephony device buzzes or does not ring.	Use the **show voice port** command to confirm that ring frequency is configured correctly. It must match the connected telephony equipment and might be country-dependent.
Distorted speech.	Use the **show voice port** command to confirm that the **cptone** keyword setting (also called *region tone*) is set to **us** for the United States. **Note** Having a wrong **cptone** setting can cause faulty voice reproduction during analog-to-digital or digital-to-analog conversions.
Music on Hold (MOH) is not heard.	Reduce the music-threshold level to make VAD less sensitive with the **music-threshold** voice-port configuration command. Valid entries are from -70 to -30.
Background noise is not heard.	Enable the **comfort-noise** command.
Long pauses occur in conversation, as if you were speaking on a walkie-talkie.	Overall delay is probably excessive; the standard for adequate voice quality is 150 ms one-way transit delay. Measure delay by using ping tests at various times of the day with different network traffic loads. If delay must be reduced, examine propagation delay of signals between the sending and receiving endpoints, voice encoding delay, and the voice packetization time for various VoIP codecs.
Jerky or choppy speech.	Variable delay, or jitter, is being introduced by congestion in the packet network. There are two possible remedies: • Reduce the amount of congestion in your packet network. Pings between VoIP endpoints provide information about the round-trip delay of a link, which should never exceed 300 ms. Network queuing and dropped packets should also be examined. • Increase the size of the jitter buffer with the **playout-delay** command. (See the "Jitter Adjustment" section on page 7 for more information.)
Clipped or fuzzy speech.	Reduce input gain. (See the "Voice Level Adjustment" section on page 14 for more information.) Change the VAD level. Sometimes VAD cuts the sound too early and the speaker's voice is clipped. You can also change the time period that VAD waits for silence.
Clipped speech.	Reduce the input level at the listener's router. (See the "Voice Level Adjustment" section on page 14 for more information)
Volume too low or missed DTMF.	Increase speaker's output level or listener's input level. (See the "Voice Level Adjustment" section on page 14 for more information.)

Table 44 Troubleshooting Voice Quality on Voice Ports (continued)

Symptom	Suggested Action
Echo interval is greater than 25 ms (sounds like a separate voice).	Configure the **echo-cancel enable** command and increase the value for the **echo-cancel coverage** keyword. (See the "Echo Adjustment" section on page 9 for more information.)
Too much echo.	Reduce the output level at the speaker's voice port. (See the "Voice Level Adjustment" section on page 14 for more information.)

Delay in Voice Networks

Delay is the time it takes for voice packets to travel between two endpoints. Excessive delay can cause quality problems with real-time traffic such as voice. However, because of the speed of network links and the processing power of intermediate devices, some delay is expected.

When listening to speech, the human ear normally accepts up to about 150 ms of delay without noticing it. The ITU G.114 standard recommends no more than 150 ms of one-way delay for a normal voice conversation. Once the delay exceeds 150 ms, a conversation is more like a "walkie-talkie" conversation in which one person must wait for the other to stop speaking before beginning to talk.

Several different types of delay combine to make up the total end-to-end delay associated with voice calls:

- Coder delay—Amount of time used by the digital signal processor (DSP) to compress samples.
- Handling delay—Amount of time it takes to process data (adding headers, taking samples, forming packets, and so forth).
- Serialization delay—Fixed delay related to the clock rate on the interface. The amount of delay needed for different frame sizes is shown in Table 45.

Table 45 Serialization Delay

	Frame Size						
Line Speed	**1 Byte**	**64 Bytes**	**128 Bytes**	**256 Bytes**	**512 Bytes**	**1024 Bytes**	**1500 Bytes**
56 kpbs	143 µs	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms
64 kbps	125 µs	8 ms	16 ms	32 ms	64 ms	128 ms	187 ms
128 kbps	62.5 µs	4 ms	8 ms	16 ms	32 ms	64 ms	93 ms
256 kbps	31 µs	2 ms	4 ms	8 ms	16 ms	32 ms	46 ms
512 kbps	15.5 µs	1.28 ms	2 ms	4 ms	8 ms	16 ms	23 ms
768 kbps	10 µs	640 µs	1.28 ms	2.56 ms	5.12 ms	10.24 ms	15 ms
1536 kbps	5 µs	320 µs	640 µs	1.28 ms	2.56 ms	5.12 ms	7.5 ms

- Propagation delay—Amount of time it takes the data to physically travel over the media.
- Queuing delay—Amount of time lost due to congestion. Every network device between two endpoints involved in a call is a potential source of queuing or buffering delays. Use a sniffer trace at each hop to see where the delay or jitter is being introduced.

Output drops can occur because of incorrectly configured priority queuing. For information about troubleshooting output drops with priority queueing, refer to *Troubleshooting Output Drops with Priority Queueing* , document ID 10105.

- Variable delay or jitter—Amount of time that causes the conversation to break and become unintelligible. Jitter is described in the "Jitter Adjustment" section on page 7.

You can measure delay by using ping tests at various times of the day with different network traffic loads. Sniffer traces can also be used to find where delay is being introduced in the network. For more information about measuring QoS, see the "Measuring QoS" section on page 2.

The recommended ranges for delay are shown in Table 46. These ranges are for connections in which echo is controlled. Echo cancellers are required when one-way delay exceeds 25milliseconds. For more information about echo cancellation, see the "Echo Adjustment" section on page 9. These ranges are for total end-to-end delay, not just the delay on the WAN side of a connection. For more information about developing a delay budget, refer to the "Building the Delay Budget" section of the *Understanding Delay in Packet Voice Networks* white paper.

Table 46 Recommended Delay Ranges

ITU Recommdation	Private Network Recommendation	Description
0 to 150 milliseconds	0 to 200 milliseconds	Acceptable for most user appliations
150 to 400 milliseconds	200 to 250 milliseconds	Acceptable if administrators are aware of the transmission time and its impact on quality.
Above 400 milliseconds	Above 250 milliseconds	Unacceptable in most situations.

Packet Loss

Packet loss is a very common cause of voice quality problems on an IP network. In a properly designed network, packet loss should be near zero. Voice codecs can tolerate some degree of packet loss without a dramatic effect on voice quality. Voice packets should be tagged for zero packet loss. On converged voice/data networks, the quantity of the voice calls and link bandwidth should be limited. The bandwidth should be guarenteed for the voice calls by giving priority to voice traffic. Packet loss can have an impact voice quality in several ways:

- Packet loss can grow as call volume increases
- Fax and modem transmissons are greatly affected by packet loss

A common cause of packet loss is duplex mismatching. This occurs when one side of the connection is set up for full duplex and the other is set up for half duplex. Look at the link between the two endpoints experiencing the packet loss and check the speed and duplex settings for each side.

Although duplex mismatch is common, there are many other opportunities for packet loss in the network. To find the cause of these packet drops, check each interface between two endpoints by using the **show interface** command. If dropped packets are showing up on any interface, the link might be oversubscribed or there might be other traffic on this link that you are not expecting. In this case, use a network analyzer (or "sniffer") to check what traffic might be congesting the link. For more information about network analyzers, see the "Network Analyzers" section on page 178.

Jitter Adjustment

Delay can cause unnatural starting and stopping of conversations, but variable-length delays (also known as *jitter*) can cause a conversation to break and become unintelligible. Jitter is not usually a problem with PSTN calls because the bandwidth of calls is fixed and each call has a dedicated circuit for the duration of the call. However, in VoIP networks, data traffic might be bursty, and jitter from the packet network can become an issue. Packets from the same conversation can arrive at different interpacket intervals, especially during times of network congestion, which can disrupt the steady, even delivery needed for voice calls. Cisco voice gateways have built-in jitter buffering to compensate for a certain amount of jitter; the **playout-delay** command can be used to adjust the jitter buffer.

Normally, the defaults in effect are sufficient for most networks. However, a small playout delay from the jitter buffer can cause lost packets and choppy audio, and a large playout delay can cause unacceptably high overall end-to-end delay.

Note For Cisco IOS Release 12.1(5)T and later, in most cases playout delay should be configured in dial-peer configuration mode on the VoIP dial peer that is on the receiving end of the voice traffic that is to be buffered. This dial peer senses network conditions and relays them to the DSPs, which adjust the jitter buffer as necessary. When multiple applications are configured on the gateway, playout delay should be configured in dial-peer configuration mode. When there are numerous dial peers to configure, it might be simpler to configure playout delay on a voice port. If there are conflicting playout delay configurations on a voice port and also on a dial peer, the dial-peer configuration takes precedence.

Jitter is more likely to occur on low-speed links because even a single packet in the queue on a low-speed link can dramatically affect the amount of time a voice packet needs to wait in the queue before being transmitted. Jitter can be an even bigger problem if you do not have priority queuing, such as low latency queuing (LLQ), enabled or configured correctly on your WAN connections. For more information about LLQ, see the *Cisco IOS Quality of Service Solutions Configuration Guide*. For information about troubleshooting VoIP over Point-to-Point Protocol (PPP) links with QoS for low latency queueing, refer to *VoIP over PPP Links with Quality of Service (LLQ / IP RTP Priority, LFI, cRTP),* document ID 7111.

Low-speed links also require special considerations when data traffic is also present to ensure that a large data packet does not cause excessive jitter. Generally on WAN links that are 768 kbps or slower, you should use some form of fragmentation and interleaving to ensure that large data packets do not starve smaller voice packets. Even with LLQ enabled, the voice packet must wait if it arrives when a data packet is in the process of being transmitted.

To configure the playout delay jitter buffer, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice-port** *slot*/*port*:ds*0-group-number*
4. **playout-delay mode** {**adaptive** | **fixed**}
5. **playout-delay** {**nominal** *value* | **maximum** *value* | **minimum** {**default** | **low** | **high**}}
6. **exit**

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. Enter your password if prompted.
Step 2	**configure** {**terminal** | **memory** | **network**} **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice-port** *slot/port*:ds*0-group-number* **Example:** `Router(config)# voice-port 1/0:0`	Enters voice-port configuration mode on the selected slot, port, and DS-0 group. • Each defined DS-0 group number is represented on a separate voice port. This allows you to define individual DS-0s on the digital T1/E1 card.
Step 4	**playout-delay mode** {**adaptive** | **fixed**} **Example:** `Router(config-voiceport)# playout-delay mode adaptive`	Determines the mode in which the jitter buffer operates for calls on this voice port. • **adaptive**—Adjusts the jitter buffer size and amount of playout delay during a call based on current network conditions. This is the default. **Note** Cisco recommends using the adaptive mode playout buffer as a place to begin checking QoS issues. Changing the playout buffers is an advanced technique that should be done last after all other QOS methods have been explored. • **fixed**—Defines the jitter buffer size as fixed so that the playout delay is not adjusted during a call. A constant playout delay is added.
Step 5	**playout-delay** {**nominal** *value* | **maximum** *value* | **minimum** {**default** | **low** | **high**}} **Example:** `Router(config-voiceport)# playout-delay nominal 0`	Tunes the playout buffer to accommodate packet jitter caused by switches in the WAN. The keywords and arguments are as follows: • **nominal**—Defines the amount of playout delay applied at the beginning of a call by the jitter buffer in the gateway. In fixed mode, this is also the maximum size of the jitter buffer throughout the call. • *value*—Specifies the range which depends on the type of DSP and configured codec complexity. For medium codec complexity, the range is from 0 to 150 ms. For high codec complexity and DSPs that do not support codec complexity, the range is from 0 to 250 ms.

	Command	Purpose
		• **maximum** (adaptive mode only)—Specifies the jitter buffer's upper limit (80 ms), or the highest value to which the adaptive delay is set. • **minimum** (adaptive mode only)—Specifies the jitter buffer's lower limit (10 ms), or the lowest value to which the adaptive delay is set. • **default**—Specifies 40 ms.

Step 6	`exit`
	
	Example:
	`Router(config-voiceport)# exit`
		Exits voice-port configuration mode.

For more information about jitter, refer to *Understanding Jitter in Packet Voice Networks (Cisco IOS Platforms)*, document ID 18902.

Echo Adjustment

Echo is a common problem, but can sometimes be difficult to troubleshoot. Echo exists in almost any telephone network but problems are more apparent in packet networks. Echo is perceived as a problem when all of the following conditions are true:

- Signal leakage between analog transmit (Tx) and receive (Rx) paths. This leakage is in one of these forms of echo:
  - Acoustic echo—Acoustic echo is caused by poor acoustic isolation between the earpiece and the microphone in handsets and hands-free devices.
  - Hybrid echo—Hybrid echo is caused by an impedance mismatch in the hybrid circuit, such as a two-wire to four-wire interface. This mismatch causes the Tx signal to appear on the Rx signal.
- Perceptible delay between when the originating signal is transmitted and when the echo returns. In most telephones, sidetone helps mask some of the echo. Echos must be delayed by at least 20 milliseconds to be perceived.
- Sufficient echo amplitude. If the amplitude of the echo is low, it can go unnoticed.

The packet segment of the voice connection introduces a significant delay, typically 30 ms in each direction. The introduction of delay causes echoes generated on analog tail circuits that were normally indistinguishable from the side tone to be perceived by the user. The delay introduced by packet voice is unavoidable, therefore, the voice gateways must prevent the echo.

For a description about echo cancellation features and how echo cancellation is implemented on different Cisco platforms and Cisco IOS software releases, refer to the "Information about Echo Cancellation" section in *Voice Port Configuration*.

To configure echo cancellation, refer to the "How to Configure the Extended G.168 Echo Canceller" section in *Voice Port Configuration*.

When troubleshooting echo problems, see the following:

- Determine Where Echo Is Occurring, page 10

Determine Where Echo Is Occurring

When troubleshooting echo, first determine who is being affected by the echo problems.

PSTN Phone Users Hears Echo

In this case, a PSTN phone user hears echo, which is caused by acoustic coupling between the earpiece and the microphone in the IP phone handset. The solution is to use a load ID on the IP phone, which includes echo suppression on the handset and headset. Available load IDs only include echo cancellation on the speaker phone.

IP Phone User Hears Echo

In this case, an IP phone user hears echo caused by hybrids in a PSTN network. The solution is to configure and verify echo cancellation operation on a Cisco IOS gateway. The echo canceller in the voice gateway cancels the echo heard by the IP phone user.

Troubleshooting Echo in Cisco IOS Gateways

It is important to make sure that the echo canceller has enough information to distinguish between echo and voice conversation. The following parameters control the distinction:

- Input level—Signal input gain is performed before the echo canceller has detected the echo.
- Output level—Signal output attenuation is performed after the echo canceller has seen the original output signal.
- Echo canceller coverage—The amount of time the echo canceller retains a signal that has been output. This parameter must be set to a value greater than the time the echo needs to return to the gateway.

To eliminate echo, perform the following steps.

SUMMARY STEPS

1. Verify echo cancellation.
2. Configure echo canceller coverage.
3. Measure the echo and adjust the echo signal level.
4. Verify the impedance configured in the voice port.

Step 1 Verify that echo cancellation is enabled on the voice port. Echo cancellation is enabled by default on most platforms in most Cisco IOS releases.

```
Gateway(config-voiceport)# echo-cancel

  coverage   Echo Cancel Coverage
  enable     Echo Cancel Enable
```

Note You must enter the **shut**, then the **no shut** commands on the voice port for the changes to take effect

Step 2 Configure the echo canceller coverage to a value greater than the time the echo needs to return to the gateway. It must be long enough to cover the worst case for your environment, but not longer.

```
Gateway(config-voiceport)# echo-cancel coverage

  16  16 milliseconds echo canceller coverage
  24  24 milliseconds echo canceller coverage
  32  32 milliseconds echo canceller coverage
  8   8 milliseconds echo canceller coverage
```

Note You must enter the **shut** and **no shut** commands on the voice port for the changes to take effect.

The default coverage is set to 8 ms, but it can be increased up to 64 ms, depending on the Cisco IOS software release. If the PSTN delay (tail length) is more than 32 ms, echo cancellers in Cisco IOS gateways cannot cancel the echo.

Step 3 Measure the echo and adjust the echo signal level as required.

Insufficient echo return loss (ERL) to handle the echo might cause the following problems:

- Echo canceller is cancelling but not enough to make echo inaudible. If the ERL value is too low, the total ERL seen by the IP network Acombined (ACOM) might be insufficient to suppress the echo. ERL should be approximately 20 dB (at least 15 dB).

  ACOM is the total ERL seen across the incoming and outgoing terminals of the echo canceller. The incoming terminal is equal to the signal sent into the ECAN toward the PSTN (voice), and the outgoing terminal is equal to the signal coming out of the ECAN toward the IP network (echo). ACOM is the sum of the ERL, plus ERLE, or the total ERL seen by the network. ACOM (Total loss) is equal to the ERL (tail loss) plus ERLE (ECAN loss).

- Echo canceller is not canceling. If the ERL value is too low, the echo signal returning to the gateway might be too loud (within 6 dB of the talker signal), causing the echo canceller to consider it as voice (double-talk) instead of echo. As a consequence, the echo canceller does not cancel it. ERL should be approximately 6 dB or higher for the echo canceller to engage. In 12.2(13)T, you can configure this ERL level.

To prevent these problems, you need to measure the ERL and signal levels, and adjust the signal levels on the Cisco IOS gateway based on the results. See the"Measuring QoS" section on page 2 for more information about measuring QoS. See the "Measuring Echo in Cisco IOS Gateways" section on page 12 for more information about measuring ERL. You can adjust these levels by configuring positive values for output attenuation and negative values for input gain. Input gain is performed before the echo canceller has seen the echo signal, and output attenuation is performed after the echo canceller has seen the original output signal.

```
voice-port 1/1:15
  input gain -3
  output attenuation 3
```

Note You must enter the **shut** and **no shut** command on the voice port for the changes to take effect.

Note In Cisco IOS Release 12.2(1) and later, output attenuation can be set to a negative value, which amplifies the output signal.

Step 4 An impedance mismatch can cause echo if both sides are not configured identically. Verify the impedance configured in the voice port and modify it if needed. A default of 600 ohms is consistent with most lines on the PSTN and PBXs.

```
Gateway(config-voiceport)# impedance

  600c 600 Ohms complex
  600r 600 Ohms real
  900c 900 Ohms complex
  complex1 complex 1
  complex2 complex 2
```

Measuring Echo in Cisco IOS Gateways

If you have a phone number to which echo is reproducible, using a Cisco IP Phone 7960 or Cisco IP Phone 7940, perform the following steps to generate test tones and measure the echo return loss (ERL).

SUMMARY STEPS

1. Enable the tone generator.
2. Place a call to the source of echo.
3. Press the ***i*** button twice.
4. Set the input and output levels to 0 dB gain/attenuation.
5. View the input and output levels for your call.
6. Configure 1 dB of attenuation in each direction.
7. Configure 3 dB of attenuation in each direction.
8. Change the coverage to 32 ms.

Step 1 To enable the tone generator, type **3 on the Cisco IP Phone 7960 or 7940 keypad while the phone is not on a call. This enables the Tone softkey for as long as this phone is registered to Cisco CallManager.

Note From Load #P0030301ESP5, you need to unlock the phone first, then press **3. If you press **# **3 consecutively you reset the phone (because of the **#** sequence). Therefore, you need to press **#, make a call, then press **3.

Step 2 Place a call to the source of echo.

Step 3 After the call is established, press the ***i*** button twice. This brings up the statistics for the call.

Note If pressing **3 worked, you should have a Tone softkey available. Press the Tone softkey and the phone begins to generate a 1004 Hz tone at -15 dB. The only way to stop the tone is to end the call.

Step 4 Make sure you start with a cleared configuration by setting the input and output levels to 0 dB gain/attenuation. After the tone is being generated, keep the call up, then perform the remaining steps to measure the decibel levels.

Step 5 Use the **show call active voice** command to view the input and output levels for your call. The following is a call to a loopback number that simply echoes back whatever is sent with no attenuation.

```
Gateway# show call active voice

  ! snip
```

```
  OutSignalLevel=-15
  InSignalLevel=-15
  ERLLevel=25

  ! snip
```

The test tone in this example is output as -15 and is looped back with 0 dB loss. Therefore, the tone is coming back at -15 dB. The ERL value in the example has no significance because the echo canceller does not consider the input signal to be echo.

Note The OutSignalLevel shows the value of the level after the output attenuation has been applied to the signal. The InSignalLevel shows the value of the level after the input gain has been applied.

Step 6 If you configure 1 dB of attenuation in each direction, the resulting levels decrease, as shown in the following configuration examples:

```
voice-port 1/1:15
  input gain -1
  output attenuation 1

Gateway#show call active voice

  ! snip

  OutSignalLevel=-16
  InSignalLevel=-17
  ERLLevel=11

  ! snip
```

Note Notice that the OutSignalLevel is -16 because the -15 dB signal is attenuated by 1 dB. The InSignalLevel is -17 dB, the result of the -1 input gain. The ERL is 11 dB in the example, but it is actually 2 dB; the echo canceller still does not acknowledge the input signal as echo.

Step 7 If you configure 3 dB of attenuation in each direction the resulting levels are shown in the following examples:

```
voice-port 1/1:15
  input gain -3
  output attenuation 3

Gateway#show call active voice

  ! snip

  OutSignalLevel=-18
  InSignalLevel=-21
  ERLLevel=6

  ! snip
```

Note Notice that the OutSignalLevel is -18 because the -15 dB signal is attenuated by 3 dB. The InSignalLevel is -21 dB as a result of the input gain of -3. The expected ERL of 6 dB is now correct.

Step 8 If you keep the same 3 dB in each direction, but change the coverage to 32 ms, the resulting levels are shown in the following examples.

```
voice-port 1/1:15
  input gain -3
  output attenuation 3
  echo-cancel coverage 32

Gateway#show call active voice

  ! snip

  OutSignalLevel=-18
  InSignalLevel=-21
  ERLLevel=6

  ! snip
```

The levels look the same as the previous result, however the echo canceller takes a few more seconds longer to converge. Do not set the echo cancel coverage higher than necessary.

For more information about troubleshooting echo problems, refer to the following documents:

- *Troubleshooting Echo Problems between IP Phones and Cisco IOS Gateways*, document ID 19640
- *Echo Analysis for Voice over IP* white paper, at the following website:

  http://www.cisco.com/en/US/tech/tk652/tk701/technologies_white_paper09186a00800d6b68.shtml

Voice Level Adjustment

Voice levels can require adjustment for a variety of reasons. The voice on a call might be too loud or quiet, or callers might have an issue with comfort noise generated when using voice activity detection (VAD). This section contains informaton about the following:

- Input and Output Levels, page 14
- Voice Activity Detection Level, page 16

Input and Output Levels

As much as possible, it is desirable to achieve a uniform input decibel level to the packet voice network. Doing so eliminates or at least reduces any voice distortion because of incorrect input and output decibel levels. Adjustments to levels might be required by the type of equipment connected to the network or by local country-specific conditions.

Incorrect input or output levels can cause echo, as can an impedance mismatch. Too much input gain can cause clipped or fuzzy voice quality. If the output level is too high at the remote router's voice port, the local caller hears an echo. If the local router's voice port input decibel level is too high, the remote side hears clipping. If the local router's voice port input decibel level is too low or the remote router's output level is too low, the remote side voice can be distorted at a very low volume, and DTMF signaling might be missed.

Use the **input gain** and **output attenuation** commands to adjust voice levels, and the **impedance** command to set the impedance value to match that of the voice circuit to which the voice port connects.

To change parameters related to voice levels, enter the following commands.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice-port** *slot*/*port*:d*s0-group-number*
4. **input gain** *value*
5. **output attenuation** *value*
6. **impedance** {**600c** | **600r** | **900c** | **complex1** | **complex2**}
7. **exit**

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure** {**terminal** | **memory** | **network**} **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice-port** *slot/port*:d*s0-group-number* **Example:** `Router(config)# voice-port 1/0:0`	Enters voice-port configuration mode on the selected slot, port, and DS-0 group. • Each defined DS-0 group number is represented on a separate voice port. This allows you to define individual DS-0s on the digital T1/E1 card.
Step 4	**input gain** *value* **Example:** `Router(config-voiceport)# input gain 0`	Specifies, in decibels, the amount of gain to be inserted at the receiver side of the interface, increasing or decreasing the signal. • After an input gain setting is changed, the voice call must be disconnected and reestablished before the change takes effect. • The *value* argument is any integer from –6 to 14. The default is 0.

	Command	Purpose
Step 5	**output attenuation** *value* **Example:** `Router(config-voiceport)# output attenuation -6`	Specifies the amount of attenuation in decibels at the transmit side of the interface, decreasing the signal. • A system-wide loss plan can be implemented through the use of the **input gain** and **output attenuation** commands. • The default value for this command is based on the assumption that a standard transmission loss plan is in effect, meaning that normally there must be –6 dB attenuation between phones. • The *value* argument is any integer from –6 to 14. The default is 0.
Step 6	**impedance** {**600c** | **600r** | **900c** | **complex1** | **complex2**} **Example:** `Router(config-voiceport)# impedance 600r`	Specifies the terminating impedance of a voice port interface, which needs to match the specifications from the specific telephony system to which it is connected. The following values are supported: • **600c**—Specifies 600 ohms complex • **600r**—Specifies 600 ohms real (default) • **900c**—Specifies 900 ohms complex • **complex1**—Specifies Complex 1 • **complex2**—Specifies Complex 2
Step 7	**exit** **Example:** `Router(config-voiceport)# exit`	Exits voice-port configuration mode.

Voice Activity Detection Level

Another way to reduce clipped or fuzzy speech is to change the VAD level. Sometimes VAD cuts the sound too early and the speaker's voice is clipped. Use the **vad** and **no vad** commands in dial-peer configuration mode to enable and disable VAD.

With VAD, voice data packets fall into three categories: speech, silence, and unknown. Speech and unknown packets are sent over the network; silence packets are discarded. The sound quality is slightly degraded with VAD, but the connection monopolizes much less bandwidth. If you use the **no** form of this command, VAD is disabled and voice data is continuously sent to the IP backbone.

When the **aggressive** keyword is used, the VAD noise threshold is reduced from -78 to -62 dBm. Noise that falls below the -62 dBm threshold is considered to be silence and is not sent over the network. Additionally, unknown packets are considered to be silence and are discarded.

You can also change the amount of time that VAD waits for silence. Use the **voice vad-time** command in global configuration mode to change the minimum silence detection time. For example, setting the VAD time to 3000 allows 30 seconds of silence before VAD is initiated.

On gateways that use MGCP, the call agent controls the amount of VAD. For H.323 gateways, VAD must be disabled on the gateway itself. VAD is enabled by default on H.323 gateways. To disable VAD, you must match a VoIP dial peer that has the **no vad** command configured.

Some codecs, such as G.729b and G.729ab, have VAD built into them and cannot be diabled. If you do not want to use VAD, use the standard G.729 or G.729a codecs instead.

Hissing or Static While You Are Using Voice Activity Detector

The VAD detects silence periods in the voice signal and temporarily discontinues transmission of the signal during these periods. This saves bandwidth and allows the far-end to adjust its jitter buffer. The downside is that the far-end phone has to generate its own signal to play to the listener during silence periods. Usually comfort noise is played out to the listener to mask the absence of an audio signal from the far-end. Comfort noise is usually modeled on the far-end noise so that there is not a stark contrast between the two. Sometimes there can be unwanted noise generated by VAD. To troubleshoot hissing or static issues that might be because of the VAD, refer to *Troubleshooting Hissing and Static*, document ID 22388.

Voice Call Tuning

The Voice Call Tuning feature monitors the interface between Cisco IOS software and a system's DSPs in real time and reports status on the following: packet flow, DSP state, echo-cancellation state, and jitter state. The feature also allows you to manipulate echo-cancellation and jitter-buffer parameters in real time.

The feature is part of the standard product and is supported under normal Cisco Technical Assistance Center (TAC) procedures. No new code is required and there are no configuration tasks for this feature. It is backward compatible with older versions of Cisco IOS software, Cisco VCWare, and Cisco voice DSP code base (DSPWare).

You should understand the following concepts to use this feature:

- Restrictions for Voice Call Tuning, page 17
- Information About Voice Call Tuning, page 17
- How to Verify Voice Call Tuning Functionality, page 19
- Voice Call Tuning Configuration Examples, page 19

Restrictions for Voice Call Tuning

If you wish to use this feature on a Cisco AS5300 platform, refer to the *Combined Version Release Notes for Cisco VCWare on Cisco AS5300 Universal Access Servers/Voice Gateways*. That document tells you which Cisco VCWare to use with this feature.

Information About Voice Call Tuning

To configure the Voice Call Tuning feature, you should understand the following concepts:

- Packet-Flow Detection, page 18
- DSP State, page 18
- Echo-Cancellation State, page 18

- Jitter-Buffer Parameters, page 18

Packet-Flow Detection

The Voice Call Tuning feature allows you to determine whether packets are flowing in each direction while a call is in progress.

The analog signal from the telephone is digitized into pulse code modulation (PCM) signals by the voice coder-decoder (codec). The PCM samples are then passed to the compression algorithm which compresses the voice into a packet format for transmission across the WAN. On the far side of the cloud similar functions are performed but in reverse order.

Depending on how the network is configured, the router or gateway can perform both the codec and compression functions or only one of them. For example, if an analog voice system is used, then the router or gateway performs the codec function and the compression function.

DSP State

The Voice Call Tuning feature allows you to check whether the DSP servicing an active call is functioning properly. Explicit configuration of a periodic ping message with both the period and timeout parameters configurable catches DSPs that are in a hung state as soon as possible. A failure can then signal this event to the console log or as a triggering event to a logging functionality. The DSP can also send periodic informational messages like jitter measures, echo canceller measures, and alarms to alert you if there is an impending problem.

Echo-Cancellation State

Echo is the sound of your own voice reverberating in the telephone receiver while you are talking. When timed properly, echo is not a problem in the conversation; however, if the echo interval exceeds approximately 25 ms, it can be distracting to the speaker. Echo is controlled by ECs. By design, ECs are limited by the total amount of time they wait for the reflected speech to be received, which is known as an echo tail. The echo tail is normally 32 ms.

In the traditional telephony network, echo is generally caused by an impedance mismatch when the four-wire network is converted to the two-wire local loop. Echo cancellation is required because of packet network latency.

Echo cancellation is implemented in DSP firmware on the gateways and is independent of other functions implemented in the DSP (the DSP protocol and compression algorithm). In voice packet-based networks, ECs are built into the low-bit-rate codecs and are operated on each DSP.

This feature gives you the ability to change echo canceller parameters while a call is in progress. Audible effects are immediately noticed, aiding in problem determination and resolution. You can change the echo canceller parameters (tail length and echo return loss (ERL) threshold) in the field.

Jitter-Buffer Parameters

Jitter is a variation in the delay of received packets. At the sending side, packets are sent in a continuous stream with the packets being spaced evenly apart. Due to network congestion, improper queuing, or configuration errors, this steady stream can become erratic, or the delay between and two packets can vary instead of remaining constant.

This feature gives you the ability to change jitter-buffer parameters while a call is in progress. Audible effects are immediately noticed, aiding in problem determination and resolution. You can change the jitter buffer parameters (delay, size, fixed state, or adaptive state) in the field.

How to Verify Voice Call Tuning Functionality

There are no configuration tasks for this feature. However, you can verify that the Voice Call Tuning feature is operating on your system by performing the following tasks:

- Use the **show vfc version** command to show the version of the software that resides on your voice feature card (VFC). This command shows information in the output of the **show vfc version vcware** and **show vfc version dspware** commands that indicates whether the Cisco VCWare or DSPWare is compatible with the Cisco IOS image.
- Use the **show voice call** command to show the real-time call status for voice ports, including packet flow indication, DSP state, echo canceller state, and jitter state. If you enable **terminal monitor** before doing this, you can also see the DSP and playout buffer statistics, overflow specifications, late packets, lost packets, and "concealment 9," in which the DSP invents a packet using interpolation.
- Use the **show voice dsp** command to show the status of all DSP voice channels when abnormal behavior in the DSP voice channels occurs.
- Use the **test call id** command to manipulate echo canceller and jitter-buffer parameters in real time. You can use this command with the extended G.168 echo canceller, which allows you to configure the voice card in a router individually, or with the Cisco G.165 echo canceller, which allows you to configure the router as a whole. Messages are visible in the command output when either an extended-only or a standard-only echo cancellation is requested, as in the following example:

  ```
  Extended echo canceller not active for CallID callID
  Basic echo canceller not active for CallID callID
  ```

Voice Call Tuning Configuration Examples

There are no specific configuration examples for the Voice Call Tuning feature, but you can verify that the Voice Call Tuning feature is operating on your system by performing the following tasks:

- Use the **show vfc version** command to show the version of the software that resides on your VFC. This command was modified to include new information in the output of the **show vfc version vcware** and **show vfc version dspware** commands. Messages are output if the Cisco VCWare or DSPWare is not compatible with the Cisco IOS image. The new information is advisory only, so there is no action taken if the software is compatible or incompatible.

  Cisco IOS Release 12.2(13)T adds new information to the output of the **show vfc version vcware** and **show vfc version dspware** commands. Messages are output if the Cisco VCWare or DSPWare is not compatible with the Cisco IOS image. The new information is advisory only, so there is no action taken if the software is compatible or incompatible.

  If the versions detected fall within the defined criteria and are compatible, nothing is output at bootup time. A confirmation line is output when the **show vfc version vcware** and **show vfc version dspware** commands are used, as in the following example:

  ```
  Router# show vfc 1 version vcware

  Voice Feature Card in Slot 1:
  VCWare Version     : 7.35
  ROM Monitor Version: 1.3
  ```

```
     DSPWare Version    : 3.4.46L
     Technology         : C549
VCWare/DSPWare version compatibility OK
```

- Use the **show voice call** command to show the real-time call status for voice ports, including packet flow indication, DSP state, echo canceller state, and jitter state. This command was modified with the **status**, *call-id*, and **sample** *sample-period* command options. This command is available on all voice platforms.

  You can use the *call-id* argument to identify active calls. If the *call-id* argument is omitted, the enquiry shows all active voice calls. In the following example, a list of all active calls with relevant identifying information is shown, as in the following example:

```
Router# show voice call status

CallID    CID    ccVdb       Port     DSP/Ch  Called #   Codec    Dial-peers
0x3       11D4   0x62972834  1/0/0    1/1      10001     g711ulaw 1/2
0x4       11D4   0x62973AD0  1/0/1    2/1     *10001     g711ulaw 2/1
0xA       11DB   0x62FE9D68  1/1/0    3/1     *2692      g729r8   0/2692
2 active calls found
```

- Use the **test call id** command to manipulate the echo canceller and jitter buffer parameters in real time. You can use this command with the extended echo canceller, which allows you to configure the voice card in a router individually, or with the standard echo canceller, in which the configuration occurs implicitly on the router.

  The following example shows query output from a Cisco AS5350 universal gateway using the NextPort version of the standard echo canceller:

```
Router# test call id 99 ?

  echo-canceller  test echo canceller on an active voice call
  playout-delay   test playout-delay parameters
```

Troubleshooting Voice Technologies

Troubleshooting H.323 Interfaces to the IP Network

This chapter provides troubleshooting information for the H.323 standard from the International Telecommunication Union Telecommunication Standardization Sector (ITU-T), of the following:

- Cisco H.323-compliant gatekeeper
- Cisco H.323-compliant gateway
- Cisco H.323-compliant features

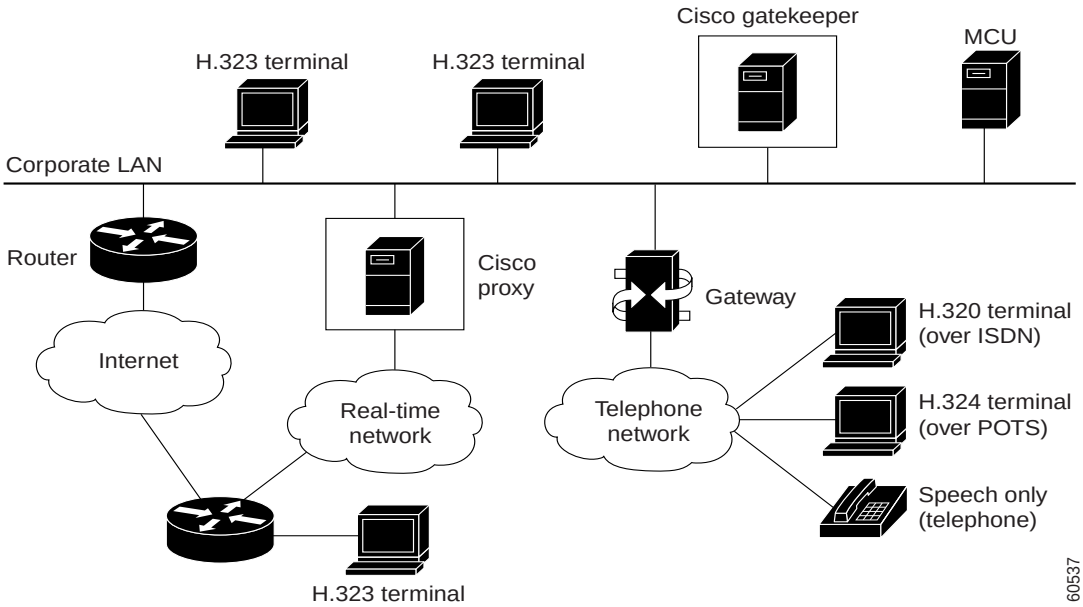
Cisco IOS software complies with the mandatory requirements and several of the optional features of the H.323 Version 2 specification. The chapter contains the following sections:

H.323 is a peer-to-peer protocol. H.323 uses messages similar to Q.931 messages to communicate between endpoints. Q.931 cause codes can be found in "Cause Codes and Debug Values".

Figure 38 shows a typical H.323 network. For detailed information on H.323, refer to *Cisco IOS Call Control Technologies (H.323, SIP, and xGCP)*.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Figure 38 Gatekeeper in an H.323 Network



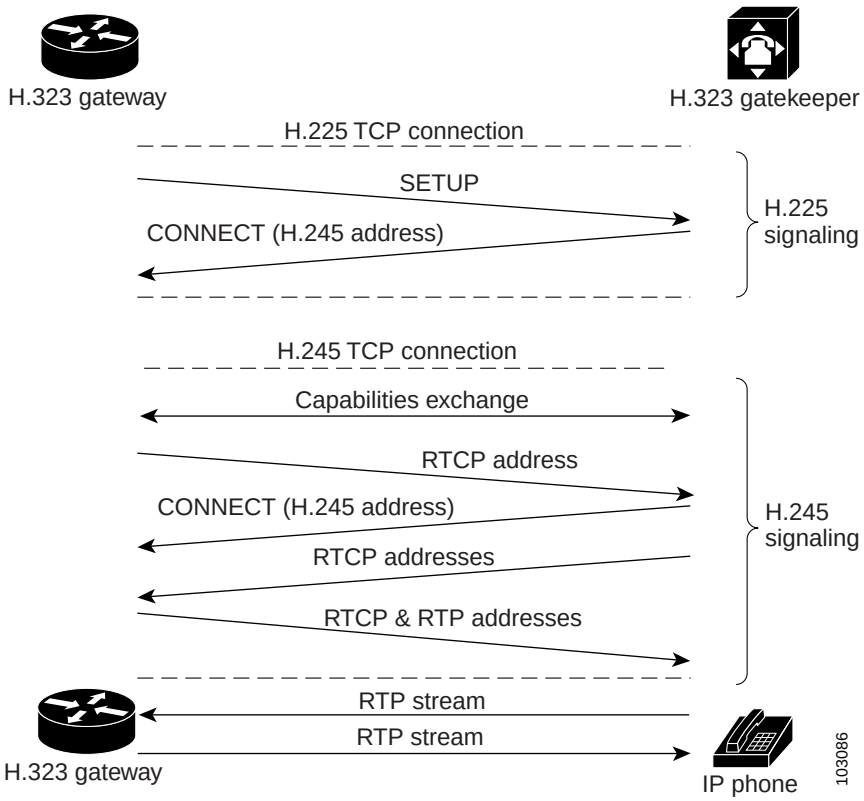
H.323-Related Standards

Because several standards are involved with setting up a call through an H.323 device, it is important to understand the role each protocol plays to be able to effectively troubleshoot such a call. The following sections describe these standards:

- H.225 Signaling, page 3
- H.245 Signaling, page 4

Figure 39 shows the basic call flow for an H.323 call.

Figure 39 H.323 Call Flow



H.225 Signaling

H.225 is used for call control functions. H.225 is very similar to Q.931. H.225 signaling is carried over a TCP connection. H.323 devices listen to port 1720 for incoming connections and show messages that might be used in an H.225 call setup, such as those shown in Table 47. Each message can contain one or more information element (IE). Each IE carries a specific piece of information within an H.225 message. For a complete list of IEs and the coding for each, refer to the ITU-T Q.931 specification.

Table 47 H.225 Call Setup Messages

Message	Description
Setup	This message is sent by a calling H.323 device to indicate its desire to set up a connection to the called device.
Setup Acknowledge	This message typically indicates that the called device wants to do overlap sending and can accept more digits before proceeding with the call.
Call Proceeding	This message is sent by the called device to indicate that it has received all the information it needs to route the call to its destination.
Progress	This message can be sent by an H.323 gateway to indicate a call's progress. You typically see progress messages when internetworking with a non-ISDN network because audio cut-through must be treated differently in this case.

Table 47 H.225 Call Setup Messages (continued)

Message	Description
Alerting	This message might be sent by the called phone to indicate that the called phone is being alerted of the incoming call. That is, the phone is ringing.
Connect	This message is sent by the called device to the calling device to indicate that the call has been accepted or answered.
User Information	This message can be sent to provide additional information. It can be used to provide information for call establishment in the case of overlap sending or miscellaneous call-related information. It can also be used to deliver proprietary features. Cisco IOS gateways use this message to get around the fact that H.323 gateways do not normally provide ringback when a call is transferred and also to generate tone on hold.
Release Complete	This message is sent by a device to indicate the call's release.
Status Inquiry	This message can be used to request call status. Normally the device receiving this message responds with a status message indicating the current call state for the specific call reference.
Status	This message is used to respond to an unknown call signaling message or to a Status Inquiry message.
Information	This message is used to send additional information for a call. For example, when using overlap sending, each digit is sent one at a time in an Information message.
Notify	This message is used to notify a device of a change that has occurred in the call.

H.245 Signaling

H.225 is responsible only for setting up the call and routing it to the proper destination. H.225 does not have any mechanism for exchanging capabilities or setting up and tearing down media streams. The called H.323 device is responsible for sending the IP address and port number that are used to establish the TCP connections for H.245 signaling. This information can be sent by the called device in either the Alerting or Connect message.

When the originating H.323 device receives the IP address and port number for H.245 negotiations, it initiates a second TCP connection to carry out the necessary capabilities exchange and logical channel negotiations. This TCP session is primarily used to do four things:

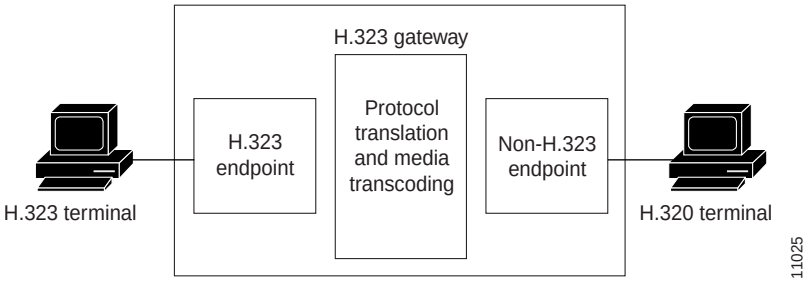
- Master/slave determination—This is used to resolve conflicts that might exist when two endpoints in a call request the same thing, but only one of the two can gain access to the resource at a time.
- Terminal capabilities exchange—This is one of the most important functions of the H.245 protocol. The two most important capabilities are the supported audio codecs and the basic audio calls.
- Logical channel signaling—This indicates a one-way audio stream.With H.323 version 2, it is possible to open and close logical channels in the middle of a call. Because H.245 messages are independent of the H.225 signaling, a call can still be connected in H.225 even if no logical channels are open. This is typical with such features as hold, transfer, and conference.
- DTMF relay—Because voice networks typically do not carry DTMF tones inband because of compression issues, these tones are carried on the signaling channel. Ensure that the type of DTMF relay configured on your gateway is compatible with your gatekeeper.

Troubleshooting Gateways

An H.323 gateway is an endpoint on the LAN that provides real-time communications between H.323 terminals on the LAN and other ITU terminals on a WAN or to other H.323 gateways.

Gateways allow H.323 terminals to communicate with devices that are running other protocols. They provide protocol conversion between the devices that are running different types of protocols. For example, Figure 40 shows a gateway between an H.323 terminal and a non-H.323 terminal.

Figure 40 Gateway Between an H.323 Terminal and an H.320 Terminal



This section contains the following information:

Troubleshooting H.323 Gateway Call Routing and Dial Peers

To troubleshoot H.323 call routing, see the following sections:

Verifying Digits Received and Sent on the POTS Call Leg

Once the on-hook and off-hook signaling are verified to be working correctly, the next step in troubleshooting and debugging a VoIP call is to verify that the correct digits are being received or sent on the voice-port (digital or analog). A dial peer is not matched or the switch (CO or PBX) is not able to ring the correct station if incomplete or incorrect digits are being sent or received. Some commands that can be used to verify the digits received/sent are:

- **show dialplan number**—This command is used to show which dial peer is reached when a particular telephone number is dialed.
- **debug vtsp session**—This command displays information on how each network indication and application request is processed, signaling indications, and DSP control messages.

- **debug vtsp dsp** —This command displays the digits as they are received by the voice-port.
- **debug vtsp all**—This command enables the following debug voice telephony service provider (VTSP) commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**.

show dialplan number

The **show dialplan number** *digit_string* command displays the dial peer that is matched by a string of digits. If multiple dial peers can be matched, they are all shown in the order in which they are matched. The output of this command looks like this:

```
Router# show dialplan number 5000
Macro Exp.: 5000

VoiceOverIpPeer2
        information type = voice,
        tag = 2, destination-pattern = `5000',
        answer-address = `', preference=0,
        group = 2, Admin state is up, Operation
        state is up,
        incoming called-number = `',
        connections/maximum = 0/unlimited,
        application associated:
        type = voip, session-target =
        `ipv4:192.168.10.2',
        technology prefix:
        ip precedence = 5, UDP checksum =
        disabled, session-protocol = cisco,
        req-qos = best-effort,
        acc-qos = best-effort,
dtmf-relay = cisco-rtp,
        fax-rate = voice,
        payload size =  20 bytes
        codec = g729r8,
        payload size =  20 bytes,
        Expect factor = 10, Icpif = 30,
        signaling-type = cas,
        VAD = enabled, Poor QOV Trap = disabled,
        Connect Time = 25630, Charged Units = 0,
        Successful Calls = 25, Failed Calls = 0,
        Accepted Calls = 25, Refused Calls = 0,
        Last Disconnect Cause is "10  ",
        Last Disconnect Text is "normal call
        clearing.",
        Last Setup Time = 84427934.
        Matched: 5000   Digits: 4
        Target: ipv4:192.168.10.2
```

debug vtsp dsp

debug vtsp dsp shows the digits as they are received by the voice port. The following output example shows the collection of DTMF digits from the DSP:

```
Router# debug vtsp dsp
Voice telephony call control dsp debugging is on


!-- ACTION: Caller picked up handset and dialed
!-- digits 5000.
!-- The DSP detects DTMF digits. Digit 5 was
!-- detected with ON time of 130msec.
```

```
*Mar 10 17:57:08.505: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=5,
*Mar 10 17:57:08.585: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=5,
duration=130
*Mar 10 17:57:09.385: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:09.485: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=150
*Mar 10 17:57:10.697: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:10.825: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=180
*Mar 10 17:57:12.865: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:12.917: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=100
```

debug vtsp session

debug vtsp session shows the activity on the voice port. The following output example shows the handset being picked up and dial tone generated:

```
Router# debug vtsp session
Voice telephony call control session debugging is on


!--- <some output have been omitted>
!-- ACTION: Caller picked up handset.
!-- The DSP is allocated, jitter buffers, VAD
!-- thresholds, and signal levels are set.


*Mar 10 18:14:22.865: dsp_set_playout: [1/0/0 (69)]
packet_len=18 channel_id=1 packet_id=76 mode=1
initial=60 min=4 max=200 fax_nom=300
*Mar 10 18:14:22.865: dsp_echo_canceller_control:
[1/0/0 (69)] packet_len=10 channel_id=1 packet_id=66
flags=0x0
*Mar 10 18:14:22.865: dsp_set_gains: [1/0/0 (69)]
packet_len=12 channel_id=1 packet_id=91
in_gain=0 out_gain=65506
*Mar 10 18:14:22.865: dsp_vad_enable: [1/0/0 (69)]
packet_len=10 channel_id=1 packet_id=78
thresh=-38act_setup_ind_ack
*Mar 10 18:14:22.869: dsp_voice_mode: [1/0/0 (69)]
packet_len=24 channel_id=1 packet_id=73 coding_type=1
voice_field_size=80
VAD_flag=0 echo_length=64 comfort_noise=1
inband_detect=1 digit_relay=2
AGC_flag=0act_setup_ind_ack(): dsp_dtmf_mod
e()act_setup_ind_ack: passthru_mode = 0,
no_auto_switchover = 0dsp_dtmf_mode
(VTSP_TONE_DTMF_MODE)


!-- The DSP is put into "voice mode" and dial-tone is
```

```
!-- generated.


*Mar 10 18:14:22.873: dsp_cp_tone_on: [1/0/0 (69)]
packet_len=30 channel_id=1 packet_id=72 tone_id=4
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
```

If you find that the digits are not being sent or received properly, you might need to use either a digit-grabber (test tool) or T1 tester to verify the digits are being sent at the correct frequency and timing interval. If they are being sent incorrectly for the switch (CO or PBX), some values on the router or switch (CO or PBX) might need to be adjusted so that they match and can interoperate. These are usually digit duration and interdigit duration values. If the digits appear to be sent correctly you might want to examine any number translation tables in the switch (CO or PBX) that might add or remove digits.

Verifying End-to-End VoIP Signaling on the VoIP Call Leg

After verifying that voice-port signaling is working properly and the correct digits have been received, verify that the end-to-end VoIP signaling is set up on the VoIP call leg.VoIP signaling involves call control. The following factors explain why call control debugging can become a complex job:

- Cisco VoIP gateways use H.323 signaling to complete calls. H.323 is made up of three layers of call-negotiation and call-establishment: H.225, H.245, and H.323. These protocols use a combination of TCP and UDP to set up and establish a call.
- End-to-end VoIP debugging shows a number of Cisco IOS state machines, and problems with any state machine can cause a call to fail.
- End-to-end VoIP debugging can be very verbose and can create a lot of debug output.

debug voip ccapi inout

The primary command to debug end-to-end VoIP calls is **debug voip ccapi inout**. The output from a sample call debug is shown below.

```
Router# debug voip ccapi inout

*Mar  1 15:35:53.588: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructTDUsrContainer: usrCo
ntainer[0x638C1BF0], magic[FACE0FFF]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDUtilAddDataToUsrContainer: con
tainer=0x638C1BF0, tagID=6, dataSize=16, instID=-1,modifier=1
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructInstanceTDObject: tdO
bject[0x638BC1AC], nxtElem[0x0], magic[0xFACE0FFF] tagID[6], dataLen[16], modif[
1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer: Addin
g tdObject[0x638BC1AC] instID[-1] into container[0x638C1BF0]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDUtilAddDataToUsrContainer: con
tainer=0x638C1BF0, tagID=5, dataSize=276, instID=-1,modifier=1
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructInstanceTDObject: tdO
bject[0x63401148], nxtElem[0x0], magic[0xFACE0FFF] tagID[5], dataLen[276], modif
[1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer: Addin
g tdObject[0x63401148] instID[-1] into container[0x638C1BF0]
```

In the following lines, the call control API (CCAPI) receives the call setup. The called number is **34999**, and the calling number is **55555**. The calling number matches dial peer **10002**.

```
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/cc_api_display_ie_subfields:
*Mar  1 15:35:53.592: cc_api_call_setup_ind:
```

```
*Mar  1 15:35:53.592:  cisco-username=
*Mar  1 15:35:53.596: ----- ccCallInfo IE subfields -----
*Mar  1 15:35:53.596:  cisco-ani=55555
*Mar  1 15:35:53.596:  cisco-anitype=0
*Mar  1 15:35:53.596:  cisco-aniplan=0
*Mar  1 15:35:53.596:  cisco-anipi=0
*Mar  1 15:35:53.596:  cisco-anisi=0
*Mar  1 15:35:53.596:  dest=34999
*Mar  1 15:35:53.596:  cisco-desttype=0
*Mar  1 15:35:53.596:  cisco-destplan=0
*Mar  1 15:35:53.596:  cisco-rdn=
*Mar  1 15:35:53.596:  cisco-rdntype=-1
*Mar  1 15:35:53.596:  cisco-rdnplan=-1
*Mar  1 15:35:53.596:  cisco-rdnpi=-1
*Mar  1 15:35:53.596:  cisco-rdnsi=-1
*Mar  1 15:35:53.596:  cisco-redirectreason=-1
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind: (vdbPtr=0x6
37EC1E0, callInfo={called=34999,called_oct3=0x80,calling=55555,calling_oct3=0x80
,calling_oct3a=0x0,calling_xlated=false,subscriber_type_str=RegularLine,fdest=1,
peer_tag=10002, prog_ind=0,callingIE_present 1, src_route_label=, tgt_route_labe
l= clid_transparent=0},callID=0x637B4278)

*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind:
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind: type 13 , p
rot 0
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: ccCheckClipClir: calling number is: "55555", calling oct3a
 is: 0x0
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: Calling Party number is User Provided
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: Leaving ccCheckClipClir
  calling number is: "55555"
  calling oct3 is:  0x80
  calling oct3a is: 0x0
```

In the next line, **44** is the CallEntry ID.

```
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: Increment ca
ll volume: 0

*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: current call
 volume: 1
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: entry's inco
ming TRUE.
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: is_incoming
is TRUE
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructHashProfileTab: profi
leTable[0x6380E11C], numBuckets[11], numEntries[0]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager: I
nvoking necessary profileTable updaters...
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContain
er: Updating profileTable[0x6380E11C] with objects in container[0x638C1BF0]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContain
er: obtained key[5] for the tag[6]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket: p
rofileTable[0x6380E11C], tdObject[0x638BC1AC]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContain
er: obtained key[0] for the tag[5]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket: p
rofileTable[0x6380E11C], tdObject[0x63401148]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager:
*Mar  1 15:35:53.600: ccTDUtilDumpAllElemInProfileTab: profileTable[0x6380E11C],
 numBuckets[11], numEntries[2]
```

```
*Mar  1 15:35:53.600: Bucket { 0 } ------>0x63401148[0x0,t-5,l-276,d-0x63401168,
m-1,u-56153,g-FACE0FFF]
*Mar  1 15:35:53.604:
*Mar  1 15:35:53.604: Bucket { 5 } ------>0x638BC1AC[0x0,t-6,l-16,d-0x638BC1CC,m
-1,u-56153,g-FACE0FFF]
*Mar  1 15:35:53.604:
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructTDUsrContainer: Contai
ner[0x638C1BF0]
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: not the Vo
IP or MMoIP
*Mar  1 15:35:53.608: //-1/xxxxxxxxxxxx/CCAPI/cc_process_call_setup_ind: (event=
0x63073AA0)
```

In the next line, **45F2AAE28044** is the GUID. The **tag 10002** entry shows that the incoming dial-peer matched the CallEntry ID.

```
*Mar  1 15:35:53.608: //44/45F2AAE28044/CCAPI/cc_process_call_setup_ind: >>>>CCA
PI handed cid 44 with tag 10002 to app "DEFAULT"
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(24=CC_EV_CALL_
SETUP_IND), cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(SSA_EV_CALL_SE
TUP_IND), cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/ssaCallSetupInd:
```

The next line shows CallEntry ID in hexadecimal form, **0x2C** (44 in decimal). The CallID and GUID numbers have been identified. The incoming dial-peer is **10002**.

```
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2C,
context=0x634A430C)

*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: cid(44),
 st(SSA_CS_MAPPING),oldst(0), ev(24)ev->e.evCallSetupInd.nCallInfo.finalDestFlag
 = 1
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: src rout
e label=, tgt route label= tg_label_flag 0x0
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: finalDes
t cllng(55555), clled(34999) tgt_route_label()tg_label_flag 0x0
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: cid(44),
 st(SSA_CS_CALL_SETTING),oldst(0), ev(24)dpMatchPeersMoreArg result= 0
```

For CallEntry ID 44, two dial-peer tags (**10001** and **20002**) were matched with called number **34999**.

```
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaDebugPeers: ssaSetupPe
er cid(44) peer list: tag(10001) called number (34999) tag(20002) called number
(34999)
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaSetupPeer: dialpeer ta
gs in rotary= 10001  20002
```

The next line shows that 5 digits were matched for this dial-peer and no prefix was added. The **encapType (2)** entry indicates a VoIP call.

```
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaSetupPeer: cid(44), de
stPat(34999), matched(5), prefix(), peer(637B0984), peer->encapType (2)
*Mar  1 15:35:53.612: //-1/xxxxxxxxxxxx/CCAPI/cc_can_gateway: Call legs: In=6, O
ut=1
```

The next line shows the voice gateway sending out a call-proceeding message to the incoming call leg with progress indicator of **0x0**.

```
*Mar  1 15:35:53.612: //44/xxxxxxxxxxxx/CCAPI/ccCallProceeding: (callID=0x2C, pr
og_ind=0x0)
```

The next line shows the voice gateway sending out the call-setup request to the outgoing call leg. The dial-peer is **10001** with the incoming CallEntry ID being **0x2C**.

```
*Mar  1 15:35:53.612: //44/xxxxxxxxxxx/CCAPI/ccCallSetupRequest: (Inbound call
= 0x2C, outbound peer =10001, dest=,
        params=0x63085D80 mode=0, *callID=0x63086314, prog_ind = 0callingIE_pres
ent 1)

*Mar  1 15:35:53.612: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.612: ccCallSetupRequest numbering_type 0x80
*Mar  1 15:35:53.612: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.616: ccCallSetupRequest: calling number is:55555

*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: calling oct3a
is:0x0

*Mar  1 15:35:53.616: //-1/xxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: ccCheckClipClir: calling number is: "55555", calling oct3a
 is: 0x0
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: Calling Party number is User Provided
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: Leaving ccCheckClipClir
  calling number is: "55555"
  calling oct3 is:  0x80
  calling oct3a is: 0x0
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: after ccCheckC
lipClir - calling oct3a is:0x0
```

The next line shows that all digits are passed.

```
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: dest pattern 3
4999, called 34999, digit_strip 0
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.616: callingNumber=55555, calledNumber=34999, redirectNumber= d
isplay_info= calling_oct3a=0
*Mar  1 15:35:53.616: accountNumber=, finalDestFlag=1,
guid=45f2.aae2.1571.11cc.8044.95f5.fabb.6b0f
*Mar  1 15:35:53.616: peer_tag=10001
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxx/CCAPI/cc_api_display_ie_subfields:
*Mar  1 15:35:53.616: ccCallSetupRequest:
*Mar  1 15:35:53.616:  cisco-username=
*Mar  1 15:35:53.616: ----- ccCallInfo IE subfields -----
*Mar  1 15:35:53.616:  cisco-ani=55555
*Mar  1 15:35:53.616:  cisco-anitype=0
*Mar  1 15:35:53.616:  cisco-aniplan=0
*Mar  1 15:35:53.616:  cisco-anipi=0
*Mar  1 15:35:53.616:  cisco-anisi=0
*Mar  1 15:35:53.620:  dest=34999
*Mar  1 15:35:53.620:  cisco-desttype=0
*Mar  1 15:35:53.620:  cisco-destplan=0
*Mar  1 15:35:53.620:  cisco-rdn=
*Mar  1 15:35:53.620:  cisco-rdntype=-1
*Mar  1 15:35:53.620:  cisco-rdnplan=-1
*Mar  1 15:35:53.620:  cisco-rdnpi=-1
*Mar  1 15:35:53.620:  cisco-rdnsi=-1
*Mar  1 15:35:53.620:  cisco-redirectreason=-1

*Mar  1 15:35:53.620: //-1/xxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate: (vdbP
tr=0x62EC61A4, dest=, callParams={called=34999,called_oct3=0x80, calling=55555,c
alling_oct3=0x80, calling_oct3a= 0x0, calling_xlated=false,  subscriber_type_str
=RegularLine, fdest=1, voice_peer_tag=10001},mode=0x0)
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 15:35:53.620: ccIFCallSetupRequestPrivate: src route label  tgt route la
bel tg_label_flag 0x0
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:  vdbP
tr type = 1
```

```
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate: (vdbP
tr=0x62EC61A4, dest=, callParams={called=34999, called_oct3 0x80,  calling=55555
,calling_oct3 0x80, calling_oct3a 0x0, calling_xlated=false,  fdest=1, voice_pee
r_tag=10001}, mode=0x0, xltrc=-5)
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
```

In the next line, outgoing CallEntry ID **45** is bound to the same GUID **45F2AAE28044**.

```
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: not incoming
 entry

*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: entry's inco
ming FALSE.
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: is_incoming
is FALSE
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/ccSaveDialpeerTag: (callID=0x2C, d
ialpeer_tag=10001)
*Mar  1 15:35:53.624: //45/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2D, co
ntext=0x634A537C) 0x2D (decimal 45 is the second call leg ID).
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/ccCallReportDigits: (callID=0x2C,
enable=0x0)
```

The voice gateway informs the incoming call leg that digits were forwarded.

```
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/cc_api_call_report_digits_done: (v
dbPtr=0x637EC1E0, callID=0x2C, disp=0)
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(54=CC_EV_CALL_
REPORT_DIGITS_DONE), cid(44), disp(0)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SS
Router#APP:10002:-1/ssaTraceSct: cid(44)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_RE
PORT_DIGITS_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaTraceSct: -cid2(45)st2
(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaDebugPeers: ssaReportD
igitsDone cid(44) peer list: tag(20002) called number (34999)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaReportDigitsDone: call
id=44 Reporting disabled.
*Mar  1 15:35:53.628: //-1/xxxxxxxxxxxx/CCAPI/cc_api_supported_data: data_mode=0
x10082
*Mar  1 15:35:53.628: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_ic_leg_obtained_numbers
: callID=0x2D
```

The next two lines show the IP address of the terminating gateway and indicate that the terminating gateway is reached through Ethernet port 0/0.

```
*Mar  1 15:35:53.628: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: remote IP
is 171.69.85.111
*Mar  1 15:35:53.632: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: hwidb is E
thernet0/0
*Mar  1 15:35:53.632: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: create ent
ry in list: 1
*Mar  1 15:35:53.636: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetInstanceCount: For tagI
D[1] of callID[45]
*Mar  1 15:35:53.636: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessMan
ager: No profileTable set for callID[45]
*Mar  1 15:35:53.636: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetInstanceCount: For tagI
D[2] of callID[45]
*Mar  1 15:35:53.636: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessMan
ager: No profileTable set for callID[45]
```

The next line shows that the voice gateway received a call proceeding message from the terminating gateway. The line after that shows that the voice gateway received a call alert from the terminating gateway.

```
*Mar  1 15:35:53.740: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_proceeding: (vdbPtr=0x
62EC61A4, callID=0x2D,
      prog_ind=0x0)
*Mar  1 15:35:53.740: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_alert: (vdbPtr=0x62EC6
1A4, callID=0x2D, prog_ind=0x0, sig_ind=0x1)
*Mar  1 15:35:53.744: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(21=CC_EV_CALL_
PROCEEDING), cid(45), disp(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_CALL_SETTING)ev(SSA_EV_CALL_PROCEEDING)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaCallProc:
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaIgnore: cid(45), st(SSA_CS
_CALL_SETTING),oldst(1), ev(21)
*Mar  1 15:35:53.744: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(7=CC_EV_CALL_A
LERT), cid(45), disp(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_CALL_SETTING)ev(SSA_EV_CALL_ALERT)
oldst(SSA_CS_CALL_SETTING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:35:53.744: //44/45F2AAE28044/SSAPP:10002:-1/ssaAlert:
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
Router#
```

The voice gateway forwarded a call alert to the originating gateway.

```
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccCallAlert: (callID=0x2C, prog_in
d=0x0, sig_ind=0x1)
Router#
```

The phone is answered at the called number.

```
Router#
!call answered
Router#
```

The voice gateway receives a connect message from the terminating gateway.

```
*Mar  1 15:36:05.016: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_connected: (vdbPtr=0x6
2EC61A4, callID=0x2D), prog_ind = 0

*Mar  1 15:36:05.016: //45/45F2AAE28044/CCAPI/cc_api_call_connected: setting cal
lEntry->connected to TRUE
```

The next line shows that the call accounting starts. The **leg_type=False** message means that message is for an outgoing call. The line that follows shows that AAA accounting is not configured.

```
*Mar  1 15:36:05.016: //45/45F2AAE28044/CCAPI/cc_api_call_connected: calling accounting
start for callID=45 leg_type=0
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/CCAPI/ccCallSetAAA_Accounting: callID=0x
2D, accounting=0
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(8=CC_EV_CALL_C
ONNECTED), cid(45), disp(0)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_ALERT_RCVD)ev(SSA_EV_CALL_CONNECTED)
oldst(SSA_CS_CALL_SETTING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_ALERT_RCVD)oldst2(SSA_CS_CALL_SETTING)
```

```
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaConnect:
*Mar  1 15:36:05.020: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
```

The next lines show a conference being set up between the two call legs **0x2C** and **0x2D**. Bridge complete messages are sent to both the terminating and originating gateways.

```
*Mar  1 15:36:05.020: //44/xxxxxxxxxxxx/CCAPI/ccConferenceCreate: (confID=0x6308
6424, callID1=0x2C, callID2=0x2D, tag=0x0)

*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done: (confID=0x15,
srcIF=0x62EC61A4, srcCallID=0x2D, dstCallID=0x2C, disposition=0, tag=0x0)
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done: (confID=0x15,
srcIF=0x637EC1E0, srcCallID=0x2C, dstCallID=0x2D, disposition=0, tag=0x0)
```

Here, the voice gateway sets up negotiating capability with the originating telephony leg.

```
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (dstVdbPtr=0x62EC
61A4, dstCallId=0x2D, srcCallId=0x2C,
     caps={codec=0x2887F, fax_rate=0xBF, vad=0x3, modem=0x2
           codec_bytes=0, signal_type=3})
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (Playout: mode 0,
 initial 60,min 40, max 300)
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(29=CC_EV_CONF_
CREATE_DONE), cid(44), disp(0)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: cid(44)st(SS
A_CS_CONFERENCING)ev(SSA_EV_CONF_CREATE_DONE)
oldst(SSA_CS_CALL_SETTING)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_CONFERENCING)oldst2(SSA_CS_ALERT_RCVD)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaConfCreateDone:
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/ccCallConnect: (callID=0x2C), prog
_ind = 0
*Mar  1 15:36:05.024: //44/45F2AAE28044/CCAPI/ccCallConnect: setting callEntry->
connected to TRUE

*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaDebugPeers: ssaFlushPe
erTagQueue cid(44) peer list: tag(20002) called number (34999)
*Mar  1 15:36:05.028: //-1/xxxxxxxxxxxx/CCAPI/cc_process_notify_bridge_done: (ev
ent=0x63067FC0)
```

The voice gateway sets up negotiating capability with the terminating VoIP leg.

```
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (dstVdbPtr=0x637E
C1E0, dstCallId=0x2C, srcCallId=0x2D,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2})
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (Playout: mode 0,
 initial 60,min 40, max 300)
```

The capabilities are acknowledged for both call legs.

```
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ack: (dstVdbPtr=0x637E
C1E0, dstCallId=0x2C, srcCallId=0x2D,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2, seq_num_start=2944})
*Mar  1 15:36:05.028: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ack: (dstVdbPtr=0x62EC
61A4, dstCallId=0x2D, srcCallId=0x2C,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2, seq_num_start=2944})

*Mar  1 15:36:05.032: //44/xxxxxxxxxxxx/CCAPI/cc_api_voice_mode_event: callID=0x
2C
*Mar  1 15:36:05.032: //44/45F2AAE28044/CCAPI/cc_api_voice_mode_event: Call Poin
ter =634A430C
*Mar  1 15:36:05.032: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(52=CC_EV_VOICE
```

```
_MODE_DONE), cid(44), disp(0)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
Router#
Router#cid(44)st(SSA_CS_ACTIVE)ev(SSA_EV_VOICE_MODE_DONE)
oldst(SSA_CS_CONFERENCING)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_ACTIVE)oldst2(SSA_CS_ALERT_RCVD)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaIgnore: cid(44), st(SS
A_CS_ACTIVE),oldst(5), ev(52)
Router#
Router#! digit punched
Router#
```

The phone at the terminating gateway enters digit **1**.

```
*Mar  1 15:36:11.204: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPt
r=0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=1, digit_begin_flags=0x0, rtp_timestamp=0x0
    rtp_expiration=0x0, dest_mask=0x2)
*Mar  1 15:36:11.504: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_end: (dstVdbPtr=
0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=1,duration=300,xruleCallingTag=0,xruleCalledTag=0, dest_mask=0x2), dig
it_tone_mode=0
```

The phone at the terminating gateway enters digit **2.**

```
*Mar  1 15:36:11.604: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPt
r=0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=2, digit_begin_flags=0x0, rtp_timestamp=0x0
    rtp_expiration=0x0, dest_mask=0x2)
*Mar  1 15:36:11.904: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_end: (dstVdbPtr=
0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=2,duration=300,xruleCallingTag=0,xruleCalledTag=0, dest_mask=0x2), dig
it_tone_mode=0
Router#
Router#
*Mar  1 15:36:14.476: //-1/xxxxxxxxxxxx/CCAPI/cc_handle_periodic_timer: Calling
the callback, ccTimerctx - 0x628B6330
*Mar  1 15:36:14.476: //-1/xxxxxxxxxxxx/CCAPI/ccTimerStart: ccTimerctx - 0x628B6
330
Router#
Router#!call hung up  The user at the terminating gateway hangs up the call.
Router#
```

The voice gateway receives a disconnect message from the terminating gateway. The cause code is **0x10** which is normal call clearing.

```
*Mar  1 15:36:22.916: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnected: (vdbPtr=
0x62EC61A4, callID=0x2D, cause=0x10)
*Mar  1 15:36:22.920: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(11=CC_EV_CALL_
DISCONNECTED), cid(45), disp(0)
*Mar  1 15:36:22.920: //45/45F2AAE28044/SSAPP:0:21/ssaTraceSct: cid(45)st(SSA_CS
_ACTIVE)ev(SSA_EV_CALL_DISCONNECTED)
oldst(SSA_CS_ALERT_RCVD)cfid(21)csize(2)in(0)fDest(0)
*Mar  1 15:36:22.920: //45/45F2AAE28044/SSAPP:0:21/ssaTraceSct: -cid2(44)st2(SSA
_CS_ACTIVE)oldst2(SSA_CS_ACTIVE)
*Mar  1 15:36:22.920: ssa: Disconnected cid(45) state(5) cause(0x10)
```

The voice gateway begins tearing down the conference and dropping the bridge.

```
*Mar  1 15:36:22.920: //-1/xxxxxxxxxxxx/CCAPI/ccConferenceDestroy: (confID=0x15,
 tag=0x0)
*Mar  1 15:36:22.920: //45/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done: (confID=0
x15, srcIF=0x62EC61A4, srcCallID=0x2D, dstCallID=0x2C, disposition=0 tag=0x0)
*Mar  1 15:36:22.920: //44/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done: (confID=0
```

```
x15, srcIF=0x637EC1E0, srcCallID=0x2C, dstCallID=0x2D, disposition=0 tag=0x0)
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(30=CC_EV_CONF_
DESTROY_DONE), cid(44), disp(0)
*Mar  1 15:36:22.924: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: cid(44)st(SS
A_CS_CONF_DESTROYING)ev(SSA_EV_CONF_DESTROY_DONE)
oldst(SSA_CS_ACTIVE)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:22.924: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_CONF_DESTROYING)oldst2(SSA_CS_ACTIVE)
*Mar  1 15:36:22.924: //45/45F2AAE28044/SSAPP:0:-1/ssaConfDestroyDone:
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/CCAPI/ccCallDisconnect: (callID=0x2C, ca
use=0x10 tag=0x0)
```

The voice gateway stops call accounting on the incoming call, indicated by the **leg_type=True** message. The cause code is then set for the originating leg.

```
*Mar  1 15:36:22.924: //44/45F2AAE28044/CCAPI/ccCallDisconnect: calling accounti
ng start for callID=44 leg_type=1
*Mar  1 15:36:22.924: //44/45F2AAE28044/CCAPI/ccCallDisconnect: existing_cause =
 0x0, new_cause = 0x10
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/CCAPI/cc_api_get_transfer_info: (callID=
0x2C)
*Mar  1 15:36:22.924: //45/xxxxxxxxxxxx/CCAPI/ccCallDisconnect: (callID=0x2D, ca
use=0x10 tag=0x0)
```

The voice gateway stops call accounting for the outgoing call, indicated by the **leg_type=False** message. The cause code is verified for the terminating leg.

```
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: calling accounti
ng start for callID=45 leg_type=0
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: existing_cause =
 0x10, new_cause = 0x10
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: using the existi
ng_cause 0x10
*Mar  1 15:36:22.928: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_transfer_info: (callID=
0x2D)
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_api_icpif: expect factor = 0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/g113_calculate_impairment: (delay=
79,
      loss=0), Io=0 Iq=0 Idte=0 Idd=0 Ie=10 Itot=10
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: the remote
 IP is 171.69.85.111
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: hwidb is E
thernet0/0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: reduce cal
lnum of entry: 0, voip: 0, mmoip: 0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: remove an
entry
*Mar  1 15:36:22.932: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnect_done: (vdbP
tr=0x62EC61A4, callID=0x2D, disp=0, tag=0x0)
*Mar  1 15:36:22.932: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessMan
ager: No profileTable set for callID[45]
*Mar  1 15:36:22.936: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetDataByRef: No tdObject
found in profileTable for tagID[6] of callID[45]
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: not incoming
 entry
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: entry's inco
ming FALSE.
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: is_incoming
is FALSE
*Mar  1 15:36:22.940: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(12=CC_EV_CALL_
DISCONNECT_DONE), cid(45), disp(0)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)
oldst(SSA_CS_ACTIVE)cfid(-1)csize(2)in(0)fDest(0)
```

```
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_DISCONNECTING)oldst2(SSA_CS_CONF_DESTROYING)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaDisconnectDone:
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaAAA_CheckAccounting: accou
nting generation enabled
*Mar  1 15:36:22.940: //45/xxxxxxxxxxxx/CCAPI/ccCallSetAAA_Accounting: callID=0x
2D, accounting=0
*Mar  1 15:36:22.944: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: not the Vo
IP or MMoIP
*Mar  1 15:36:22.948: //44/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnect_done: (vdbP
tr=0x637EC1E0, callID=0x2C, disp=0, tag=0x0)
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: ccFreeRawMsg
Info(0x6307595C)
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: Decrement ca
ll volume counter 1
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: current call
 volume: 0
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: entry's inco
ming TRUE.
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: is_incoming
is TRUE
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: Deleting pro
fileTable[0x6380E11C]
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructTDHashProfileTab: Dest
ructor Profile Table (0x6380E11C)
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructInstanceTDObject: tdOb
ject[0x63401148] tagID[5]
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructInstanceTDObject: tdOb
ject[0x638BC1AC] tagID[6]
*Mar  1 15:36:22.956: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(12=CC_EV_CALL_
DISCONNECT_DONE), cid(44), disp(0)
*Mar  1 15:36:22.956: //44/45F2AAE28044/SSAPP:10002:-1/ssaTraceSct: cid(44)st(SS
A_CS_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)
oldst(SSA_CS_CONF_DESTROYING)cfid(-1)csize(1)in(1)fDest(1)
Router#
*Mar  1 15:36:22.956: //44/45F2AAE28044/SSAPP:10002:-1/ssaDisconnectDone:
```

If the call is failing and the cause appears to be in the VoIP portion of the call setup, you might need to look at the H.225 or H.245 TCP part of the call setup, as opposed to just the UDP portion of the H.323 setup. The commands that can be used to debug the H.225 or H.245 call setup are:

- **debug ip tcp transaction** and **debug ip tcp packet** — These examine the TCP portion of the H.225 and H.245 negotiation. They returns the IP addresses, TCP ports and states of the TCP connections.
- **debug cch323 h225** —This examines the H.225 portion of the call negotiation. Think of this as the Layer1 part of the 3 part H.323 call setup.
- **debug ch323 h245**—This examines the H.245 portion of the call negotiation. Think of this as the Layer2 part of the 3 part H.323 call setup.

For more information about voice troubleshooting, refer to *Troubleshooting and Debugging VoIP Call Basics*, document 14081.

Troubleshooting H.323 Gateway Dial Tone

A common problem encountered in a VoIP network is being unable to break dial tone. The router puts a seizure on the local PBX but when digits are dialed, the dial tone stays. The calling party is unable to pass the DTMF tones or digits to the terminating device, resulting in callers being unable to dial the desired extension or interact with the device that needs DTMF tones such as a voice mail or interactive voice response (IVR) application. This problem can result from a number of reasons such as:

- DTMF tones not being passed
- DTMF tones not being understood
- DTMF tones being passed too distorted to be understood
- Other signaling and cabling issues

In the case of a VoIP call from an originating gateway (OGW) to a terminating gateway (TGW), terminating the call to a telephony device might not be understood. When you are passing DTMF tones through a compressed VoIP audio path, some or part of the dual-tones could become slightly distorted because DSP codecs are designed to interpret human speech, not machine tones. Usually, this does not occur with lesser compression codecs such as G.732 or G.711. But the higher compression codecs might result in distortion of in-band tones. However, Cisco IOS allows the DTMF tones to be passed out-of-band between VoIP gateways using three different techniques. All of these techniques use the H.245 capabilities-exchange (part of H.323v2) to signal to the remote VoIP gateway that a DTMF-tone has been received and the remote VoIP gateway should regenerate it.

Make sure the **dtmf-relay** command is configured under the VoIP dial-peer on both sides. Three different types of DTMF relays can be configured.

```
Router(config-dial-peer)#dtmf-relay ?
  cisco-rtp Cisco Proprietary RTP
  h245-alphanumeric DTMF Relay via H245 Alphanumeric IE
  h245-signal DTMF Relay via H245 Signal IE
```

If the problem persists, try a different setting of the **dtmf-relay** command.

For more information, refer to *Inability To Break Dialtone in a Voice over IP Network*, document 22376.

Troubleshooting H.323 Gateway Busy Tone

This section addresses call progress in-band related issues that arise when you are interworking ISDN and H.323 signaling between VoIP and the PSTN. Challenges arise when Cisco VoIP gateways exchange signaling capabilities with the telco switch. The following are some common problem scenarios and symptoms:

- No DTMF Digits or Audio Passed on VoIP Calls to PSTN or PBX—This occurs when the IP phone user makes a call, is able to hear announcement messages (for example “enter your account number.”) but cannot pass DTMF digits. This symptom applies for both VoIP toll-bypass calls and Cisco IP phone to PSTN/PBX calls.
- No Busy Tone or Announcement Message Received When Placing VoIP Outbound Calls—This occurs when Cisco IP phone (CallManager scenario) or POTS phone (VoIP toll-bypass scenario) does not hear a busy tone or announcement message from the PSTN. This symptom applies for both VoIP toll-bypass calls and Cisco IP phone to PSTN/PBX calls.
- No Busy Tone on Inbound Call from Telephony (ISDN) to Cisco CallManager IP Phone, Cisco IOS Gateway, or Third-Party H.323 Device—This occurs when a call from PSTN through gateway to a Cisco CallManager IP phone, Cisco IOS gateway or third party H.323 device does not hear busy tone when running either an application or two-stage dialing on the originating gateway.

These scenarios are described in the following sections.

No DTMF Digits or Audio Passed on VoIP Calls to PSTN or PBX

Symptom

Caller makes a call, hears announcement messages (for example “enter your account number...”) but cannot pass DTMF digits. This symptom applies for both VoIP toll-bypass and Cisco IP phone calls to PSTN/PBX calls.

Problem Description

A Cisco IP phone (using Cisco CallManager) or POTS phone (VoIP) call leaves through a Cisco IOS gateway, where the called number is usually an interactive voice response (IVR). The IVR system sends back an ISDN progress message but does not connect until some account information is entered. By default, the audio path is cut through in the backward direction (toward the Cisco IP phone or originating gateway), but not in the forward direction, until the terminating gateway receives a connect message. Therefore, there is no voice path for the transmission of DTMF tones or speech towards the terminating switch.

Solution

Configure the Cisco IOS global configuration command **voice rtp send-recv** to establish (cut through) the audio path in both directions prior to receiving an ISDN connect message from the PSTN. For more information on this command refer to the *Cisco IOS Voice Command Reference*, Release 12.3.

No Busy Tone or Announcement Message Received When Placing VoIP Outbound Calls

Symptom

A Cisco IP phone (using CallManager) or POTS phone (VoIP) does not hear a busy tone or announcement message from the PSTN.

Solution

Configure the **voice call convert-discpi-to-prog** command. This command converts an inbound ISDN disconnect message with a progress indicator (PI) to an H.225 progress message with the same progress indicator (PI) value. This command can help when an announcement is played on the terminating PSTN side, but the calling party does not hear the response. In the VoIP toll-bypass scenario, most of these issues are resolved by upgrading the gateways to Cisco IOS Release 12.2(1) or later. However, if the originating device or originating ISDN switch does not keep the call active when an H.225/ISDN disconnect message is received, try using the **voice call convert-discpi-to-prog** command.

This might come up when the announcement in-band is a busy tone, as well. Beyond that, the busy signal should be provided by either the terminating device, the originating device, or the network. Some aspects of this can be controlled.

No Busy Tone on Inbound Call from Telephony (ISDN) to Cisco CallManager IP Phone, Cisco IOS Gateway, or Third-Party H.323 Device

Symptom

You cannot hear busy tone on a call from PSTN through a gateway to a Cisco CallManager IP phone, Cisco IOS gateway, or third-party H.323 device when running either an application or two-stage dialing on the originating gateway.

Solution

This can occur when the originating gateway is either running a voice application such as debit-card, or is running two-stage dialing. The latter refers to the calling party dialing the number to the gateway first, receiving the dial tone and then dialing the called party. In either case, the call has connected in terms of the PSTN network once it terminates on the originating gateway. If the IP call leg comes back with a release with the cause user-busy, the busy state cannot be indicated back towards the telephony session that is in connect state.

This problem has been addressed by having the originating gateway generate a busy tone when the release from the IP call leg is received with a cause code of user busy. The telephony leg is released either by the calling party or by the gateway after several minutes, with the cause code of normal call clearing.

For more information, refer to *Troubleshooting No Busy Tone and No Announcement Messages on ISDN-VoIP (H.323) Calls*, document 14066.

Troubleshooting H.323 Gateway Ringback

This section addresses call progress in-band related issues when you are interworking ISDN and H.323 signaling between VoIP and PSTN networks. Challenges arise when Cisco VoIP gateways exchange signaling capabilities with the telco switch. The following are common problem symptoms:

- No Ringback Tone on VoIP Toll-Bypass Calls—This occurs when a POTS (PSTN/PBX) user places a call (through Cisco gateways) and does not hear ringback tone before the call is answered.
- No Ringback Tone on VoIP Inbound Calls to Cisco CallManager (or Third-Party VoIP Devices) Through Cisco IOS Gateway—This occurs when a POTS (PSTN/PBX) user places a call to an IP phone (through a Cisco gateway) and does not hear ringback tone before the call is answered.
- No Ringback Tone on VoIP Outbound Calls from Cisco CallManager (or Third-Party Device) Through Cisco IOS Gateway—This occurs when a user places a call from an IP phone or third-party device to an outside number through a Cisco gateway and does not hear ringback tone.
- No Ringback to PSTN When IP Phones Initiate a Call Transfer (Cisco CallManager or Cisco Unity Voice Mail)—This occurs when an incoming call from a Cisco gateway that is transferred to Cisco CallManager or to Unity Voice Mail does not hear ringback.

No Ringback Tone on VoIP Toll-Bypass Calls

Symptom

A POTS user (from the PSTN or a PBX) places a call through a Cisco gateway and does not hear ringback tone before the call is answered.

Problem Description

The call terminating switch is sending the ringback tone, and signals a PI=8 to the terminating Cisco gateway. The PI information is then forwarded to the originating gateway via an H.225 progress message. The originating gateway is unable to decode the progress message and does not cut through the backward audio path to permit the transmission of the ringback tones. Some common scenarios for this problem are as follows:

- Terminating gateway running Cisco IOS software Release 12.1(5)T or later with originating gateway running Cisco IOS software Release 12.1T. The originating gateway does not understand the H.225 progress message and does not cut through the audio path until the connect message is received.

- Terminating Cisco gateway is connected to a CAS or analog interface and sends the PI information in an H.225 progress message to the originating gateway. The originating gateway is unable to decode the H.225 progress message.
- Third-party originating gateways and gatekeepers cannot properly parse H.225 progress messages.
- ISDN switch sends back inband ringback but Alert message does not contain a PI.

Solutions

Try the following solutions:

1. Configure the Cisco IOS global configuration command **voice call send-alert command** in the terminating gateway. This command enables the terminating gateway to send an Alert message instead of a Progress message after it receives a call setup. If this command is unavailable or does not work, go to step 2.

   For more information on this command refer to the *Cisco IOS Voice Command Reference*, Release 12.3.

2. Upgrade the Cisco IOS software onthe originating gateway to a current Cisco IOS software release. If this does not work, go to step 3.

3. Configure the terminating gateway to send a PI = 8 in the alert message. Configure the command **progress_ind alert enable 8** under the **voice dial-peer # pots** configuration. This command overrides the PI value received in ISDN alert message and causes the router to cut through the audio path back towards the calling party prior to connect.

   For more information on this command refer to the *Cisco IOS Voice Command Reference*, Release 12.3.

Note The **progress_ind alert** and the **progress_ind setup** commands are hidden in some versions of Cisco IOS and might not be visible within CLI help. However, if the **progress_ind progress** command is available in the help parser, the above commands are also available and can be entered into the dial peer in their entirety. These commands subsequently appear in the running configuration output.

No Ringback Tone on VoIP Inbound Calls to Cisco CallManager (or Third-Party VoIP Devices) Through Cisco IOS Gateway

Symptom

A POTS user (on the PSTN or PBX) places a call to an IP phone through a Cisco gateway and does not hear ringback tone before the call is answered.

Problem Description

This commonly occurs when the inbound call does not come in to the Cisco gateway/ router with a PI=3. ISDN switches send a PI=3 in the setup message to inform the gateway that the originating call is non-ISDN and that in-band messages are expected.

Solutions

Use one of the following solutions:

- Configure the **progress_ind setup enable 3** command when configuring the VoIP dial peer in the Cisco gateway. This command forces the gateway to treat the inbound ISDN setup message as if it came in with a PI = 3 generates an in-band ringback tone towards the calling party when the H.225 alert message does not contain a PI of 1, 2, or 8.

For more information on this command refer to the *Cisco IOS Voice Command Reference*, Release 12.3.

Note The **progress_ind alert** and the **progress_ind setup** commands are hidden in some versions of Cisco IOS and might not be visible within CLI help. However, if the **progress_ind progress** command is available in the help parser, the above commands are also available and can be entered into the dial peer in their entirety. These commands subsequently appear in the running configuration.

- An alternative to the **progress_ind setup** command is the **tone ringback alert-no-pi** command in **dial-peer voice** configuration mode. This command causes the gateway to generate ringback towards the calling party if an alert is received on the IP call leg with no PI present. It differs from the **progress_ind setup** command in that the outbound H.225 setup message does not contain a PI of 3 with the **tone ringback** command. Some devices might not accept setup messages when a PI is included.

No Ringback Tone on VoIP Outbound Calls from Cisco CallManager (or Third-Party Device) Through Cisco IOS Gateway

Symptom

A user makes an outbound call from a Cisco IP phone to the PSTN through a Cisco IOS gateway and does not hear ringback tone.

Problem Description

In this situation, the originating device expects in-band ringback tones, but either of the following might be happening:

- The PSTN or switch is not providing the ringback tone.
- The Cisco IOS gateway is not cutting through the audio to the originating device.

If the PSTN is providing in-band ringback, but the Q.931 alert message is not providing a PI indicating that there is in-band information, the gateway does not cut through the audio until the call is connected.

Solutions

Follow one of the solutions below:

- Ringback tones must come from the PSTN for trunk circuits in this situation. There are two dial-peer subcommands which may help. On the Cisco IOS gateway under the outgoing POTS dial peer, configure the command **progress_ind alert enable 8**. This command presents the Q.931 alert message to the software on the gateway as if the alert message had a PI of 8 and cut through the audio path.

Note The **progress_ind alert** and the **progress_ind setup** commands are hidden in some versions of Cisco IOS and might not be visible within CLI help. However, if the **progress_ind progress** command is available in the help parser, the above commands are also be available and can be entered into the dial peer in their entirety. These commands subsequently appear in the running configuration.

- In Cisco IOS software releases 12.2(2)T and later, configure the command **progress_ind setup enable 3** when configuring the POTS dial peer. This command causes the gateway to send a PI with a value of 3 in the ISDN setup message, indicating to the PSTN or PBX that the originating device is a non-ISDN device and in-band information should be presented. This command should be used with the command **progress_ind alert enable 8**.
- If the PSTN device, such as an ISDN phone directly connected to a BRI port on the gateway, is not able to generate ringback in-band, the gateway can be configured to generate ringback on the IP call leg. Configure the **tone ringback alert-no-pi** command when configuring the POTS dial peer. When the ISDN alert is received with no PI present, the gateway generates the ringback and includes PI=0x8 in the H.225 alert message.

No Ringback to PSTN When IP Phones Initiate a Call Transfer (Cisco CallManager or Cisco Unity Voice Mail)

Symptom

When a call to an IP phone is answered and then transferred, the caller does not hear ringback. When the transferred call is answered, both parties are able to hear each other.

Problem Description

From the perspective of the Cisco IOS gateway, the call is completed once the call is answered by an IP phone through Cisco CallManager or Unity Voice Mail system. Any further progress tones (in case of a call transfer) should be generated by the terminating device. However, Cisco CallManager and Unity are not capable of generating the in-band progress tones.

Solution for Cisco CallManager 3.0

To solve this problem you can either check the following conditions, or configure the Cisco IOS gateway as an MGCP gateway, instead of an H.323 gateway.

- You must have Cisco CallManager Release 3.0 (8) or higher.
- From the CallManager Administration page go to the Service menu and select **Service Parameters**. Perform the following steps for each active CallManager server:
  - In the Configured Services box, select **Cisco CallManager**.
  - In the Param drop-down list box, select **ToSendH225UserInfoMsg**.
  - Set the Value drop-down list box, to T for true.
  - Upgrade the gateway to Cisco IOS Release 12.2 (2) or higher.

This problem is documented in DDTS software bug CSCds11354.

Note These fixes are valid for ringback tones, but not for other progress tones, such as busy signal.

For more information, refer to *Troubleshooting No Ringback Tone on ISDN-VoIP (H.323) Calls*, document 22983.

Solution for Cisco CallManager 3.3 or Newer

To solve this problem for Cisco CallManager 3.3 or newer, perform the following steps:

Step 1 From the Cisco CallManager Administration page, go to the **Service** menu and select **Service Parameters**.

Step 2 Select the Cisco CallManager server from the drop down box.

Step 3 Select the system to modify from the drop down box. Make sure to select **Cisco CallManager**.

Step 4 From the Cisco CallManager Administration page go to the **Service** menu and select **Service Parameters**.

Step 5 From the drop-down box, select the server you wish to change.

Step 6 In the second drop-down box, select the service:**Cisco CallManager**.

Step 7 In the **Cluster Wide Parameters (Device - H323)** section, under the **Send H25User Info Message** selection, verify that the box for **No Ringback** is not checked and the box for **User Infro for Ring Back Tone** is checked.

Troubleshooting H.323 Gateway One-Way or No Audio

One-way audio and no audio are problems that are fairly common during a new VoIP network installation. Most of these problems are caused by misconfigurations. This section addresses some of the common issues that are known to result in IP telephony one-way audio conversations involving Cisco gateways.

This section provides scenarios and solutions to the following problems:

- On a phone call from an IP station through a Cisco IOS voice gateway, only one party receives audio (one-way communication).
- On a toll-bypass call between two Cisco gateways, only one party receives audio (one-way communication).

The causes for one-way audio or no audio in IP telephony can be varied, however the root of the problem is usually related to IP routing issues. For one-way audio, pay attention to which direction is experiencing the audio loss. Check the following topics if you are experiencing this issue:

- Ensuring IP Routing Is Enabled on Cisco IOS Gateways, page 24
- Checking Basic IP Routing, page 25
- Binding the H.323 Signaling to a Specific IP Address, page 25
- Checking That Answer Supervision Is Being Sent and Received Correctly From the Telco or Switch, page 26
- Using the voice rtp send-recv Command to Establish Early Two-Way Audio, page 26
- Checking cRTP Settings on a Link-by-Link Basis, page 26
- Verifying Minimum Software Level for NAT on Cisco IOS Gateways, page 27
- Disabling voice-fastpath on Cisco AS5350 and Cisco AS5400 Universal Gateways, page 27
- Configurinng the VPN IP Address with SoftPhone, page 27
- Verifying One-Way Audio, page 28

Ensuring IP Routing Is Enabled on Cisco IOS Gateways

Some Cisco IOS gateways, such as the VG200, have IP routing disabled by default. This leads to one-way voice problems. To enable IP routing, simply type the following global configuration command in your Cisco IOS gateway:

```
Router(config)#ip routing
```

Checking Basic IP Routing

Basic IP reachability should always be checked first. Because Real-Time Protocol (RTP) streams are connectionless (transported over UDP), traffic might travel successfully in one direction, but get lost in the opposite direction.

Once a call is established, an RTP stream carrying audio needs to flow in both directions between the endpoints. It could be that subnet A can be reached from subnet B, but subnet B cannot be reached from subnet A, so the audio stream from A to B always gets lost.

This is a basic routing issue, so use IP routing troubleshooting methods to get to the stage at which you can successfully ping phone A from gateway B. Bear in mind that ping is a bidirectional verification.

Check the following IP routing issues:

- Default gateways configured at the end stations
- IP routes on the default gateways, mentioned above, leading to the destination networks

Verify the default gateway configuration on various Cisco IP phones:

- Cisco IP Phone 7910—Press **Settings**, select option **6**, push volume down until the Default Router field shows up.
- Cisco IP Phone 7960/40—Press **Settings**, select option **3**, scroll down until the Default Router field shows up.
- Cisco IP Phone 2sp+/30vip—Press ****#**, then press **#** until gtwy= shows up.

Note If you are using the Cisco IP SoftPhone application and more than one NIC (network interface card) is installed in the box, make sure the box sources the correct NIC. This issue is often present in IP SoftPhone software Version 1.1.x.

Note If you are using Cisco DT24+ gateways, check the DHCP scope and make sure there is a default gateway (003 router) option in the scope. The 003 router parameter is what populates the Default Gateway field in the devices and PCs. Scope option 3 should have the IP address of the router interface that is doing routing for the gateway.

Binding the H.323 Signaling to a Specific IP Address

When the Cisco IOS gateway has multiple active IP interfaces, some of the H.323 signaling might be sourced from one IP address and other parts of it might reference different source addresses. This can generate various kinds of problems, one of them being one-way audio.

To get around this problem, the H.323 signaling can be bound to a specific source address, which can belong to a physical or virtual interface (loopback). The command syntax to use under interface configuration mode is **h323-gateway voip bind srcaddr** *ip_address*. Configure this command under the interface by using the IP address that Cisco CallManager points to.

Caution A bug exists in Cisco IOS Release 12.2(6) where this solution can actually cause a one-way audio problem. For more information, refer to bug ID CSCdw69681 in Cisco Software Bug Toolkit (registered customers only).

Checking That Answer Supervision Is Being Sent and Received Correctly From the Telco or Switch

In an implementation that has a Cisco IOS gateway connected to a telco or switch, verify that answer supervision is being sent correctly when the called device answers the call from behind the switch. Failure to receive answer supervision causes the Cisco IOS gateway to fail cut through (open) the audio path in a forward direction, causing a one-way voice. A workaround is to configure **voice rtp send-recv on**.

For more information, see the next section, "Using the voice rtp send-recv Command to Establish Early Two-Way Audio".

Using the voice rtp send-recv Command to Establish Early Two-Way Audio

The voice path is established in the backward direction as soon as the RTP stream is started. The forward audio path is not cut through until the Cisco IOS gateway receives a connect message from the remote end.

In some cases it is necessary to establish a two-way audio path as soon as the RTP channel is opened, before the connect message is received. To achieve this, use the **voice rtp send-recv** global configuration command.

Checking cRTP Settings on a Link-by-Link Basis

This issue applies to scenarios, such as toll-bypass, where more than one Cisco IOS gateway is involved in the voice path and compressed RTP (cRTP) is used. cRTP, or RTP header compression, is a method for making the VoIP packet headers smaller to regain bandwidth. cRTP takes the 40-byte IP/UDP/RTP header on a VoIP packet and compresses it to 2–4 bytes per packet, yielding approximately 12 KB of bandwidth for a G.729 encoded call with cRTP. For more information on cRTP, refer to *Voice Over IP - Per Call Bandwidth Consumption*, document ID 7934.

cRTP is done on a hop-by-hop basis with decompression and recompression on every hop. Each packet header needs to be examined for routing, therefore cRTP needs to be enabled on both sides of an IP link.

It is also important to verify that cRTP is working as expected on both ends of the link. Cisco IOS levels vary in terms of switching paths and concurrent cRTP support.

In summary, the history of cRTP support on Cisco IOS is:

- Until Cisco IOS software Release 12.0.5T, cRTP is process-switched.
- In Cisco IOS software Release 12.0(7)T through Cisco IOS Release 12.1(1)T, fast- and Cisco-express forwarding (CEF)-switching support for cRTP are introduced.
- In Cisco IOS software Release 12.1(2)T, algorithmic performance improvements are introduced.

If you are running cRTP using Cisco IOS Release 12.1, verify that bug CSCds08210 does not affect your Cisco IOS version (VoIP and fax not working with RTP header compression on).

Verifying Minimum Software Level for NAT on Cisco IOS Gateways

If you are using Network Address Translation (NAT), the minimum software level requirements must be met. Earlier versions of NAT do not support skinny protocol translation and leads to one-way voice issues.

The minimum software levels required for using NAT and skinny simultaneously are Cisco IOS® Software 12.1(5)T for IOS gateways to support skinny and H.323v2 with NAT.

Note If your Cisco CallManager is using a TCP port for skinny signaling different from the default port (2000), you need to adjust the NAT router with the **ip nat service skinny tcp port** *number* global configuration command.

The minimum software level required for using NAT and skinny simultaneously on a PIX firewall is Release 6.0.

Note These levels of software do not necessarily support all of the RAS messages necessary for full gatekeeper support. Gatekeeper support is outside the scope of this document.

Disabling voice-fastpath on Cisco AS5350 and Cisco AS5400 Universal Gateways

The Cisco IOS command **voice-fastpath enable** is a hidden global configuration command for the Cisco AS5350 and Cisco AS5400. This command is enabled by default. To disable it, use the **no voice-fastpath enable** global configuration command.

When enabled, this command caches the IP address and UDP port number information for the logical channel opened for a specific call and prevents the RTP stream from getting to the application layer, but rather forwards the packets at a lower layer, which helps reduce CPU utilization marginally in high call volume scenarios.

When supplementary services, such as hold or transfer are used, the **voice-fastpath** command causes the router to stream the audio to the cached IP address and UDP port, disregarding the new logical channel information generated after a call on hold is resumed or a transfer is completed. To get around this problem, traffic should go to the application layer constantly so that redefinition of the logical channel is taken into account and audio is streamed to the new IP address/UDP port pair. That is why you should disable **voice-fastpath** to support supplementary services.

Configurinng the VPN IP Address with SoftPhone

Cisco IP SoftPhone enables you to make a PC work like a Cisco IP Phone 7900 Series phone. Remote users who connect back to their company network through VPN need to configure some additional settings in order to avoid a one-way voice problem. This is a result of the media stream needing to have the endpoint of the connection specified.

The solution is to configure the VPN IP address, instead of the IP address of the network adapter, under the Network Audio Settings.

Verifying One-Way Audio

A useful command to verify packet flow is **debug cch323 rtp**. This command displays packets transmitted (X) and received (R) by the router. An uppercase character indicates successful transmission/reception, and a lowercase character indicates a dropped packet.

```
Router# debug cch323 rtp
RTP packet tracing is enabled
Router#

!--- This is an unanswered outgoing call.
!--- Notice that voice path only cuts through in forward
!--- direction and that packets are dropped. Indeed,
!--- received packets are traffic from the IP phone to the PSTN
!--- phone. These will be dropped until the call is answered.

Mar 3 23:46:23.690: ****** cut through in FORWARD direction *****
XXXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XrXrXXrrrrrrrrrrrrrrr
Router#
Router#

!--- This is an example of an answered call:

Router#
Router#
*Mar 3 23:53:26.570: ****** cut through in FORWARD direction *****
XXXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XXrrrrrXrXrXrXrXrXrXrXrXrXrXrrXXrrXrXrXrXrXrXXXXXXXXXXXXXXXXrXXXXXXXXrXrXrXXrrXr
XrXrXrXrXrXrXrXrXXrrrrrrrrrrrrrrrrrrrrrrrrrrrr

!-- At this point the remote end picks up the phone.

*Mar 3 23:53:30.378: ****** cut through in BOTH direction *****
XRXRXRXRXRXRXRXXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXR
XXRRRXRXRXXRRXRXRXRXRXXRXRXRXRXRXRRXRXXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXXRRRRRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXXRRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XXRRRRRRRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXXRRRXR

!-- End of conversation.
```

Using the Test Call Feature to Verify Voice Path

The Test Call feature provides the ability for a remote station or gateway to establish a call to any destination address from a Test Call station located at a network operations center and to audibly verify the voice path:

- The system administrator can manually place a specific CIC on the target TG in test mode.
- The TCL script in OGW can specify a destination CIC or DS0 number.

- This circuit number can then be carried to the terminating gateway.
- The TGW can override any local circuit selection configured and use the specified circuit number for the outgoing call.

Information About the Test Call Feature

This section provides summary information about the test call feature. This section also describes some of the restrictions and limitations that should be taken into account when you are enabling and activating the test call feature.

Route Server

- The administrator can configure parameters, termination trunk group, and destination gateway IP address in the route server.
- The route server detects this test call by the “Test Call” source trunk group label, bypasses the normal routing logic, and routes the call directly to a specified trunk group based on the configured destination (gateway IP address plus trunk group label). All trunk group members (GW IP address plus trunk group label) in a trunk group will be displayed in the route server routing hierarchy and the test call person can select any trunk requisite.
- The group member can be specified as the destination so that the test call can be routed to a specific destination trunk group label on a specific gateway.
- A test call syslog is generated in the route server. This syslog covers the route information (incoming carrier ID/trunk group ID/trunk group number, outgoing carrier ID/trunk group ID/trunk group number), the IP address of the destination gateway, and the called number, and can be browsed from its GUI web.
- Because the ARQ message sent from the test station OGW will not have any GTD payload, the route server builds a GTD if the Termination trunk group is ISDN/ISUP/R2. The route server also sends the same format of the currently used route descriptor to the test station OGW.

TCL Script

- TCL scripts in the OGW send the route descriptor to the mediation server for accounting purposes.
- TCL scripts in the OGW override the GTD CPC value received from the route server with the configured value.

  The configured value may be changed using the **application**, **service**, and **paramspace** command structure shown in the “Sample Tasks” section on page 32.
- TCL scripts in the OGW insert the termination CIC port number in the outgoing H225 Setup message.
- The TCL script in the OGW handles two-stage calls. If the call is originated from the test call trunk group, the TCL script in the OGW parses the dialed digits to identify the delimiter and split the dialed digits to the E164 number and CIC number.

Limitations

- Test-mode status maintained within the gateway is reset if any interface is added or removed from the trunk group.
- If the DS0 is placed in test mode, it is not selected for nontest calls. The system administrator must reset the DS0 from the test mode.

- The test mode configuration is not saved for reloads. Once the gateway reloads, the test mode configuration is lost. The administrator has to place the CIC in test mode again to place a test call.

Test Mode Limitations

- In some SS7 networks, the signaling on the terminating switch can override the local channel selection. In such cases, there is no guarantee that the test call will be placed on that specified channel.
- The number of the DS0 or CIC in test-mode status does not affect the call capacity update to the H323 client.
- When a DS0 or channel is placed in test-mode, the trunk group circuit selection algorithm avoids using the channel for outgoing nontest calls. But if call routing on the gateway is not based on the trunk group or carrier-ID, then this channel may be used. Similarly, the DS0 or channel is not guaranteed to be reserved if the gateway supports hybrid routing (some calls are based on the trunk group or carrier-ID routing and some calls not)
- Placing a DS0 or channel in test mode would only guarantee that no new outgoing calls will be placed on the DS0 in the trunk. However, no control is exercised on existing calls or incoming calls in that channel. To clear the existing call through the channel (if desired), enter this command:

  clear call voice causecode *cause id callID*
- All incoming calls through the channel in test mode will not be rejected. It is up to the terminating switch to correspondingly place the channel in test mode and avoid sending calls through this channel.

Feature Limitations

- The Test Call feature allows the Tcl script in the OGW to send the CIC number for Test Call to the TGW. If the corresponding DS0 is not placed in test mode, the trunk selection API would make a best-effort attempt to select the corresponding CIC or DS0 number if it is free and would continue with the test call setup.
- The CIC number is carried in H225 Setup message H323 V4 field. Thus this feature would work only in H323v4 networks,
- The test call feature is not supported in SIP and H323 v2 networks.

Test Call Command-Line Interface

To enable the test call feature, use a new CLI to place specific CIC or DS0s of a trunk group in test mode. The normal circuit selection algorithm would not select this DS0 or CIC number for nontest calls.

```
gateway# test trunkgroup tg-label [set-tgCic | reset-tgCic] cic-number [cpc test cpc value]
```

In the syntax, the **set-tgCic** keyword is used to set the test mode, and the **reset-tgCic** is used to reset from test mode.

To display the mapping of trunk groups to sequential CIC numbers, use the modified **show trunk group** command. A new keyword **cic** has been added:

```
gateway# show trunk group tg-label [cic]
```

In the syntax, the new **cic** keyword displays the mapping of trunk group DS0 numbers to sequential CIC numbers. A sample output follows:

```
gateway# show trunk group 1 cic
```

```
trunk group: 1
        Description:
        trunk group label: 1


Se3/0:23
Timeslot     1     2     3     4     5     6     7     8     9     10    11    12
cic          1     2     3     4     5     6     7     8     9     10    11    12
Timeslot     13    14    15    16    17    18    19    20    21    22    23    24
cic          13    14    15    16    17    18    19    20    21    22    23    24

Se5/1:23
Timeslot     1     2     3     4     5     6     7     8     9     10    11    12
cic          25    26    27    28    29    30    31    32    33    34    35    36
Timeslot     13    14    15    16    17    18    19    20    21    22    23    24
cic          37    38    39    40    41    42    43    44    45    46    47    48
```

```
gateway# show trunk group

trunk group: 1
        Description:
        trunk group label: 1

        Translation profile (Incoming):
        Translation profile (Outgoing):

        Hunt Scheme is least-used
        Max Calls (Incoming):   NOT-SET (Any)   50 (Voice)      NOT-SET (Data)
        Max Calls (Outgoing):   NOT-SET (Any)   50 (Voice)      NOT-SET (Data)
        Retries: 0

        Trunk Se3/0:23  Preference DEFAULT
                Member Timeslots : 1-24
                Total channels available : 12
                Data = 0, Voice = 0, Modem = 0, Pending = 0, Free = 12
        Trunk Se5/1:23  Preference DEFAULT
                Member Timeslots : 1-24
                Total channels available : 23
                Data = 0, Voice = 0, Modem = 0, Pending = 0, Free = 23
                Test mode = 1                    <---- new
                Testmode Timeslot(s) : 19        <---- new

        Total calls for trunk group: Data = 0, Voice = 0, Modem = 0
                                     Pend = 0, Free = 35

        advertise_flag 0x00000040, capacity timer 25 sec tripl_config_mask 0x00000000
        AC_curr 35, FD_curr 0, SD_curr 0

        succ_curr 7 tot_curr 7
        succ_report 7 tot_report 7
        changed 1 replacement position 0
```

Sample Tasks

The following is a sample set of tasks, showing the existing and new or modified CLI that enables the test call feature. This sample is provided for reference purposes only.

Originating Analog Gateway Configuration

Step 1 Trunk group configuration:

```
!
trunk group  41001
 max-retry 1
 hunt-scheme sequential both up
!
```

Step 2 Add trunk group to analog voice port (FXS):

```
voice-port 1/0/0
 trunk-group 41001
!
```

Step 3 Application with kyuss script configuration:

```
application
 service mkyuss flash:app_kyuss.3_0_1.tcl
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://<tftp-server-ip>/prompts/en
  paramspace english prefix en
  param test_call_CPC 10
  param test_call_src_tg 41001
!
```

Step 4 Dial-peer configuration:

```
dial-peer voice 1025 pots
 trunkgroup 41001
 service mkyuss
 destination-pattern 2015550100
 trunk-group-label source 41001
dial-peer voice 1000 voip
 destination-pattern 2015550...
 session target ras
 incoming called-number 2015551...
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 no vad
```

======= Terminating side ==========

Step 1 Gateway configuration:

```
trunk group  41001
 max-retry 1
 hunt-scheme sequential both up
!
!

!
controller E1 3/0
 ds0-group 0 timeslots 1-15,17-31 type e&m-immediate-start
 cas-custom 0
 trunk-group 41001
!
dial-peer voice 1025 pots
 trunkgroup 41001
 service mkyuss
 destination-pattern 2015550100
!
dial-peer voice 1000 voip
 destination-pattern 2015550...
 session target ras
 incoming called-number 2015551...
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 no vad
```

Step 2 Put terminating CIC in test mode:

```
test trunkgroup 41001 reset_tgCic 21 cpc 10
```

Step 3 Dial destination phone number:

```
Dialed string from orig gw analog phone : <dest_phone_num>*<cic>#
```

Troubleshooting SIP Interfaces to the IP Network

The Cisco Session Initiation Protocol (SIP) implementation enables Cisco access platforms to signal the setup of voice and multimedia calls over IP networks.

SIP is an ASCII-based, application-layer control protocol that can be used to establish, maintain, and terminate calls between two or more endpoints. SIP is an alternative protocol developed by the Internet Engineering Task Force (IETF) for multimedia conferencing over IP. SIP features are compliant with IETF RFC 2543, SIP: Session Initiation Protocol, published in March 1999. You can view RFC 2543 at http://www.ietf.org/rfc/rfc2543.txt.

Like other Voice-over-IP protocols, SIP is designed to address the functions of signaling and session management within a packet telephony network. Signaling allows call information to be carried across network boundaries. Session management provides the ability to control the attributes of an end-to-end call.

This chapter provides procedural and reference information that you can use to determine and resolve problems with SIP interfaces to the IP network.

Note Call flows can help in troubleshooting SIP problems. SIP call flow information can be found in the *Session Initiation Protocol Gateway Call Flows* document.

This chapter contains the following information:

Troubleshooting the Cisco SIP IP Phone 7960

This section describes troubleshooting features and tips for the Cisco SIP IP phone 7960.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Troubleshooting Features

The following is a list of features on the Cisco SIP IP phone that you can use for troubleshooting:

- Settings button to Network Configuration soft key—Use to view or modify the network configuration of the phone.
- Settings button to SIP Configuration soft key—Use to view or modify a phone's SIP settings.
- Settings button to Status—Display configuration or initialization errors.
- Call messages on LED screen—Display basic SIP message flows.
- Pressing ***i*** or ***?*** key twice during a call—Displays real-time transferring and receiving call statistics. This option is recommended for troubleshooting voice-quality issues.

In addition to the features listed above, the EIA/TIA-232 (RS-232) port located on the back of the Cisco SIP IP phone 7960 is a console port and can be used to gather debug information.

The EIA/TIA-232 port is password-protected and requires a custom RJ-11-to-RJ-45 cable.

Note For a PC connection, the RJ-45 connection needs a DB-9 female DTE adapter or an RJ-45 crossover cable for an octal async connection. You must enter the password "cisco" must be entered to enable any output to be seen via the EIA/TIA-232 port. The connection baud rate, parity, start bits, and stop bits are 9600, N, 8, and 1.

To use the console port, use a RJ-11-to-RJ-45 custom cable to connect the EIA/TIA-232 port to a PC.

Table 48 lists the RJ-11-to-RJ-45 cable pinouts.

Table 48 RJ-11-to-RJ-45 Pinouts

RJ-11 or RJ-12	RJ-45
2	6
3	4
4	3

To connect the console port, complete the following tasks:

Step 1 Insert the RJ-11 end of the rolled cable into the EIA/TIA-232 port on the back of the phone.

Step 2 Use an RJ-45-to-DB-9 female DTE adapter (labeled TERMINAL) to connect the console port to a PC running terminal emulation software.

Step 3 Insert the RJ-45 end of the rollover cable into the DTE adapter.

Step 4 From the console terminal, start the terminal emulation program.

Step 5 Type "cisco". A prompt is displayed.

Step 6 At the prompt, you can issue the following commands to assist you in troubleshooting and debugging the phone:

- **debug error**—Displays error messages that are occurring in the call flow process
- **debug sip-message**—Enables you to view a text display of a call flow

Troubleshooting Tips

This section provides tips for resolving the following Cisco SIP IP phone problems:

For more information about Cisco SIP IP phones, see the *Cisco IP Phone Administrator Guides for SIP*.

Cisco SIP IP Phone Is Unprovisioned or Is Unable to Obtain an IP Address

To determine why a phone is unprovisioned or unable to obtain an IP address, perform the following tasks as necessary:

- If using TFTP to download configuration files, verify that the SIPDefault.cnf file and the phone-specific configuration file (SIP*mac*.cnf where *mac* is the MAC address of the phone) exist and are configured correctly.
- Verify that the TFTP server is working properly.
- Verify that the Cisco SIP IP phone network configuration parameters are properly configured and the phone is obtaining the proper IP addressing information (IP address, subnet mask, default gateway, TFTP server, and so forth.)
- Press the **Settings** button, select **Status**, and then **Status Messages** to view messages for missing files or other errors.
- If the DHCP server has an IP subnet mask that is different from the one for the Cisco SIP IP phone, verify that “ip helper-address” address is enabled on the local router.
- Verify that the Cisco SIP IP phone software image (P0S3*xxyy*.bin where *xx* is the version number and *yy* is the subversion number) was downloaded from the Cisco website in binary format.

Cisco SIP IP Phone Does Not Register With the SIP Proxy or SIP Registrar Server

To determine why a phone does not register with a SIP proxy or SIP registrar server, perform the following tasks as necessary:

Note The character “x” displayed to the right of a line icon indicates that registration has failed.

- Verify that phone registration with a proxy server is enabled (via the proxy_register parameter in the configuration files). By default, registration during initialization is disabled.
- Verify that the IP address (proxy1_address parameter) of the primary SIP proxy server to be used by the phones is valid.

- If a Fully Qualified Domain Name (FQDN) is specified in the proxy1_address parameter, verify that the DNS server is configured to resolve the FQDN as a DNS A-record type.
- Verify that the Cisco SIP proxy server has been configured to require authentication. If it has, ensure that an authentication name and password have been defined in the Cisco SIP IP phone-specific configuration file (through the use of the linex_authname and linex_password parameters).
- The Cisco SIP IP phone currently supports the HTTP Digest authentication method. Verify that the authentication method required by the Cisco SIP proxy server (through the use of the AuthScheme directive in the sipd.conf file) is HTTP Digest.
- Verify that a registration request hasn't expired. By default, Cisco SIP IP phones reregister every 3600 seconds, but this value can be modified through the use of the time_reqister_expires parameter.

Outbound Calls Cannot Be Placed from a Cisco SIP IP Phone

If a call cannot be placed from a Cisco SIP IP phone, perform the following tasks as necessary:

- Verify that the Cisco SIP IP phone network configuration IP address parameters have been correctly entered or received from a DHCP server.
- Verify that the Cisco SIP proxy server used by the phone is working properly.
- Verify that the Cisco SIP proxy server is correctly configured for routes or registrations to the remote destination.
- Verify that the remote SIP device is available.
- Verify that a dial plan has been defined in the dialplan.xml file and if so, that the configuration is correct. This file should have been downloaded from CCO to the root directory of your TFTP server.
- Determine which error tones are being received and map those tones to the messages displayed on the phone's LCD (SIP 4xx messages, and so forth.)

Inbound Calls Cannot Be Received on a Cisco SIP IP Phone

If inbound calls cannot be received on a Cisco SIP IP phone, perform the following tasks as necessary:

- Verify that the line (user portion) was defined in the Request-URI or the SIP INVITE request. The Cisco SIP IP phone requires this information to determine the proper line to ring.
- Verify that the Request-URI is sent to port 5060 of the phone's IP address. The phone listens on UDP port 5060.
- Verify that the Cisco SIP IP phone is registered with the local proxy server.

Poor Voice Quality on the Cisco SIP IP Phone

If a call's voice quality is compromised on the Cisco SIP IP phone, perform the following tasks as necessary:

- Check the network path for errors, packet drops, loss, loops, and so forth.
- Verify that the ToS level for the media stream being used has been correctly set (through the tos_media parameter in the configuration file).
- Verify that the Cisco SIP IP phone is plugged into a switch rather than a hub to avoid excessive collisions and packet loss.
- Ensure that there is enough bandwidth on the network for the selected codec (especially for calls over a WAN).

- Press the **i** or **?** button twice on the phone during the call to view realtime transferring and receiving call statistics.
- Determine whether the problem occurs with the handset, headset, or speaker phone, or with all of them.

DTMF Digits Do Not Function Properly

If DTMF digits are not functioning properly, perform the following tasks as necessary:

- If out-of-bound signaling through the AVT tone method has been enabled (through the dtmf_outofband configuration file parameter), verify that the remote device supports AVT tones (as defined in RFC 2833). If AVT tones have been enabled and the remote device does not support AVT tones, check for packet loss in the end-to-end path.
- Find out which codec is being used. Lower bandwidth codecs yield poorer results if AVT tones are not supported because the DTMF digits are carried in audio.
- Verify the length of the tones being created. The tone must have a minimum signal duration of 40 ms with signaling velocity (tone and pause) of no less than 93 ms (as defined in RFC 2833).

Cisco SIP IP Phones Do Not Work When Plugged into a Line-Powered Switch

If the Cisco SIP IP phones do not work when plugged into a line-powered switch, perform the following tasks:

- Verify that the phone is running version 2.0 or higher of the Cisco SIP IP Phone software. (Line-powered support was not available in version 1.0.)
- Verify that the network media type Network Settings parameter is set to auto-negotiation (auto).

Call Transfer Does Not Work Correctly

If call transfer does not work correctly, verify that the remote SIP device that is sending the call is using the SIP BYE/Also: method (as defined in Internet draft sip-cc-01.txt.)

Some SIP Messages are Retransmitted Too Often

The Cisco SIP IP phone has several timers (INVITE request retries, BYE request retries, etc.) that can be configured using the sip_invite_retx and sip_retx configuration file parameters. In most networks, the default values work fine, however, conditions such as network delay, slower-processing proxy servers, and packet loss might require that the timers be adjusted. If some SIP messages appear to be retransmitted too often, adjust these parameters.

Troubleshooting the Cisco SIP Gateway

This section provides tips for resolving the following Cisco SIP gateway problems:

Unable to Make Outbound Calls from the Cisco SIP Gateway to a SIP Endpoint

If a call cannot be placed from the Cisco SIP gateway, perform the following tasks as necessary:

- Verify that the voice ports are properly configured and enabled for the PSTN-side signaling protocol.
- Verify that there is a valid VoIP dial peer configured that meets the following requirements:
  - Matches the required destination pattern
  - Is SIP-enabled (through the **session protocol sipv2** command)
  - Has the correct dial peer session target defined (through the **session target *sip-server*** command
  - Has the codec correctly defined
- Using the **ping** command, verify that the SIP gateway can communicate through IP with the SIP proxy or remote SIP device.
- If the SIP proxy server is defined through the use of a FQDN, verify that the DNS server is correctly configured to resolve that address using a DNS SRV record.
- Ensure that the time zone format configured on the SIP gateway is GMT.
- Check the **debug ccsip all | calls | error | events | messages | states** command output for protocol errors.

Unable to Make Inbound Calls to a PSTN Through a Cisco SIP Gateway

If inbound calls to a PSTN cannot be made through the Cisco SIP gateway, perform the following tasks as necessary to determine the cause:

- Verify that the voice ports are correctly configured and enabled for the PSTN-side signaling protocol.
- Verify that a valid POTS dial peer is configured and that it matches the required destination pattern.
- Using the **ping** command, verify that the Cisco SIP gateway can communicate with the SIP proxy server or remote SIP device through IP.
- If the inbound call has any hostnames defined as a FQDN, ensure that the proper DNS configuration is enabled on the Cisco SIP gateway (to resolve the hosts).
- View the **debug ccsip all | calls | error | events | messages | states** command output for protocol errors.

Calls to a PSTN via the Cisco SIP Gateway Fail with a "400 Bad Request" Response

If the Cisco SIP gateway does not like part of a SIP message (header or SDP), the call attempt fails with a "400 Bad Request" response.

To determine whether the call failed because of a SIP header error, issue the **debug ccsip** command that displays information on the error message, or verify that the required SIP header elements exist as defined in RFC 2543. SIP header fields are shown in Table 49.

Table 49 SIP Header Fields

Header Field	Definition
Call-ID	The Call-ID general-header field uniquely identifies a specific invitation or all registrations of a specific client. Note that a single multimedia conference can give rise to several calls with different Call-IDs. For example, if a user invites a single individual several times to the same (long-running) conference.
Contact	The Contact general-header field MUST appear in INVITE and REGISTER requests and in 200 responses. It can appear in ACK, and in other 1xx, 2xx, 3xx, 485 responses. In general, it provides a URL where the user can be reached for further communications.
Content-Length	The Content-Length entity-header field indicates the size of the message-body, in decimal number of octets, sent to the recipient.
Content-Type	The Content-Type entity-header field indicates the media type of the message-body sent to the recipient.
Cseq	Users MUST add the CSeq (command sequence) general-header field to every request. A CSeq header field in a request contains the request method and a single decimal sequence number chosen by the requesting client, unique within a single value of Call-ID. The sequence number MUST be expressed as a 32-bit unsigned integer. The initial value of the sequence number is arbitrary, but MUST be less than 2**31. Consecutive requests that differ in request method, headers, or body, but have the same Call-ID MUST contain strictly monotonically increasing and contiguous sequence numbers; sequence numbers do not wrap around. Retransmissions of the same request carry the same sequence number, but an INVITE with a different message body or different header fields (a "re-invitation") acquires a new, higher sequence number. A server MUST echo the CSeq value from the request in its response. If the Method value is missing in the received CSeq header field, the server fills it in appropriately.
Date	Date is a general-header field. Its syntax is: SIP-date = rfc1123-date Note that unlike HTTP/1.1, SIP only supports the most recent RFC 1123 [29] formatting for dates.
Diversion	**Note** Currently gateway uses Diversion header in initial outgoing messages.

Table 49 SIP Header Fields (continued)

Header Field	Definition
Expires	The Expires entity-header field gives the date and time after which the message content expires. This header field is currently defined only for the REGISTER and INVITE methods. For REGISTER, it is a request and response-header field. In a REGISTER request, the client indicates how long it wants the registration to be valid. In the response, the server indicates the earliest expiration time of all registrations. The server MAYchoose a shorter time interval than that requested by the client, but SHOULD NOT choose a longer one.
From	Requests and responses MUST contain a From general-header field, indicating the initiator of the request. The From field MUST contain a tag. The server copies the From header field from the request to the response. The optional “display-name” is meant to be rendered by a human-user interface. A system SHOULD use the display name “Anonymous” if the identity of the client is to remain hidden. The SIP-URL MUST NOT contain the “transport-param”, “maddr-param”, “ttl-param”, or “headers”elements. A server that receives a SIP-URL with these elements removes them before further processing.
Max-Forwards	The Max-Forwards request-header field may be used with any SIP method to limit the number of proxies or gateways that can forward the request to the next downstream server. This can also be useful when the client is attempting to trace a request chain which appears to be failing or looping in mid chain. The Max-Forwards value is a decimal integer indicating the remaining number of times this request message is allowed to be forwarded. Each proxy or gateway recipient of a request containing a Max-Forwards header field MUST check and update its value before forwarding the request. If the received value is zero (0), the recipient MUST NOT forward the request. Instead, for the OPTIONS and REGISTER methods, it MUST respond as the final recipient. For all other methods, the server returns 483 (too many hops). If the received Max-Forwards value is greater than zero, then the forwarded message MUST contain an updated Max-Forwards field with a value decremented by one (1).
Require	The Require request-header field is used by clients to tell useragent servers about options that the client expects the server to support in order to properly process the request. If a server does not understand the option, it MUST respond by returning status code 420 (bad extension) and list those options it does not understand in the Unsupported header.
Server	The Server response-header field contains information about the software used by the user agent server to handle the request.
Timestamp	The timestamp general-header field describes when the client sent the request to the server. The value of the timestamp is of significance only to the client and it MAY use any time scale. The server MUST echo the exact same value and MAY, if it has accurate information about this, add a floating point number indicating the number of seconds that have elapsed since receiving the request. The timestamp is used by the client to compute the round-trip time to the server so that it can adjust the time out value for retransmissions.

Table 49 SIP Header Fields (continued)

Header Field	Definition
To	The To general-header field specifies recipient of the request, with the same SIP URL syntax as the From field. Requests and responses MUST contain a To general-header field, indicating the desired recipient of the request. The optional “display-name”is meant to be rendered by a human-user interface. The UAS or redirect server service processing a request MUST always add a tag to To-header.
User-Agent	The User-Agent general-header field contains information about the client user agent originating the request.
Via	The Via field indicates the path taken by the request so far. This prevents request looping and ensures replies take the same path as the requests, which assists in firewall traversal and other unusual routing situations. When the UAC creates a request, it MUST insert a Via into that request.

Possible SDP-related errors are as follows:

- SDP_ERR_INFO_UNAVAIL
- SDP_ERR_VERSINFO_INVALID
- SDP_ERR_CONNINFO_IN
- SDP_ERR_CONNINFO_IP
- SDP_ERR_CONNINFO_NULL
- SDP_ERR_CONNINFO_INVALID
- SDP_ERR_MEDIAINFO_TYPE
- SDP_ERR_MEDIAINFO_INVALID
- SDP_ERR_MEDIAINFO_NULL
- SDP_ERR_OWNERINFO_NULL
- SDP_ERR_OWNERINFO_SESSID_NULL
- SDP_ERR_OWNERINFO_SESSID_INVALID
- SDP_ERR_OWNERINFO_VERSID_NULL
- SDP_ERR_OWNERINFO_VERSID_INVALID
- SDP_ERR_OWNERINFO_IN
- SDP_ERR_OWNERINFO_IP
- SDP_ERR_TIMEINFO_ST_NULL
- SDP_ERR_TIMEINFO_ET_NULL
- SDP_ERR_TIMEINFO_ST_INVALID
- SDP_ERR_TIMEINFO_ET_INVALID
- SDP_ERR_ATTRINFO_INVALID
- SDP_ERR_ATTRINFO_NULL
- SDP_ERR_AUDIO_MEDIA_UNAVAIL
- SDP_ERR_MEDIAINFO_PORT_INVALID

- SDP_ERR_MEDIAINFO_MALLOC_FAIL
- SDP_ERR_ATTRINFO_MALLOC_FAIL

Possible CheckRequest errors are as follows:

- CHK_REQ_FAIL_MISMATCH_CSEQ
- CHK_REQ_FAIL_INVALID_CSEQ
- CHK_REQ_FAIL_FROM_TO
- CHK_REQ_FAIL_VERSION
- CHK_REQ_FAIL_METHOD_UNKNOWN
- CHK_REQ_FAIL_REQUIRE_UNSUPPORTED
- CHK_REQ_FAIL_CONTACT_MISSING
- CHK_REQ_FAIL_MISMATCH_CALLID
- CHK_REQ_FAIL_MALFORMED_CONTACT
- CHK_REQ_FAIL_MALFORMED_RECORD_ROUTE

Voice Quality Is Compromised on Calls Through or From the Cisco SIP Gateway

If the voice quality on calls through or from the Cisco SIP gateway is compromised, perform the following tasks as necessary to determine the cause:

- Check the network path for errors, packet drops, loss, loops, and so forth.
- Verify that the TOS bits have been correctly set in the VoIP dial peer through the use of the **ip precedence** command.
- To minimize excessive collisions and packet loss, connect the Cisco SIP gateway to a switch rather than a hub.
- Verify that enough bandwidth exists on the network for the configured codec (especially for calls over a WAN).
- View the output of the show interface command for packet drops. View the output of the s**how voice dsp** command for DSP-related issues.
- Determine whether errors exist on the voice ports that could be causing the problems.

Some SIP Messages Are Retransmitted Too Often

The Cisco SIP gateway has SIP timers (INVITE request retries, BYE request retries) configured under the SIP UA through the use of the **timers trying number**, **timers expires time**, and **retry invite number** commands. In most networks, the default values work well, but conditions such as network delay, slower-processing proxy servers, and packet loss might require that the timers be adjusted. If some SIP messages appear to be retransmitted too often, adjust these parameters.

Call Transfer Does Not Work Correctly

If call transfer does not work correctly, perform the following tasks to determine the cause:

- Verify that the application session is defined on the VoIP and POTS dial peers.

- Verify that the remote SIP device that is sending the call through the use of the SIP BYE/Also: method (as defined in Internet draft sip-cc-01.txt).
- Use the **debug voip ccapi inout** command to verify that a dial peer that has application session defined is matched. The application used after the BYE request is sent should be “session” instead of “SESSION.”

Troubleshooting Commands

There are several debug commands that are useful for troubleshooting problems with SIP, as follows:

- **debug ccsip all**
- **debug ccsip calls**
- **debug ccsip error**
- **debug ccsip events**
- **debug ccsip info**
- **debug ccsip media**
- **debug ccsip messages**
- **debug ccsip preauth**
- **debug ccsip states**

Details about these commands can be found in the *Cisco IOS Debug Command Reference*.

Note The output from these commands can be filtered. For more information, see the “SIP Debug Output Filtering Support” section on page 83.

The following show and debug commands shown can be used to troubleshoot the Cisco SIP gateway:

- **show sip status**—Displays the SIP user agent listener status.

```
sip-2600a# show sip status

SIP User Agent Status
SIP User Agent for UDP : ENABLED
SIP User Agent for TCP : ENABLED
SIP max-forwards : 6
```

- **show sip statistics**—Displays SIP user agent statistics.

```
router# show sip statistics

SIP Response Statistics (Inbound/Outbound)
Informational:
Trying 3/0, Ringing 3/0,
Forwarded 0/0, Queued 0/0,
SessionProgress 0/0
Success:
OkInvite 3/0, OkBye 2/0,
OkCancel 0/0, OkOptions 0/0
Redirection (Inbound only):
MultipleChoice 0, MovedPermanently 0,
MovedTemporarily 0, SeeOther 0,
UseProxy 0, AlternateService 0
Client Error:
BadRequest 0/3, Unauthorized 0/0,
```

```
PaymentRequired 0/0, Forbidden 0/0,
NotFound 0/0, MethodNotAllowed 0/0,
NotAcceptable 0/0, ProxyAuthReqd 0/0,
ReqTimeout 0/0, Conflict 0/0, Gone 0/0,
LengthRequired 0/0, ReqEntityTooLarge 0/0,
ReqURITooLarge 0/0, UnsupportedMediaType 0/0,
BadExtension 0/0, TempNotAvailable 0/0,
CallLegNonExistent 0/0, LoopDetected 0/0,
TooManyHops 0/0, AddrIncomplete 0/0,
Ambiguous 0/0, BusyHere 0/0

Server Error:
InternalError 0/0, NotImplemented 0/0,
BadGateway 0/0, ServiceUnavail 0/0,
GatewayTimeout 0/0, BadSipVer 0/0

Global Failure:
BusyEverywhere 0/0, Decline 0/0,
NotExistAnywhere 0/0, NotAcceptable 0/0

SIP Total Traffic Statistics (Inbound/Outbound)
Invite 3/7, Ack 2/1, Bye 0/2,
Cancel 0/0, Options 0/0
Retry Statistics
Invite 2, Bye 0, Cancel 0, Response 1
```

- **debug ccsip**—Displays the different **debug ccsip** commands.

```
router# debug ccsip ?

all      Enable all SIP debugging traces
calls    Enable CCSIP SPI calls debugging trace
error    Enable SIP error debugging trace
events   Enable SIP events debugging trace
messages Enable CCSIP SPI messages debugging trace
states   Enable CCSIP SPI states debugging trace
```

The following is a sample of debug output from one side of a call:

```
Router1# debug ccsip all
All SIP call tracing enabled
Router1#
*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
Router1#
```

The debug output is as follows from the other side of the call:

```
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
```

Troubleshooting the Cisco SIP Proxy Server

This section describes troubleshooting features and tips for the Cisco SIP proxy server. It provides tips under the following headings:

Troubleshooting Tips

When trying to troubleshoot problems with the Cisco SIP proxy server, remember the following:

- Each module with the Cisco SIP proxy server has debugging capabilities that can be set via a debug flag in the sipd.conf file. When these debug flags are set to On, and the server is running in multi-process mode, the debug output is written to the ./logs/error_log file. When the flags are set to On and single-process mode is enabled, the debug output is written to standard output.
- Changes to the sipd.conf file do not automatically take effect. To have any changes take effect, issue a graceful restart by issuing the following command:

  ./sipdctl graceful

Cisco SIP Proxy Server Does Not Start

If the Cisco SIP proxy server does not start, perform the following tasks as necessary to determine the cause:

- Verify that the /usr/local/sip directory (on Linux) or the opt/local/sip/ directory (on Solaris) has the read and write permissions set that allow you to start the Cisco SIP proxy server.
- Verify that the LD_LIBRARY_PATH environment variable has been enabled as defined in the *Cisco SIP Proxy Server Administrator Guide*, version 2.0.
- If using the Linux RPM version of the Cisco SIP proxy server, verify that the software has been correctly installed.
- Verify that an older version of the Cisco SIP proxy server is not still running. Issue the following command:

  ps -ef | grep -i sip

  If another version is running, disable these processes by issuing the following command:

  ./sipdctl stop

- Verify that the Cisco SIP Proxy Server can resolve its hostname through DNS.

Cisco SIP Proxy Server Does Not Allow Devices to Register

If the Cisco SIP proxy server does not allow devices to register, perform the following tasks as necessary to determine the cause:

- Verify that registration services are enabled in the mod_sip_registry module in the sipd.conf file.
- If authentication is required, ensure that the SIP UA and a password are defined in the MySQL database subscriber table and that the Cisco SIP proxy server can connect to the MySQL database.
- Verify which type of HTTP Digest authentication method that the SIP UAs are using.

Cisco SIP Proxy Server Does Not Route Calls Properly

If the Cisco SIP proxy server does not properly route calls, perform the following tasks as necessary to determine the cause:

- Verify that numbering plan statements are configured correctly in the mod_sip_numexpand module in the sipd.conf file.
- Verify that the translation modules (mod_sip_registry, mod_sip_enum, and mod_sip_gktmp are correctly configured and have the correct entries populated.
- Verify that the correct routes exist in the static routing table of the sipd.conf file.
- Verify that the DNS server is configured for DNS SRV and DNS A records of the devices to be routed.
- View the error_log file for error messages (bad SIP messages, process errors, and so forth.).

Cisco SIP Proxy Server Reports That SIP Messages Are Bad

If the Cisco SIP proxy server reports SIP messages as bad, enable the StateMachine debug flag in the sipd.conf file and view the SIP messages in the error_log file. The error_log file should contain SIP messages that are received in ASCII format. Check the SIP headers of those messages against the headers defined in RFC 2543 or check the SDP information against the information defined in RFC 2327.

Cisco SIP Proxy Server Farming Does Not Work Correctly

If Cisco SIP proxy server farming does not work correctly, perform the following tasks as necessary to determine the cause:

- Verify that all members of the farm have the same sipd.conf file configuration.
- Verify that all members of the farm have an entry for the other farm members defined in the Cisco_Registry_Farm_Members directive in their sipd.conf file.
- Verify that all members of the farm are running the same version of the Cisco SIP proxy server.
- Verify that all members of the farm are synchronized to the same clock source through Network Time Protocol (NTP).

Voice Quality Problems

SIP uses RTP to transmit media between two endpoints. The Cisco SIP proxy server is only involved with the SIP signaling and not the media. Therefore, voice-quality issues should be determined in the endpoint devices, not the Cisco SIP proxy server, because the media does not pass through it.

SIP Messages and Methods

All SIP messages are either requests from a server or client or responses to a request. The messages are formatted according to RFC 822, “Standard for the format of ARPA internet text messages.” For all messages, the general format is:

- A start line
- One or more header fields
- An empty line
- A message body (optional)

Each line must end with a carriage return-line feed (CRLF).

Requests

SIP uses six types (methods) of requests:

- INVITE—Indicates that a user or service is being invited to participate in a call session
- ACK—Confirms that the client has received a final response to an INVITE request
- BYE—Terminates a call and can be sent by the calling or called party
- CANCEL—Cancels any pending searches but does not terminate a call that has already been accepted
- OPTIONS—Queries the capabilities of servers
- REGISTER—Registers the address listed in the To header field with a SIP server

Responses

The following types of responses are used by SIP and generated by the Cisco SIP proxy server:

- SIP 1*xx*—Informational responses
- SIP 2*xx*—Successful responses
- SIP 3*xx*—Redirection responses
- SIP 4*xx*—Client failure responses
- SIP 5*xx*—Server failure responses
- SIP 6xx—Global failure responses

Registration Process

A registration occurs when a client needs to inform a proxy or redirect server of its location. During this process, the client sends a REGISTER request to the proxy or redirect server and includes the address (or addresses) at which it can be reached.

Invitation Process

An invitation occurs when one SIP endpoint (user A) “invites” another SIP endpoint (user B) to join in a call. During this process, user A sends an INVITE message requesting that user B join a particular conference or establish a two-party conversation. If user B wants to join the call, it sends an affirmative response (SIP 2xx). Otherwise, it sends a failure response (SIP 4xx). Upon receiving the response, user A acknowledges the response with an ACK message. If user A no longer wants to establish this conference, it sends a BYE message instead of an ACK message.

Note For examples of SIP call flows, refer to the "Cisco SIP Proxy Server (CSPS) Call Flows" chapter in the *Cisco SIP Proxy Server Administrator Guide, Version 2.0*.

Troubleshooting MGCP and Related Protocol Interfaces to the IP Network

Media gateway control protocol (MGCP) bases its call control and intelligence in centralized *call agents*, also called media gateway controllers. The call agents issue commands to simple, low-cost endpoints, which are housed in media gateways (MGs), and they also receive event reports from the gateways. MGCP messages between call agents and media gateways are sent over IP/UDP.

For information about configuring MGCP, refer to the *MGCP and Related Protocols Configuration Guide*.

This chapter contains the following sections:

MGCP Overview

Two basic MGCP constructs are endpoints and connections. An endpoint is a source or sink for call data (RTP/IP) that is flowing through the gateway. A common type of endpoint is found at the physical interface between the POTS or PSTN service and the gateway; this type of endpoint might be an analog voice port or a digital DS0 group. There are other types of endpoints as well, and some are logical rather than physical. An endpoint is identified by a two-part endpoint name that contains the name of the entity on which it exists (for example, an access server or router) and the local name by which it is known (for example, a port identifier).

Call agents manage call flow using standard MGCP *commands* that are sent to the endpoints under their control. The commands are delivered in standard ASCII text, and can contain session descriptions transmitted in Session Description Protocol (SDP), a text-based protocol. These messages are sent over IP/UDP.

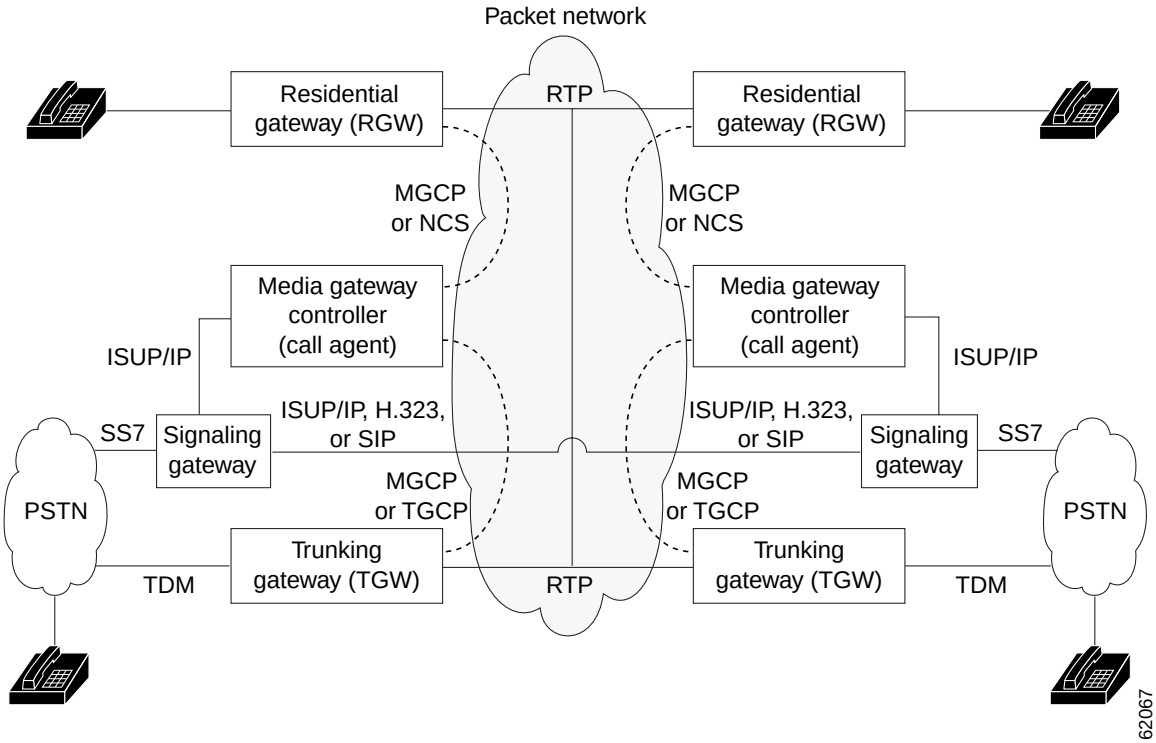
Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Call agents keep track of endpoint and connection status through the gateway's reporting of standard events that are detected from endpoints and connections. Call agents also direct gateways to apply certain standard signals when a POTS/PSTN connection expects them. For example, when someone picks up a telephone handset, an off-hook event is detected on an endpoint on the residential gateway to which the telephone is connected. The gateway reports the event to a call agent, which orders the gateway to apply the dial-tone signal to the endpoint reporting the off-hook event. The person picking up the handset hears dial tone.

Figure 41 shows a hypothetical MGCP network with both residential and trunking gateways. The residential gateway has telephone sets connected to the gateway's FXS voice ports. MGCP or NCS over IP/UDP is used for call control and reporting to the call agent, and Real-time Transmission Protocol (RTP) is used to transmit the actual voice data.

Figure 41 also shows two trunking gateways with T1 (or E1) connections to the PSTN. Incoming time-division multiplexing (TDM) data is sent through the gateway into the packet network through the use of RTP. MGCP or TGCP over IP/UDP is used for call control and reporting to the call agent. Signaling System 7 (SS7) data travels a different route, however, bypassing the trunking gateway entirely in favor of a specialized signaling gateway, where the signaling data is transformed to ISUP/IP format and relayed to the call agent. Communication between two signaling gateways in the same packet network can be done with ISUP/IP, H.323, or Session Initiation Protocol (SIP).

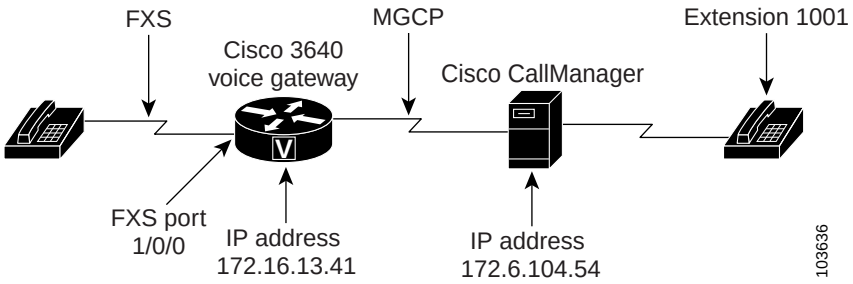
Figure 41 MGCP Network Model



MGCP Gateway Call Flow

The following example shows an MGCP call flow. The network topology is shown in Figure 42 . In this example, the FXS port is registered as an endpoint to Cisco CallManager.

Figure 42 MGCP Call Flow Topology



User Access Verification

In this example, the **show log** command is used to show the MGCP packets sent between the Cisco gateway and Cisco CallManager.

```
Router#
Router#show log
Syslog logging: enabled (0 messages dropped, 1 messages rate-limited, 0 flushes,
 0 overruns, xml disabled)
    Console logging: level debugging, 280 messages logged, xml disabled
    Monitor logging: level debugging, 109 messages logged, xml disabled
    Buffer logging: level debugging, 69 messages logged, xml disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages: disabled
    Trap logging: level informational, 39 message lines logged

Log Buffer (10000000 bytes):
```

The FXS port goes off hook and notifies the CallAgent of the status through observed event. Here, the MGCP packet is sent by the gateway.

```
*Mar  1 02:16:29.399: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:29.399: NTFY 186 aaln/S1/SU0/0@Router MGCP 0.1
X: 1b
O: L/hd
```

Here, the gateway receives the response from the CallAgent.

```
*Mar  1 02:16:29.399: MGCP Packet received from 172.6.104.54-
200 186
```

The CallAgent directs the port to provide the dial tone and requests a notification if there is any change of events such as a port disconnect or digits received.

```
*Mar  1 02:16:29.411: MGCP Packet received from 172.6.104.54-
RQNT 40 AALN/S1/SU0/0@Router MGCP 0.1
X: 1c
R: L/hu, D/[0-9ABCD*#]
S: L/dl
```

```
Q: process,loop

*Mar  1 02:16:29.419: send_mgcp_msg, MGCP Packet sent to 172.6.104.54
```

The FXS port (endpoint) notifies the Call Agent about the digits that it received.

```
*Mar  1 02:16:29.419: 200 40 OK

*Mar  1 02:16:33.595: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:33.595: NTFY 187 AALN/S1/SU0/0@Router MGCP 0.1
X: 1c
O: D/1
*Mar  1 02:16:33.599: MGCP Packet received from 172.6.104.54-
200 187

*Mar  1 02:16:33.603: MGCP Packet received from 172.6.104.54-
RQNT 41 AALN/S1/SU0/0@Router MGCP 0.1
X: 1d
R: L/hu, D/[0-9ABCD*#], L/hf
S:
Q: process,loop

*Mar  1 02:16:33.607: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:33.607: 200 41 OK

*Mar  1 02:16:35.655: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:35.655: NTFY 188 AALN/S1/SU0/0@Router MGCP 0.1
X: 1d
O: D/0

*Mar  1 02:16:35.659: MGCP Packet received from 172.6.104.54-
200 188

*Mar  1 02:16:37.275: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:37.275: NTFY 189 AALN/S1/SU0/0@Router MGCP 0.1
X: 1d
O: D/0

*Mar  1 02:16:37.279: MGCP Packet received from 172.6.104.54-
200 189


*Mar  1 02:16:38.815: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:38.815: NTFY 190 AALN/S1/SU0/0@Router MGCP 0.1
X: 1d
O: D/1
<---
*Mar  1 02:16:38.819: MGCP Packet received from 172.6.104.54-
200 190

*Mar  1 02:16:38.839: MGCP Packet received from 172.6.104.54-
CRCX 42 AALN/S1/SU0/0@Router MGCP 0.1
C: A00000000200001d
X: 1e
M: inactive
R: L/hu
Q: process,loop
```

```
*Mar  1 02:16:38.851: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:38.851: 200 42 OK
I: 4

v=0
c=IN IP4 172.16.13.41
m=audio 19156 RTP/AVP 0 8 99 101 102 2 15 103 4 104 105 106 107 18 100
a=rtpmap:99 G.729a/8000
a=rtpmap:101 G.726-16/8000
a=rtpmap:102 G.726-24/8000
a=rtpmap:103 G.723.1-H/8000
a=rtpmap:104 G.723.1-L/8000
a=rtpmap:105 G.729b/8000
a=rtpmap:106 G.723.1a-H/8000
a=rtpmap:107 G.723.1a-L/8000
a=rtpmap:100 X-NSE/8000
a=fmtp:100 200-202
a=X-sqn:0
a=X-cap: 1 audio RTP/AVP 100
a=X-cpar: a=rtpmap:100 X-NSE/8000
a=X-cpar: a=fmtp:100 200-202
a=X-cap: 2 image udptl t38
<---
*Mar  1 02:16:38.855: MGCP Packet received from 172.6.104.54-
RQNT 43 AALN/S1/SU0/0@Router MGCP 0.1
X: 1f
R: L/hu
S: G/rt
Q: process,loop
```

At this time the call is sent to the IP Phone. The CA directs the IP Phone to start playing ringbacks.

```
*Mar  1 02:16:38.859: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:38.859: 200 43 OK
<---
*Mar  1 02:16:46.159: MGCP Packet received from 172.6.104.54-
MDCX 44 AALN/S1/SU0/0@Router MGCP 0.1
C: A00000000200001d
I: 4
X: 20
L: p:20, a:PCMU, s:off
M: recvonly
R: L/hu
Q: process,loop

*Mar  1 02:16:46.167: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:46.167: 200 44 OK
<---
*Mar  1 02:16:46.171: MGCP Packet received from 172.6.104.54-
RQNT 45 AALN/S1/SU0/0@Router MGCP 0.1
X: 21
R: L/hu, D/[0-9ABCD*#], L/hf
S:
Q: process,loop

*Mar  1 02:16:46.175: send_mgcp_msg, MGCP Packet sent to 172.6.104.54 --->

*Mar  1 02:16:46.175: 200 45 OK

*Mar  1 02:16:46.179: MGCP Packet received from 172.6.104.54-
```

```
MDCX 46 AALN/S1/SU0/0@Router MGCP 0.1
C: A00000000200001d
I: 4
X: 22
L: p:20, a:PCMU, s:off
M: sendrecv
R: L/hu, L/hf, D/[0-9ABCD*#]
S:
Q: process,loop

v=0
o=- 4 0 IN EPN AALN/S1/SU0/0@Router
s=Cisco SDP 0
t=0 0
c=IN IP4 10.17.179.33
m=audio 28390 RTP/AVP 96
a=rtpmap:96 PCMU

*Mar  1 02:16:46.187: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:46.191: 200 46 OK
```

Now the call is answered. Note that media is cut in both directions.After about 12 seconds the called party on the IP Phone drops the call.

```
*Mar  1 02:16:58.355: MGCP Packet received from 172.6.104.54-
MDCX 47 AALN/S1/SU0/0@Router MGCP 0.1
C: A00000000200001d
I: 4
X: 23
M: recvonly
R: L/hu
Q: process,loop

*Mar  1 02:16:58.359: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:58.359: 200 47 OK

*Mar  1 02:16:58.379: MGCP Packet received from 172.6.104.54-
DLCX 48 AALN/S1/SU0/0@Router MGCP 0.1
C: A00000000200001d
I: 4
X: 24
R: L/hu
S:
Q: process,loop

*Mar  1 02:16:58.383: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:16:58.383: 250 48 OK
P: PS=608, OS=97280, PR=604, OR=96640, PL=0, JI=64, LA=0

*Mar  1 02:17:01.403: MGCP Packet received from 172.6.104.54-
RQNT 49 AALN/S1/SU0/0@Router MGCP 0.1
X: 25
R: L/hu, D/[0-9ABCD*#]
S: L/dl
Q: process,loop
```

Since this call was made from an FXS phone the calling party hears a dial tone

```
*Mar  1 02:17:01.411: send_mgcp_msg, MGCP Packet sent to 172.6.104.54
```

```
*Mar  1 02:17:01.411: 200 49 OK

*Mar  1 02:17:03.135: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:17:03.135: NTFY 191 AALN/S1/SU0/0@Router MGCP 0.1
X: 25
O: L/hu

*Mar  1 02:17:03.139: MGCP Packet received from 172.6.104.54-
200 191


*Mar  1 02:17:03.143: MGCP Packet received from 172.6.104.54-
RQNT 50 AALN/S1/SU0/0@Router MGCP 0.1
X: 26
R: L/hd
S:
Q: process,loop

*Mar  1 02:17:03.147: send_mgcp_msg, MGCP Packet sent to 172.6.104.54

*Mar  1 02:17:03.147: 200 50 OK
```

Troubleshooting Guidelines

The following suggestions help with troubleshooting:

- Reset the MGCP statistical counters with the **clear mgcp statistics** command.
- If RTP traffic is not getting through, make sure IP routing is enabled. Use the **show rtp statistics** command, then turn on the **debug ip udp** command and track down the MGCP RTP packets.

  ```
  Router# show rtp statistics
  RTP Statistics info:
  No. CallId Xmit-pkts Xmit-bytes Rcvd-pkts  Rcvd-bytes Lost pkts  Jitter Latenc
  1   17492  0x8A      0x5640     0x8A       0x5640     0x0        0x0      0x0

  Router# show rtp statistics
  RTP Statistics info:
  No. CallId Xmit-pkts Xmit-bytes Rcvd-pkts  Rcvd-bytes Lost pkts  Jitter Latenc
  1   17492  0xDA      0x8840     0xDB       0x88E0     0x0        0x160    0x0
  ```

- If an RSIP message is not received by the call agent, make sure that the **mgcp call-agent** command or the MGCP profile **call-agent** command is configured with the correct call agent name or IP address and UDP port. Use the **show mgcp** command or the **show mgcp profile** command to display this information:

  ```
  Router# show mgcp
  MGCP Admin State ACTIVE, Oper State ACTIVE - Cause Code NONE
  MGCP call-agent: 172.29.248.51  Initial protocol service is MGCP, v. 1.0
  ...
  MGCP gateway port: 2727, MGCP maximum waiting delay 3000
  ...

  Router# show mgcp profile
  MGCP Profile nycprofile
  Description: NY branch office configuration
  Call-agent: 10.14.2.200 Initial protocol service is MGCP, v. 1.0
  ...
  ```

- To verify connections and endpoints, use the **show mgcp** command:

```
Router# show mgcp connection
Endpoint  Call_ID(C) Conn_ID(I) (P)ort (M)ode (S)tate (C)odec (E)vent[SIFL]
(R)esult[EA]
1. S0/DS1-1/5      C=F123AB,5,6  I=0x3  P=16506,16602  M=3  S=4  C=1 E=2,0,0,2  R=0,0
2. S0/DS1-1/6      C=F123AB,7,8  I=0x4  P=16602,16506  M=3  S=4  C=1 E=0,0,0,0  R=0,0

Router# show mgcp endpoint
T1/0 ds0-group 0 timeslots 1-24
T1/1 ds0-group 0 timeslots 1-24
T1/2 ds0-group 0 timeslots 1-24
T1/3 ds0-group 0 timeslots 1-24
```

- If an MGCP message is rejected, it might be because the remote media gateway does not support SDP mandatory parameters (the *o*=, *s*=, and *t*= lines). If this is the case, configure the **mgcp sdp simple** command to send SDP messages without those parameters.
- If you notice problems with voice quality, make sure that the **cptone** (voice-port configuration) command is set for the correct country code. Capturing RTP packets from the sniffer might help to debug the problem; you can decide such questions as whether the payload type or timestamps are set correctly.
- To check operation of interfaces, use the **show interface** command.
- To view information about activity on the T1 or E1 line, use the **show controllers** command. Alarms, line conditions, and other errors are displayed. The data is updated every 10 seconds; and every 15 minutes, the cumulative data is stored and retained for 24 hours.
- When necessary, you can enable debug traces for errors, events, media, packets, and parser. The command **debug mgcp packets** can be used to monitor message flow in general. Note that there is always a performance penalty when using debug commands. The sample output below shows the use of the optional **input-hex** keyword to enable display of hexadecimal values.

```
Router# debug mgcp {all | errors | events | packets {input-hex}| parser}

Router# debug mgcp packets input-hex
Media Gateway Control Protocol input packets in hex value debugging is on

MGCP Packet received -
DLCX 49993 * MGCP 0.1

MGCP Packet received in hex -
44 4C 43 58 20 34 39 39 39 33 20 2A 20 4D 47 43 50 20 30 2E 31 A

send_mgcp_msg, MGCP Packet sent --->
250 49993
```

Call Routing and Dial Peers

To troubleshoot xGCP call routing, see the following sections:

- Verifying Digits Received and Sent on the POTS Call Leg, page 9
- Verifying End-to-End VoIP Signaling on the VoIP Call Leg, page 11

Verifying Digits Received and Sent on the POTS Call Leg

Once the on-hook and off-hook signaling are verified to be working correctly, the next step in troubleshooting and debugging a VoIP call is to verify that the correct digits are being received or sent on the voice port (digital or analog). A dial peer is not matched or the switch (CO or PBX) cannot ring the correct station if incomplete or incorrect digits are being sent or received. Some commands that can be used to verify the digits received/sent are:

- **show dialplan number**—This command is used to show which dial peer is reached when a particular telephone number is dialed.
- **debug vtsp session**—This command displays information on how each network indication and application request is processed, signaling indications, and DSP control messages.
- **debug vtsp dsp** —This command displays the digits as they are received by the voice port.
- **debug vtsp all**—This command enables the following debug voice telephony service provider (VTSP) commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**.

show dialplan number

The **show dialplan number** *digit_string* command displays the dial peer that is matched by a string of digits. If multiple dial peers can be matched, they are all shown in the order in which they are matched. The output of this command looks like this:

```
Router# show dialplan number 5000
Macro Exp.: 5000

VoiceOverIpPeer2
        information type = voice,
        tag = 2, destination-pattern = `5000',
        answer-address = `', preference=0,
        group = 2, Admin state is up, Operation
        state is up,
        incoming called-number = `',
        connections/maximum = 0/unlimited,
        application associated:
        type = voip, session-target =
        `ipv4:192.168.10.2',
        technology prefix:
        ip precedence = 5, UDP checksum =
        disabled, session-protocol = cisco,
        req-qos = best-effort,
        acc-qos = best-effort,
dtmf-relay = cisco-rtp,
        fax-rate = voice,
        payload size =  20 bytes
        codec = g729r8,
        payload size =  20 bytes,
        Expect factor = 10, Icpif = 30,
        signaling-type = cas,
        VAD = enabled, Poor QOV Trap = disabled,
        Connect Time = 25630, Charged Units = 0,
        Successful Calls = 25, Failed Calls = 0,
        Accepted Calls = 25, Refused Calls = 0,
        Last Disconnect Cause is "10  ",
        Last Disconnect Text is "normal call
        clearing.",
        Last Setup Time = 84427934.
        Matched: 5000   Digits: 4
        Target: ipv4:192.168.10.2
```

debug vtsp dsp

debug vtsp dsp shows the digits as they are received by the voice port. The following output shows the collection of DTMF digits from the DSP:

```
Router# debug vtsp dsp
Voice telephony call control dsp debugging is on


!-- ACTION: Caller picked up handset and dialed
!-- digits 5000.
!-- The DSP detects DTMF digits. Digit 5 was
!-- detected with ON time of 130msec.


*Mar 10 17:57:08.505: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=5,
*Mar 10 17:57:08.585: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=5,
duration=130
*Mar 10 17:57:09.385: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:09.485: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=150
*Mar 10 17:57:10.697: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:10.825: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=180
*Mar 10 17:57:12.865: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:12.917: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=100

Router# debug vtsp session
Voice telephony call control session debugging is on


!--- <some output have been omitted>
!-- ACTION: Caller picked up handset.
!-- The DSP is allocated, jitter buffers, VAD
!-- thresholds, and signal levels are set.


*Mar 10 18:14:22.865: dsp_set_playout: [1/0/0 (69)]
packet_len=18 channel_id=1 packet_id=76 mode=1
initial=60 min=4 max=200 fax_nom=300
*Mar 10 18:14:22.865: dsp_echo_canceller_control:
[1/0/0 (69)] packet_len=10 channel_id=1 packet_id=66
flags=0x0
*Mar 10 18:14:22.865: dsp_set_gains: [1/0/0 (69)]
packet_len=12 channel_id=1 packet_id=91
in_gain=0 out_gain=65506
*Mar 10 18:14:22.865: dsp_vad_enable: [1/0/0 (69)]
packet_len=10 channel_id=1 packet_id=78
thresh=-38act_setup_ind_ack
*Mar 10 18:14:22.869: dsp_voice_mode: [1/0/0 (69)]
packet_len=24 channel_id=1 packet_id=73 coding_type=1
```

```
voice_field_size=80
VAD_flag=0 echo_length=64 comfort_noise=1
inband_detect=1 digit_relay=2
AGC_flag=0act_setup_ind_ack(): dsp_dtmf_mod
e()act_setup_ind_ack: passthru_mode = 0,
no_auto_switchover = 0dsp_dtmf_mode
(VTSP_TONE_DTMF_MODE)


!-- The DSP is put into "voice mode" and dial-tone is
!-- generated.


*Mar 10 18:14:22.873: dsp_cp_tone_on: [1/0/0 (69)]
packet_len=30 channel_id=1 packet_id=72 tone_id=4
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
```

If you determine that the digits are not being sent or received properly, then you might need to use either a digit-grabber (test tool) or T1 tester to verify that the digits are being sent at the correct frequency and timing interval. If they are being sent “incorrectly” for the switch (CO or PBX), some values on the router or switch (CO or PBX) might need to be adjusted so that they match and the devices can interoperate. These are usually digit duration and interdigit duration values. If the digits appear to be sent correctly, you can also check any number translation tables in the switch (CO or PBX) that might be adding or removing digits.

Verifying End-to-End VoIP Signaling on the VoIP Call Leg

After verifying that voice-port signaling is working properly and that the correct digits have been received, move to the VoIP call control troubleshooting and debugging. The following factors explain why call control debugging can be a complex job:

- H.323 is made up of three layers of call-negotiation and call-establishment: H.225, H.245, and H.323. These protocols use a combination of TCP and UDP to set up and establish a call.
- End-to-End VoIP debugging shows a number of Cisco IOS state-machines, and problems with any state-machine can cause a call to fail.
- End-to-End VoIP debugging can be very verbose and create a lot of debug output.

Call Admission Control

MGCP VoIP call admission control (CAC) has several commands available to analyze call statistics and operation of applications on the gateway. They are classified into these groups for clarity:

- Troubleshooting MGCP
- Troubleshooting MGCP SRC CAC
- Troubleshooting MGCP RSVP CAC
- Troubleshooting MGCP SA Agent CAC

Troubleshooting MGCP

To provide information about the operation of the MGCP application, use the following commands in privileged EXEC mode:

Command	Purpose
Router# **debug mgcp all**	Displays real-time information about MGCP errors, events, media, packets, parser, system resource check (SRC), and VoIP call admission control (CAC)
Router# **debug mgcp errors** {**endpoint** *endpoint-name*}	Displays MGCP errors
Router# **debug mgcp events** {**endpoint** *endpoint-name*}	Displays MGCP events
Router# **debug mgcp media** {**endpoint** *endpoint-name*}	Displays MGCP tone and signal information
Router# **debug mgcp packets** {**endpoint** *endpoint-name* | **input-hex**}	Displays MGCP packet information, with input packets optionally in hexadecimal format
Router# **debug mgcp parser**	Displays MGCP parser and builder information
Router# **debug mgcp src**	Displays MGCP SRC CAC information
Router# **debug mgcp voipcac**	Turns on debugging messages for the VoIP CAC process at the MGCP application layer

Troubleshooting MGCP SRC CAC

To help identify SRC CAC problems, use the following commands in privileged EXEC mode:

Command	Purpose
Router# **show call threshold** {**status** [**unavailable**] | **stats**}	Displays status of configured triggers or statistics for application programming interface (API) calls that were made to global and interface resources
Router# **show mgcp statistics**	Displays MGCP statistics, including those for MGCP SRC VoIP CAC
Router# **clear call threshold stats**	Clears call threshold statistics
Router# **clear mgcp src-stats**	Clears statistics gathered for MGCP SRC CAC
Router# **debug call threshold**	Displays details of trigger actions
Router# **debug mgcp src**	Provides debug information for MGCP SRC CAC calls

Troubleshooting MGCP RSVP CAC

To identify and trace RSVP CAC problems, use the following commands in privileged EXEC mode:

Command	Purpose
`Router# show call fallback cache`	Displays a network congestion level check result if one has been cached
`Router# show call rsvp-sync stats`	Displays statistics for calls that attempted RSVP reservation
`Router# show call rsvp-sync conf`	Displays the configuration settings for RSVP synchronization
`Router# show ip rsvp reservation`	Displays the RSVP-related receiver information currently in the database
`Router# debug call rsvp-sync func-trace`	Displays messages about software functions called by RSVP
`Router# debug call rsvp-sync events`	Displays events that occur during RSVP setup
`Router# debug ip rsvp detail`	Displays detailed information about RSVP-enabled and Subnetwork Bandwidth Manager (SBM) message processing

Troubleshooting MGCP SA Agent CAC

To help identify Service Assurance (SA) Agent CAC problems, use the following commands in privileged EXEC mode:

Command	Purpose
`Router# show call fallback cache`	Displays a network congestion level check result if one has been cached
`Router# debug call fallback probes`	Verifies that probes are being sent correctly
`Router# debug call fallback detail`	Displays details of the VoIP call fallback
`Router# show rtr application {tabular | full}`	Displays global information about the SA agent feature. There are a number of other options for the **show rtr** command; use CLI help to browse a list of choices
`Router# debug rtr error`	Enables logging of SA agent run-time errors
`Router# debug rtr trace`	Traces the execution of an SA agent operation

Verifying Connections and Endpoints

To verify MGCP-controlled endpoints configured for SS7 and ISDN PRI, use the **show mgcp endpoint** command in privileged EXEC mode.

Note The **show mgcp endpoint** command does not show configured endpoints for CAS, including FGD-OS.

The following example shows MGCP endpoints for a NAS package:

```
controller T3 9/0
 framing m23
 cablelength 224
 t1 1-8 controller
!
controller T1 9/0:1
 framing esf
 ds0-group 0 timeslots 1-24 type e&m-fgb mf dnis
!
controller T1 9/0:2
 framing esf
 ds0-group 0 timeslots 1-24 type e&m-fgb dtmf dnis
!
controller T1 9/0:3
 framing esf
 ds0-group 0 timeslots 1-24 type e&m-immediate-start
!
controller T1 9/0:4
 framing esf
 ds0-group 0 timeslots 1-24 type e&m-fgb
!
controller T1 9/0:5
 framing esf
 pri-group timeslots 1-24 service mgcp
!
controller T1 9/0:6
 framing esf
 ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 9/0:7
 framing esf
 extsig mgcp
 guard-timer 10 on-expiry  reject
 pri-group timeslots 1-24 service mgcp
!
controller T1 9/0:8
 framing esf
 extsig mgcp
 guard-timer 10 on-expiry  reject
 ds0-group 0 timeslots 1-24 type none service mgcp
!
```

The following output shows available varieties of CAS:

- 9/0:5—ISDN backhauling
- 9/0:6—SS7
- 9/0:7—ISDN backhauling with NAS package
- 9/0:8—SS7 with NAS package

```
slot7# show mgcp endpoint
T1 S9/0:5 pri-group timeslots 1-23 type backhaul
T1 S9/0:6 ds0-group 0 timeslots 1-24 type none
T1 S9/0:7 pri-group timeslots 1-23 type backhaul
T1 S9/0:8 ds0-group 0 timeslots 1-24 type none

NAS Endpts

T1 9/0:7 ds0-group 0 timeslots 1-24 type none

T1 9/0:8 ds0-group 0 timeslots 1-24 type none
```

```
slot7#
```

MGCP Testing Commands

The **show** commands are useful for displaying the current status of the configuration as well as verifying that the changes that you made took effect. The following commands are described:

show ccm-manager

If your MGCP network includes Cisco CallManager, use this command to verify the active and redundant configured Cisco CallManager servers. This command also indicates if the gateway is currently registered with Cisco CallManager.

Note The following **show ccm-manager** command output was captured in a separated environment.

```
Router# show ccm-manager

MGCP Domain Name: Router
Total number of host: 2
Priority Status Host
============================================================
Primary Registered 10.89.129.210
First backup Backup ready 10.89.129.211
Second backup Undefined
Current active Call Manager: 10.89.129.210
Current backup Call Manager: 10.89.129.211
Redundant link port: 2428
Failover Interval: 30 seconds
Keepalive Interval: 15 seconds
Last keepalive sent: 1d00h (elapsed time: 00:00:03)
Last MGCP traffic time: 1d00h (elapsed time: 00:00:03)
Last switchover time: 04:49:39 from (10.89.129.211)
Switchback mode: Graceful
```

show mgcp

Use this command to verify the status of the router's MGCP parameters. You should see the IP address of the server that you are using (172.16.1.252 in this case.) All of the other parameters were left at their default behavior in this configuration.

```
VG200A# show mgcp
MGCP Admin State ACTIVE, Oper State ACTIVE - Cause Code NONE
MGCP call-agent: 172.16.1.252  Initial protocol service is MGCP
MGCP block-newcalls DISABLED
MGCP dtmf-relay codec all mode out-of-band
MGCP modem passthrough:  CA
MGCP request timeout 500, MGCP request retries 3
MGCP gateway port: 2427, MGCP maximum waiting delay 3000
MGCP restart delay 0, MGCP vad DISABLED
MGCP simple-sdp ENABLED
MGCP codec type g711ulaw, MGCP packetization period 20
MGCP JB threshold lwm 30, MGCP JB threshold hwm 150
MGCP LAT threshold lmw 150, MGCP LAT threshold hwm 300
MGCP PL threshold lwm 1000, MGCP PL threshold hwm 10000
MGCP playout mode is adaptive 60, 4, 200 in msec
MGCP IP ToS low delay disabled, MGCP IP ToS high throughput disabled
MGCP IP ToS high reliability disabled, MGCP IP ToS low cost disabled
MGCP IP precedence 5, MGCP default package: line-package
MGCP supported packages: gm-package dtmf-package trunk-package line-package
                         hs-package

VG200A#
```

Table 50 Explanation of Fields in the show mgcp Command

Field Output	Description
MGCP Admin State ...	The administrative and operational state of the MGCP daemon. The administrative state controls starting and stopping the application using the **mgcp** and **mgcp block-newcalls** commands. The operational state controls normal MGCP operations.
MGCP call-agent	The address of the call agent specified in the **mgcp** command.
MGCP block-newcalls enabled	The state of the **mgcp block-newcalls** command.
MGCP request timeout	The setting for the **mgcp request timeout** command.
MGCP request retries	The setting for the **mgcp request retries** command.
MGCP gateway port	The UDP port specification.
MGCP maximum waiting delay	The setting for the **mgcp max-waiting-delay** command.
MGCP restart delay	The setting for the **mgcp restart-delay** command.
MGCP VAD	The setting for the **mgcp vad** command.
MGCP codec type	The setting for the **mgcp codec** command.
MGCP packetization period	The packetization period parameter setting for the **mgcp codec** command.
MGCP JB threshold low water mark	The jitter buffer minimum threshold parameter setting for the **mgcp quality-threshold** command.

Table 50 Explanation of Fields in the show mgcp Command (continued)

JB threshold high water mark	The jitter buffer maximum threshold parameter setting for the **mgcp quality-threshold** command.
MGCP LAT threshold low water mark	The latency minimum threshold parameter setting for the **mgcp quality-threshold** command.
LAT threshold high water mark	The latency maximum threshold parameter setting for the **mgcp quality-threshold** command.
MGCP PL threshold low water mark	The packet loss minimum threshold parameter setting for the **mgcp quality-threshold** command.
PL threshold high water mark	The packet loss minimum threshold parameter setting for the **mgcp quality-threshold** command.
MGCP IP ToS low delay	The low-delay parameter setting for the **mgcp ip-tos** command.
MGCP IP ToS high throughput	The high-throughput parameter setting for the **mgcp ip-tos** command.
MGCP IP ToS high reliability	The high-reliability parameter setting for the **mgcp ip-tos** command.
MGCP IP ToS low cost	The low-cost parameter setting for the **mgcp ip-tos** command.
MGCP IP precedence	The precedence parameter setting for the **mgcp ip-tos** command.
MGCP default package type	The default-package parameter setting for the **mgcp default-package** command.
Supported MGCP packages	The packages supported in this session.

show mgcp endpoint

Use this command to show the voice ports (endpoints) that are under MGCP control in the router. This command verifies which voice ports have been bound to the MGCP application. This is related to the **application mgcp** command and the **port** commands that were entered when configuring the POTS dial peer.

```
VG200A#show mgcp endpoint
voice-port 1/0/0
voice-port 1/0/1
voice-port 1/1/0
voice-port 1/1/1

VG200A#
```

show mgcp connection

Use this command to display any active MGCP connections. The endpoint in this example is Slot1/Module 1/Port 0. This corresponds to the MGCP Member Configuration identifier in Cisco CallManager. This tells you which port on the router is the endpoint in the call.

In the screen output below there is one active call.

```
VG200A#show mgcp connection
Endpoint        Call_ID(C)             Conn_ID(I) (P)ort     (M)ode (S)tate (C)odec EC
(R)esult[EA]
```

```
1. aaln/S1/SU1/0 C=A000000001000008,23,24 I=0xD       P=16390,0  M=4    S=4,4   CO=1
EC=1  R=0,0

Total number of active calls 1

VG200A#
```

Table 51 Explanation of Fields in the show mgcp connection Command

Field Output	Description
Endpoint	The endpoint for each call shown in the digital endpoint naming convention of slot number (S0) and digital line (DS1-0) number (1).
Call_ID(C)	The MGCP call ID send by the call agent, the internal Call Control Application Programming Interface (CCAPI) call ID for this endpoint, and the peer call legs CCAPI call ID. (CCAPI is an API to provide call control facilities to applications.)
Conn_ID(I)	The connection ID generated by the gateway and sent in the ACK message.
(P)ort	The ports used for this connection. The first port is the local UDP port. The second port is the remote UDP port.
(M)ode	The call mode, where: 0—Indicates an invalid value for mode. 1—Indicates the gateway should only send packets. 2—Indicates the gateway should only receive packets. 3—Indicates the gateway can send and receive packets. 4—Indicates the gateway should neither send nor receive packets. 5—Indicates the gateway should place the circuit in loopback mode. 6—Indicates the gateway should place the circuit in test mode. 7—Indicates the gateway should use the circuit for network access for data. 8—Indicates the gateway should place the connection in network loopback mode. 9—Indicates the gateway should place the connection in network continuity test mode. 10—Indicates the gateway should place the connection in conference mode. All other values are used for internal debugging.
(S)tate	The call state. The values are used for internal debugging purposes.
(C)odec	The codec identifier. The values are used for internal debugging purposes.
(E)vent [SIFL]	Used for internal debugging.
(R)esult [EA]	Used for internal debugging.

show voice port mod_num/slot_num/port_num

Use this command to verify the current status and configuration of the voice ports on the router.

The following is sample output from the **show voice port** command for a foreign exchange office (FXO) voice port:

```
VG200A#show voice port 1/0/0

Foreign Exchange Office 1/0/0 Slot is 1, Sub-unit is 0, Port is 0
 Type of VoicePort is FXO
 Operation State is DORMANT
 Administrative State is UP
 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
 Music On Hold Threshold is Set to -38 dBm
 In Gain is Set to 0 dB
 Out Attenuation is Set to 0 dB
 Echo Cancellation is enabled
 Echo Cancel Coverage is set to 8 ms
 Playout-delay Mode is set to default
 Playout-delay Nominal is set to 60 ms
 Playout-delay Maximum is set to 200 ms
 Connection Mode is normal
 Connection Number is not set
 Initial Time Out is set to 10 s
 Interdigit Time Out is set to 10 s
 Ringing Time Out is set to 180 s
 Companding Type is u-law
 Region Tone is set for US

 Analog Info Follows:
 Currently processing none
 Maintenance Mode Set to None (not in mtc mode)
 Number of signaling protocol errors are 0
 Impedance is set to 600r Ohm
 Wait Release Time Out is 30 s
 Station name None, Station number None

 Voice card specific Info Follows:
 Signal Type is loopStart
 Number Of Rings is set to 1
 Supervisory Disconnect active
 Hook Status is On Hook
 Ring Detect Status is inactive
 Ring Ground Status is inactive
 Tip Ground Status is inactive
 Dial Type is dtmf
 Digit Duration Timing is set to 100 ms
 InterDigit Duration Timing is set to 100 ms
 Pulse Rate Timing is set to 10 pulses/second
 InterDigit Pulse Duration Timing is set to 750 ms
 Percent Break of Pulse is 60 percent
 GuardOut timer is 2000 ms
VG200A#
```

The following is sample output from the **show voice port** command for a foreign exchange station (FXS) voice port:

```
VG200A#show voice port 1/1/0

Foreign Exchange Station 1/1/0 Slot is 1, Sub-unit is 1, Port is 0
 Type of VoicePort is FXS
 Operation State is DORMANT
 Administrative State is UP
 No Interface Down Failure
 Description is not set
 Noise Regeneration is enabled
 Non Linear Processing is enabled
```

```
 Music On Hold Threshold is Set to -38 dBm
 In Gain is Set to 0 dB
 Out Attenuation is Set to 0 dB
 Echo Cancellation is enabled
 Echo Cancel Coverage is set to 8 ms
 Playout-delay Mode is set to default
 Playout-delay Nominal is set to 60 ms
 Playout-delay Maximum is set to 200 ms
 Connection Mode is normal
 Connection Number is not set
 Initial Time Out is set to 10 s
 Interdigit Time Out is set to 10 s
 Ringing Time Out is set to 180 s
 Companding Type is u-law
 Region Tone is set for US

 Analog Info Follows:
 Currently processing none
 Maintenance Mode Set to None (not in mtc mode)
 Number of signaling protocol errors are 0
 Impedance is set to 600r Ohm
 Wait Release Time Out is 30 s
 Station name None, Station number None

 Voice card specific Info Follows:
 Signal Type is loopStart
 Ring Frequency is 25 Hz
 Hook Status is On Hook
 Ring Active Status is inactive
 Ring Ground Status is inactive
 Tip Ground Status is inactive
 Digit Duration Timing is set to 100 ms
 InterDigit Duration Timing is set to 100 ms
 Ring Cadence is defined by CPTone Selection
 Ring Cadence are [20 40] * 100 msec
VG200A#
```

Table 52 Explanation of Fields in the show voice port Command

Field Output	Description
Administrative State	Administrative state of the voice port.
Alias	User-supplied alias for this voice port.
Clear Wait Duration Timing	Time of inactive seizure signal to declare call cleared.
Connection Mode	Connection mode of the interface
Connection Number	Full E.164 telephone number used to establish a connection with the trunk or PLAR mode.
Currently Processing	Type of call currently being processed: none, voice, or fax.
Delay Duration Timing	Maximum delay signal duration for delay dial signaling.
Delay Start Timing	Timing of generation of delayed start signal from detection of incoming seizure.
Dial Type	Out-dialing type of the voice port.
Digit Duration Timing	DTMF Digit duration in milliseconds.
E&M Type	Type of E&M interface.
Echo Cancel Coverage	Echo Cancel Coverage for this port.

Table 52 Explanation of Fields in the show voice port Command (continued)

Echo Cancellation	Whether or not echo cancellation is enabled for this port.
Hook Flash Duration Timing	Maximum length of hook flash signal.
Hook Status	Hook status of the FXO/FXS interface.
Impedance	Configured terminating impedance for the E&M interface.
In Gain	Amount of gain inserted at the receiver side of the interface.
In Seizure	Incoming seizure state of the E&M interface.
Initial Time Out	Amount of time the system waits for an initial input digit from the caller.
InterDigit Duration Timing	DTMF interdigit duration in milliseconds.
InterDigit Pulse Duration Timing	Pulse dialing interdigit timing in milliseconds.
Interdigit Time Out	Amount of time the system waits for a subsequent input digit from the caller.
Maintenance Mode	Maintenance mode of the voice-port.
Music On Hold Threshold	Configured Music-On-Hold Threshold value for this interface.
Noise Regeneration	Whether or not background noise should be played to fill silent gaps if VAD is activated.
Number of signaling protocol errors	Number of signaling protocol errors.
Non-Linear Processing	Whether or not Non-Linear Processing is enabled for this port.
Operations State	Operation state of the port.
Operation Type	Operation of the E&M signal: 2-wire or 4-wire.
Out Attenuation	Amount of attenuation inserted at the transmit side of the interface.
Out Seizure	Outgoing seizure state of the E&M interface.
Port	Port number for this interface associated with the voice interface card.
Pulse Rate Timing	Pulse dialing rate in pulses per second (pps).
Regional Tone	Configured regional tone for this interface.
Ring Active Status	Ring active indication.
Ring Frequency	Configured ring frequency for this interface.
Ring Ground Status	Ring ground indication
Signal Type	Type of signaling for a voice port: loop-start, ground-start, wink-start, immediate, and delay-dial.
Slot	Slot used in the voice interface card for this port.
Sub-unit	Sub-unit used in the voice interface card for this port.
Tip Ground Status	Tip ground indication.
Type of VoicePort	Type of voice port: FXO, FXS, and E&M.
The Interface Down Failure Cause	Text string describing why the interface is down.

Table 52 Explanation of Fields in the show voice port Command (continued)

Wink Duration Timing	Maximum wink duration for wink start signaling.
Wink Wait Duration Timing	Maximum wink wait duration for wink start signaling.

show mgcp statistics

Use this command to show statistical information related to MGCP activity on the router.

```
VG200A#show mgcp statistics
 UDP pkts rx 3791, tx 3830
 Unrecognized rx pkts 0, MGCP message parsing errors 0
 Duplicate MGCP ack tx 0, Invalid versions count 0
 CreateConn rx 12, successful 12, failed 0
 DeleteConn rx 12, successful 12, failed 0
 ModifyConn rx 42, successful 42, failed 0
 DeleteConn tx 0, successful 0, failed 0
 NotifyRequest rx 8, successful 8, failed 0
 AuditConnection rx 0, successful 0, failed 0
 AuditEndpoint rx 20, successful 20, failed 0
 RestartInProgress tx 6, successful 6, failed 0
 Notify tx 3704, successful 3704, failed 0
 ACK tx 68, NACK tx 0
 ACK rx 3703, NACK rx 0

 IP address based Call Agents statistics:
 IP address 172.16.1.252, Total msg rx 3791,
                 successful 3791, failed 0

VG200A#
```

Table 53 Explanation of Fields in the show mgcp statistics Command

Field Output	Description
UDP pkts	The number of UDP packets received (rx) and transmitted (tx).
Unrecognized rx pkts	The number of packets received that are of unknown type.
MGCP message parsing errors	The number of MGCP message parsing errors.
Duplicate MGCP ack tx	The number of duplicate MGCP ACK transmission messages.
Invalid versions count	The number of invalid versions.
CreateConn rx ...	The number of Create Connection messages received from the call agent by the media gateway. Messages received are classified as being successful or failed.
DeleteConn rx ...	The number of Delete Connection messages received from the call agent by the media gateway. Messages received are classified as being successful or failed.
ModifyConn rx ...	The number of Modify Connection messages received from the call agent by the media gateway. Messages received are classified as being successful or failed.
DeleteConn tx ...	The number of Delete Connection messages sent by the call agent. Messages received are classified as being successful or failed.

Table 53 Explanation of Fields in the show mgcp statistics Command (continued)

NotifyRequest rx ...	The number of Notify messages received by the call agent from the media gateway. Messages received are classified as being successful or failed.
AuditConnection rx ...	The number of Audit Connection messages received from the call agent by the media gateway. Messages received are classified as being successful or failed.
AuditEndpoint rx ...	The number of Audit Endpoint messages received from the call agent by the media gateway. Messages received are classified as being successful or failed.
RestartInProgress tx ...	The number of Restart In Progress (RSIP) messages transmitted by the call agent. Messages received are classified as being successful or failed.
Notify tx ...	The number of Notify messages transmitted by the call agent. Messages received are classified as being successful or failed.
ACK tx ...	The number of acknowledgement messages transmitted by the call agent.
NACK tx ...	The number of negative acknowledgement messages transmitted by the call agent.
ACK rx ...	The number of acknowledgement messages received by the gateway.
NACK rx ...	The number of negative acknowledgement messages received by the gateway.
IP address	The IP address of the call agent.
Total msg rx ...	The total number of messages received by the gateway. Messages received are classified as being successful or failed.

Other debug mgcp Commands

Use **debug mgcp** {**all** | **error** | **events** | **packets** | **parser**} when you are experiencing problems that you believe are not related to configuration errors or hardware problems. It is recommended that you keep an example of each **debug** command from a working configuration to use as a baseline for comparison when you are experiencing problems.

Troubleshooting Voice over Frame Relay Interfaces to the IP Network

Voice over Frame Relay (VoFR) enables a router to carry voice traffic (for example, telephone calls and faxes) over a Frame Relay network, using the FRF.11 protocol. This protocol specification defines multiplexed data, voice, fax, dual-tone multifrequency (DTMF) digit-relay, and channel-associated signaling (CAS).

The Cisco VoFR implementation enables you to make dynamic- and tandem-switched calls and Cisco trunk calls. Dynamic-switched calls have dial-plan information included that processes and routes calls based on the telephone numbers. The dial-plan information is contained within dial-peer entries.

This chapter contains the following topics:

Verifying the Voice Connections

To verify switched call voice connections, perform the following tasks:

- Pick up the telephone handset and verify that there is a dial tone.
- Call from a local telephone to the configured dial peer and verify that the call completes.

To verify the FXO-FXS trunk calls to a remote PBX, perform the following tasks:

- Pick up the telephone and listen for a dial tone from the remote PBX.
- Dial a telephone number, so that the remote PBX routes the call.

To verify voice connections, perform the following tasks:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Check the validity of the dial peer and voice port configuration by performing the following tasks:
  - Enter the **show dial-peer voice** command to verify that the data configured is correct.
  - Enter the **show dial-peer voice summary** command to check the validity of the dial peer configurations.
  - Enter the **show voice port** command to show the status of the voice ports.
  - Enter the **show call active voice** with the keyword **brief** to show the call status for all voice ports.
  - Enter the **show voice call** command to check the validity of the voice port configuration.
  - Enter the **show voice dsp** command to show the current status of all DSP voice channels.
  - Enter the **show voice permanent** command to show the status of Cisco trunk permanent calls.
  - Enter the **show call history** command to show the active call table.
- Check the validity of the VoFR configuration on the DLCI by entering the **show frame-relay vofr** command to show the VoFR configuration.

Verifying the Frame Relay Configuration

Check the validity of the configuration by performing the following tasks:

- Enter the **show frame-relay pvc** command to show the status of the PVCs.
- Enter the **show frame-relay vofr** command with the arguments *interface*, *dlci*, and *cid* to show statistics and information on the open subchannels.
- Enter the **show frame-relay fragment** command with the arguments *interface number* and *dlci* to show the Frame Relay fragmentation configuration.
- Enter the **show traffic-shape queue** command to display the traffic-shaping information if Frame Relay traffic shaping is configured. The **queue** option displays the queueing statistics. For more information about traffic shaping, refer to *Frame Relay Traffic Shaping for VoIP and VoFR*, document 14073.

Troubleshooting Tasks

To troubleshoot and resolve configuration issues, perform the following tasks:

- If no calls are going through, ensure that the **frame-relay voice bandwidth** command is configured.
- If VoFR is configured on a PVC and there are problems with data connectivity on that PVC, ensure that the **frame-relay fragment** command has been configured.
- If data is not being transmitted but fragmentation is configured, ensure that Frame Relay traffic shaping is turned on.
- If the problem is with the dial plan or the dial peers, use the **show dial-plan number** command with the argument *dial string* to display which dial peers are being used when a specific number is called.
- If there are problems connecting an FRF.11 trunk call, ensure that the **session protocol** command in dial peer configuration is set to **frf11-trunk**.
- If FRF.11 trunk calls on the Cisco 2600 or Cisco 3600 series routers are being configured, verify that the **called-number vofr** command in dial peer configuration is configured and that its number matches the destination pattern of the corresponding POTS dial peer.

- Ensure that the voice port is set to **no shutdown**.
- Ensure that the serial port or the T1/E1 controller is set to **no shutdown**.
- Toggle the voice port by first entering **shutdown** and then **no shutdown** every time the **connection trunk** or **no connection trunk** command is entered.

Monitoring and Maintaining the VoFR Configuration

To monitor and maintain the VoFR configuration, use the following commands in EXEC mode as needed:

Command	Purpose
`Router# show call active voice [brief]`	Displays the active call table.

`Router# show call history voice [last number] | [brief]`
or
`Router# show call history voice record`
	Displays the call history table.

`Router# show dial-peer voice`	Displays configuration information and call statistics for dial peers.
`Router# show frame-relay fragment`	Displays information about the Frame Relay fragmentation taking place in the Cisco router.
`Router# show frame-relay pvc`	Displays statistics about PVCs for Frame Relay interfaces.
`Router# show frame-relay vofr`	Displays the FRF.11 subchannel information on VoFR DLCIs.
`Router# show interfaces serial`	Displays information about a serial interface.
`Router# show traffic-shape queue`	Displays information about the elements queued at a particular time at the VC (DLCI) level.
`Router# show voice permanent-call`	Displays information about the permanent calls on a voice interface.

Troubleshooting VoFR with QoS

This section provides information you can use to confirm that your VoFR configuration with QoS is working properly.

Certain **show** commands are supported by the Output Interpreter Tool (registered customers only), which allows you to view an analysis of **show** command output.

LLQ and IP RTP Priority Commands

The following **show** and **debug** commands can help you verify your LLQ and IP RTP priority configurations.

- **show policy-map interface serial** *interface#*—This command is useful for viewing the LLQ operation and any drops in the PQ.
- **show policy-map** *policy_map_name*—Displays information about the policy-map configuration.

- **show queue** *interface-type interface-number*—Lists fair queueing configuration information and statistics for a particular interface.
- **debug priority**—Displays PQ events and shows whether dropping occurs in this queue.
- **show class-map** *class_name*—Displays information about the class-map configuration.
- **show call active voice**—Used to check for lost packets at the DSP level.
- **show frame-relay ip rtp header-compression**—Displays RTP header compression statistics.

For more information about low-latency queueing for VoFR, refer to the *Low Latency Queueing for Frame Relay* feature document.

Fragmentation Commands

Use the following **debug** and **show** commands to verify and troubleshoot fragmentation configurations.

- **show frame-relay fragment**—Displays information about the Frame Relay fragmentation taking place in the Cisco router.
- **debug frame-relay fragment**—Displays event or error messages related to Frame Relay fragmentation. It is enabled at the PVC level on the selected interface.

Frame Relay Interface Commands

Use the following **show** commands to verify and troubleshoot the Frame Relay interface configurations.

- **show traffic-shape queue** *interface*—Displays information about the elements queued at the VC data-link connection identifier (DLCI) level. The command is used to verify the operation of IP RTP priority over Frame-Relay. When the link is congested, voice flows are identified with a weight of zero. This indicates that the voice flow is using the PQ.
- **show traffic-shape—**Displays information such as Tc, Bc, Be, and CIR configured values.
- **show frame-relay pvc** *dlci-#*—Displays information such as traffic shaping parameters, fragmentation values, and dropped packets.

For more information about VoIP over Frame Relay with quality of service (QoS), refer to *VoIP over Frame Relay with Quality of Service (Fragmentation, Traffic Shaping, LLQ / IP RTP Priority)*, document 12156.

Troubleshooting VoIP over Frame Relay with Multipoint PVCs and Prioritization

When you are troubleshooting the traffic-shaping and prioritization for a VoIP over Frame Relay network with hub and spoke topology, the configuration of the hub is such that there are two permanent virtual circuits (PVCs) for each remote spoke, and both data and voice are sent over both PVCs.

Note The prioritization and fragmentation discussed in this document applies not only to this scenario but also to a scenario where you may have one PVC with voice and data and another with only data. The data PVCs need to be traffic-shaped just as the voice and data PVCs are. This is because when a single physical pipe is shared, in this case at the hub, the serialization delay affects all data items, regardless of the PVC they are destined for.

The following debug commands can help you troubleshoot traffic-shaping and prioritization for VoIP over Frame Relay.

- **debug priority**—Displays PQ events and indicates whether dropping occurs in this queue. Frame Relay is a special case with respect to policing the priority queue. The PQ is policed to ensure that the fair queues are not starved of bandwidth. When you configure the PQ, you specify in kbps the maximum amount of bandwidth available to that queue. Packets that exceed that maximum are dropped. Originally, the priority queue of a service-policy configured in a Frame Relay map class was policed during periods of congestion and noncongestion. Cisco IOS Release 12.2 removes this exception. PQ is still policed with FRF.12, but other nonconforming packets are only dropped if there is congestion.
- **debug frame-relay fragment**—Displays event or error messages related to Frame Relay fragmentation. The command is only enabled at the PVC level on the selected interface.

```
newyork# debug priority
Priority output queueing debugging is on
newyork#ping 172.16.120.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.120.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/57/60 ms
newyork#
00:42:40: PQ: Serial2/0: ip -> normal
00:42:40: PQ: Serial2/0 output (Pk size/Q 104/2)
00:42:40: PQ: Serial2/0: ip -> normal
00:42:40: PQ: Serial2/0 output (Pk size/Q 104/2)
00:42:40: PQ: Serial2/0: ip -> normal
00:42:40: PQ: Serial2/0 output (Pk size/Q 104/2)
00:42:40: PQ: Serial2/0: ip -> normal
00:42:40: PQ: Serial2/0 output (Pk size/Q 104/2)
00:42:40: PQ: Serial2/0: ip -> normal
00:42:40: PQ: Serial2/0 output (Pk size/Q 104/2)
00:42:41: PQ: Serial2/0 output (Pk size/Q 13/0)
```

```
newyork# debug frame-relay fragment interface serial 2/0 100
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 Serial2/0 dlci 100 only


Serial2/0(i): dlci 100, rx-seq-num 126, exp_seq-num 126, BE bits set, frag_hdr 04 C0 7E

Serial2/0(o): dlci 100, tx-seq-num 82, BE bits set, frag_hdr 04 C0 52
```

For more information about VoIP over Frame Relay with Multipoint PVCs and Prioritization, refer to *VoIP over Frame Relay with Multipoint PVCs and Prioritization*, document 12162.

VoFR Testing Commands

To diagnose problems in switched Frame Relay networks, use one or both of the following privileged EXEC commands:

Command	Purpose
`Router# show frame-relay pvc`	Displays statistics about PVCs for Frame Relay interfaces.
`Router# debug frame-relay switching`	Displays debug messages for switched Frame Relay PVCs.

Troubleshooting Voice over ATM Interfaces to the IP Network

Voice over ATM (VoATM) enables a router to carry voice traffic (for example, telephone calls and faxes) over an ATM network. An ATM network is a cell-switching and multiplexing technology that combines the benefits of circuit switching (constant transmission delay and guaranteed capacity) and packet switching (flexibility and efficiency for intermittent traffic).

All traffic to or from an ATM network is prefaced with a virtual path identifier (VPI) and virtual channel identifier (VCI). A VPI-VCI pair is seen as identifying a single virtual circuit. Each virtual circuit is a private connection to another node on the ATM network. Each virtual circuit is treated as a point-to-point mechanism to another router or host and is capable of supporting bidirectional traffic.

Each ATM node establishes a separate connection to every other node in the ATM network with which it must communicate. All such connections are established by means of a permanent virtual circuit (PVC) or a switched virtual circuit (SVC) with an ATM signaling mechanism. This signaling is based on the ATM Forum User-Network Interface (UNI) Specification version 3.0.

Each virtual circuit is considered a complete and separate link to a destination node. Data can be encapsulated as needed across the connection, and the ATM network disregards the contents of the data. The only requirement is that data be sent to the ATM processor card of the router in a manner that follows the specific ATM adaptation layer (AAL) format.

An ATM connection transfers raw bits of information to a destination router or host. The ATM router takes the common part convergence sublayer (CPCS) frame, carves it up into 53-byte cells, and sends the cells to the destination router or host for reassembly. In AAL5 format, 48 bytes of each cell are used for the CPCS data, and the remaining 5 bytes are used for cell routing. The 5-byte cell header contains the destination VPI-VCI pair, payload type, cell loss priority (CLP), and header error control (HEC) information.

To troubleshoot VoATM issues, see the following sections:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Checking the ATM Configuration

To check for problems with the ATM configuration, perform the following tasks:

SUMMARY STEPS

1. **show dial-peer voice**
2. **show interface**
3. **no shutdown**

DETAILED STEPS

Step 1 Use the **show dial-peer voice** command on the local and remote concentrators to verify that the data is configured correctly on both.

Step 2 Use the **show interface** command to verify that the ATM interface is up.

Step 3 Ensure that the voice port, serial port, and controller T1 0 is set to **no shutdown**.

Note ATM defaults to Interim Local Management Interface (ILMI). If the carrier is using LMI, be sure to configure LMI support on the router.

Verifying the Voice Connection

To verify that the voice connection is working, perform the following steps:

SUMMARY STEPS

1. Verify dial tone.
2. Verify that a call attempt is successful.
3. **show dial-peer voice**
4. **show voice port**
5. **show voice call**
6. show voice dsp

DETAILED STEPS

Step 1 Pick up the telephone handset and verify that a dial tone is present.

Step 2 Make a call from the local telephone to a configured dial peer and verify that the call attempt is successful.

Step 3 Use the **show dial-peer voice** command to verify that the configured data is correct.

Step 4 Use the **show voice port** command to show the status of the voice ports.

Step 5 Use the **show voice call** command to show the call status for all voice ports.

Step 6 Use the **show voice dsp** command to show the current status of all DSP voice channels.

Verifying the ATM Interface Configuration

To verify the ATM interface configuration, perform the following tasks:

- Enter the privileged EXEC **show atm vc** command to view the SVC (data only) and PVC set. The following is sample output:

```
Router# show atm vc

VCD /                                        Peak  Avg/Min Burst
Interface   Name       VPI   VCI  Type   Encaps   SC   Kbps   Kbps   Cells  Sts
0           1           0     5   PVC    SAAL     UBR       0                UP
0           2           0    16   PVC    ILMI     UBR       0                UP
0           379         0    60   SVC    SNAP     UBR       0                UP
0           986         0    84   SVC    SNAP     UBR       0                UP
0           14          0   133   SVC    VOICE    VBR      64     16   10    UP
0           15          0   134   SVC    VOICE    VBR      64     16   10    UP
0           16          0   135   SVC    VOICE    VBR      64     16   10    UP
0           17          0   136   SVC    VOICE    VBR      64     16   10    UP
0           18          0   137   SVC    VOICE    VBR      64     16   10    UP
0           19          0   138   SVC    VOICE    VBR      64     16   10    UP
0           20          0   139   SVC    VOICE    VBR      64     16   10    UP
0           21          0   140   SVC    VOICE    VBR      64     16   10    UP
0           22          0   141   SVC    VOICE    VBR      64     16   10    UP
0           23          0   142   SVC    VOICE    VBR      64     16   10    UP
0           24          0   143   SVC    VOICE    VBR      64     16   10    UP
0           25          0   144   SVC    VOICE    VBR      64     16   10    UP
0           26          0   145   SVC    VOICE    VBR      64     16   10    UP
0           27          0   146   SVC    VOICE    VBR      64     16   10    UP
0           28          0   147   SVC    VOICE    VBR      64     16   10    UP
```

Note VoATM SVCs are not supported since Cisco IOS Release12.0(7)XK. ATM SVCs for data are still supported.

- Enter the **show atm pvc** command with the VPI/VCI specified to view the PVCs that are set up for ILMI management and Q.SAAL signaling. The following is sample output:

```
Router# show atm pvc 0/5

ATM0: VCD: 2, VPI: 0, VCI: 5, Connection Name: SAAL
UBR, PeakRate: 56
AAL5-SAAL, etype:0x4, Flags: 0x26, VCmode: 0x0
OAM frequency: 0 second(s), OAM retry frequency: 1 second(s), OAM retry frequenc
y: 1 second(s)
OAM up retry count: 3, OAM down retry count: 5
OAM Loopback status: OAM Disabled
OAM VC state: Not Managed
ILMI VC state: Not Managed
InARP DISABLED
InPkts: 2044, OutPkts: 2064, InBytes: 20412, OutBytes: 20580
InPRoc: 2044, OutPRoc: 2064, Broadcasts: 0
InFast: 0, OutFast: 0, InAS: 0, OutAS: 0
OAM cells received: 0
F5 InEndloop: 0, F5 InSegloop: 0, F5 InAIS: 0, F5 InRDI: 0
F4 InEndloop: 0, F4 InSegloop: 0, F4 InAIS: 0, F4 InRDI: 0
```

```
OAM cells sent: 0
F5 OutEndloop: 0, F5 OutSegloop: 0, F5 OutRDI: 0
F4 OutEndloop: 0, F4 OutSegloop: 0, F4 OutRDI: 0
OAM cell drops: 0
Compress: Disabled
Status: INACTIVE, State: NOT_IN_SERVICE
!
Router# show atm pvc 0/16

ATM0: VCD: 1, VPI: 0, VCI: 16, Connection Name: ILMI
UBR, PeakRate: 56
AAL5-ILMI, etype:0x0, Flags: 0x27, VCmode: 0x0
OAM frequency: 0 second(s), OAM retry frequency: 1 second(s), OAM retry frequenc
y: 1 second(s)
OAM up retry count: 3, OAM down retry count: 5
OAM Loopback status: OAM Disabled
OAM VC state: Not Managed
ILMI VC state: Not Managed
InARP DISABLED
InPkts: 398, OutPkts: 421, InBytes: 30493, OutBytes: 27227
InPRoc: 398, OutPRoc: 421, Broadcasts: 0
InFast: 0, OutFast: 0, InAS: 0, OutAS: 0
OAM cells received: 0
F5 InEndloop: 0, F5 InSegloop: 0, F5 InAIS: 0, F5 InRDI: 0
F4 InEndloop: 0, F4 InSegloop: 0, F4 InAIS: 0, F4 InRDI: 0
OAM cells sent: 0
F5 OutEndloop: 0, F5 OutSegloop: 0, F5 OutRDI: 0
F4 OutEndloop: 0, F4 OutSegloop: 0, F4 OutRDI: 0
OAM cell drops: 0
Compress: Disabled
Status: INACTIVE, State: NOT_IN_SERVICE
```

- Enter the **show atm interface** command in privileged EXEC mode and specify ATM 0 to display the ATM interface. The following is sample output from the command:

```
Router# show interface atm 0

ATM0 is up, line protocol is up
  Hardware is PQUICC Atom1
  Internet address is 9.1.1.6/8
  MTU 1500 bytes, sub MTU 1500, BW 1536 Kbit, DLY 20000 usec,
     reliability 255/255, txload 22/255, rxload 11/255
  NSAP address: 47.0091810000000002F26D4901.000011116666.06
  Encapsulation ATM
  292553397 packets input, -386762809 bytes
  164906758 packets output, 1937663833 bytes
  0 OAM cells input, 0 OAM cells output, loopback not set
  Keepalive not supported
  Encapsulation(s):, PVC mode
  1024 maximum active VCs, 28 current VCCs
  VC idle disconnect time: 300 seconds
  Signalling vc = 1, vpi = 0, vci = 5
  UNI Version = 4.0, Link Side = user
  Last input 00:00:00, output 2d05h, output hang never
  Last clearing of "show interface" counters never
  Input queue: -1902/75/0 (size/max/drops); Total output drops: 205
  Queueing strategy: weighted fair
  Output queue: 0/1000/64/0 (size/max total/threshold/drops)
     Conversations  0/0/256 (active/max active/max total)
     Reserved Conversations 0/0 (allocated/max allocated)
  5 minute input rate 67000 bits/sec, 273 packets/sec
  5 minute output rate 136000 bits/sec, 548 packets/sec
     76766014 packets input, 936995443 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
```

```
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
367264676 packets output, 3261882795 bytes, 0 underruns
0 output errors, 0 collisions, 2 interface resets
0 output buffer failures, 0 output buffers swapped out
```

- Enter the **show atm video-voice address** privileged EXEC command to display the ATM interface address and confirm the ILMI status (ILMI PVC is set up to enable SVC management). The ATM interface is assigned automatically with the **atm voice aesa** command. The following is a sample output:

```
Router# show atm video-voice address

nsap address                                      type         ilmi status
47.0091810000000002F26D4901.00107B4832E1.FE   VOICE_AAL5   Confirmed
47.0091810000000002F26D4901.00107B4832E1.C8   VIDEO_AAL1   Confirmed
```

Verifying the VoATM Connection

Verify that the voice connection is working by performing the following steps:

SUMMARY STEPS

1. Verify dial tone.
2. Verify that a call attempt is successful.
3. **show dial-peer voice**
4. **show voice port**
5. **show voice call**
6. show voice dsp

DETAILED STEPS

Step 1 Pick up the handset on a telephone connected to the configuration and verify that there is dial tone.

Step 2 Make a call from the local telephone to a configured dial peer to verify the connection.

Step 3 If there are relatively few dial peers configured, use the **show dial-peer voice** command to verify that the data configured is correct.

Step 4 To show the status of the voice ports, use the **show voice port** command.

Step 5 To show the call status for all voice ports, use the **show voice call** command.

Step 6 To show the current status of all DSP voice channels, use the **show voice dsp** command.

Troubleshooting Tips

If a call does not connect, resolve the problem by performing the following tasks:

- Verify dial peer configuration by using the **show dial-peer voice** command on the local and remote concentrators.
- Verify that ATM interface 0 is up by using the **show interface** command.

- Ensure that the voice port, serial port, and controller T1 0 are set to **no shutdown**.

Troubleshooting Telephony Applications

Troubleshooting Voice Applications

Tcl and VoiceXML applications on the Cisco gateway provide Interactive Voice Response (IVR) features and call control capabilities such as call forwarding and voice mail.

The Cisco voice gateway allows voice applications to be used during call processing. Typically, application scripts contain both executable files and audio files that interact with the system software. Tcl scripts and VoiceXML documents can be stored in any of the following locations: TFTP, FTP, or HTTP servers, Flash memory of the gateway, or on the removable disks of the Cisco 3600 series. The audio files that they reference can be stored in any of these locations, and on Real Time Streaming Protocol (RTSP) servers. A Cisco voice gateway can have several voice applications to accommodate many different services, and you can customize the voice applications to present different interfaces to the various callers. IP phones can also originate calls to a gateway running a voice application.

Voice applications on Cisco gateways can be developed using a choice of two scripting languages:

- Tcl IVR 2.0—Tcl-based scripting with an API.
- VoiceXML—Standards-based markup language for voice browsers.

Applications can also be developed using a hybrid of both Tcl and VoiceXML. The following sections describe troubleshooting Cisco IOS Tcl and VoiceXML applications:

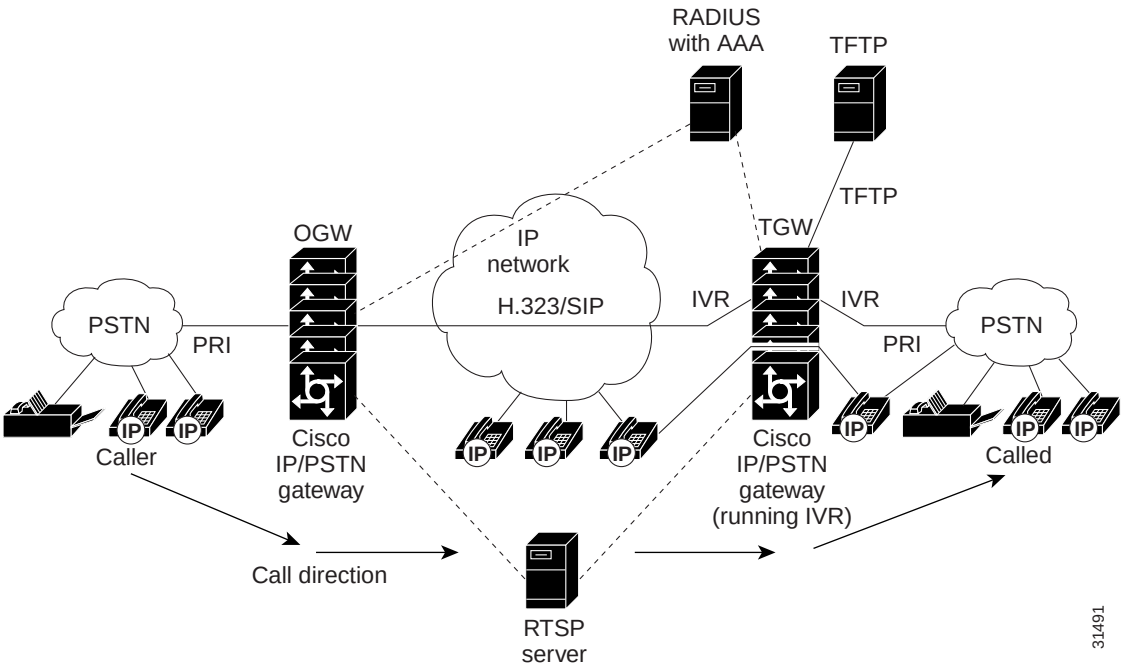
Troubleshooting Tcl IVR

Tcl IVR Version 2.0 uses Tcl scripts to gather data and to process accounting information. For example, a Tcl IVR script can play an audio prompt that asks callers to enter a specific type of information, such as a personal identification number (PIN). After playing the audio prompt, the Tcl IVR application collects the predetermined number of touch tones and sends the collected information to an external server for caller authentication and service authorization.

Figure 43 displays a Tcl IVR application on the gateway.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Figure 43 IVR Control of Tcl Scripts on an IP Call Leg



For information on developing Tcl scripts for voice applications, refer to the *TCL IVR API Version 2.0 Programmer's Guide*.

Refer to the following sections to troubleshoot a Tcl application:

- Testing and Debugging Your Script, page 2
- Loading Your Script, page 3
- Associating Your Script with an Inbound Dial Peer, page 4
- Displaying Information About IVR Scripts, page 4
- Using URLs in IVR Scripts, page 7
- Tips for Using Your Tcl IVR Script, page 8

Testing and Debugging Your Script

It is important to thoroughly test a script before it is deployed. To test a script, you must place it on a router and place a call to activate the script. When you test your script, make sure that you test every procedure in the script and all variations within each procedure.

You can view debugging information applicable to the Tcl IVR scripts that are running on the router. The **debug voip ivr** command allows you to specify the type of debug output you want to view. To view debug output, enter the following command in privileged-EXEC mode:

```
[no] debug voip ivr {states | error | tclcommands | callsetup | digitcollect | script |
dynamic | applib | settlement | all}
```

For more information about the **debug voip ivr** command, see the *Cisco IOS Debug Command Reference*.

The output of any Tcl **puts** commands is displayed if script debugging is on.

Possible sources of errors are:

- An unknown or misspelled command (for example, if you misspell media play as mediaplay)
- A syntax error (such as, specifying an invalid number of arguments)
- Executing a command in an invalid state (for example, executing the **media pause** command when no prompt is playing)
- Using an information tag (info-tag) in an invalid scope (for example, specifying evt_dcdigits when not handling the ev_collectdigits_done event).

In most cases, an error such as these causes the underlying infrastructure to disconnect the call legs and clean up.

Loading Your Script

To associate an application with your Tcl IVR script, use the following command:

```
(config)# call application voice application_name script_url
```

After you associate an application with your Tcl IVR script, use the following command to configure parameters:

```
(config)# call application voice application_name script_url [parameter value]
```

In this command:

- *application_name* specifies the name of the Tcl application that the system is to use for the calls configured on the inbound dial peer. Enter the name to be associated with the Tcl IVR script.
- *script_url* is the pathname where the script is stored. Enter the pathname of the storage location first and then the script filename. Tcl IVR scripts can be stored in Flash memory or on a server that is acceptable using a URL, such as a TFTP server.
- *parameter value* allows you to configure values for specific parameters, such as language or PIN length.

For more information about the **call application voice** command, refer to the *Interactive Voice Response Version 2.0 on Cisco VoIP Gateways* document on Cisco.com.

In the following example, the application named “test” is associated with the Tcl IVR script called newapp.tcl, which is located at tftp://keyer/debit_audio/:

```
(config)# call application voice test tftp://keyer/debit_audio/newapp.tcl
```

Note If the script cannot be loaded, it is placed in a retry queue and the system periodically retries to load it. If you modify your script, you can reload it using only the script name:**(config)# call application voice load** *script_name*

For more information about using the **call application voice** and **call application voice load** commands, refer to the *Cisco IOS TCL IVR and VoiceXML Application Guide*. For more information about these commands, refer to the *Cisco IOS Voice Command Reference*.

Associating Your Script with an Inbound Dial Peer

To invoke your Tcl IVR script to handle a call, you must associate the application configured with an inbound dial peer. To associate your script with an inbound dial peer, enter the following commands in configuration mode:

```
(config)# dial-peer voice number voip

(conf-dial-peer)# incoming called-number destination_number

(conf-dial-peer)# application application_name
```

In these commands:

- *number* uniquely identifies the dial peer. (This number has local significance only.)
- *destination_number* specifies the destination telephone number. Valid entries are any series of digits that specify the E.164 telephone number.
- *application_name* is the abbreviated name that you assigned when you loaded the application.

For example, the following commands indicate that the application called “newapp” should be invoked for calls that come in from an IP network and are destined for the telephone number of 125.

```
(config)# dial-peer voice 3 voip
(conf-dial-peer)# incoming called-number 125
(conf-dial-peer)# application newapp
```

For more information about inbound dial peers, refer to *Dial Peer Configuration on Voice Gateway Routers* and the *Cisco IOS TCL IVR and VoiceXML Application Guide*.

Displaying Information About IVR Scripts

To view a list of the voice applications that are configured on the router, use the **show call application voice** command. A one-line summary of each application is displayed.

```
show call application voice {name | summary}
```

In this command:

- *name* indicates the name of the desired IVR application. If you enter the name of a specific application, the system supplies information about that application.
- **summary** indicates that you want to view summary information. If you specify the summary keyword, a one-line summary is displayed about each application. If you omit this keyword, a detailed description of the specified application is displayed.

The following is example output of the **show call application voice** command:

```
Router# show call application voice session2
Idle call list has 0 calls on it.
Application session2
    The script is read from URL tftp://dirt/sarvi/scripts/tcl/app_session.tcl
    The uid-len is 10               (Default)
    The pin-len is 4                (Default)
    The warning-time is 60          (Default)
    The retry-count is 3            (Default)
    It has 0 calls active.

The TCL Script is:
-----------------
# app_session.tcl
```

```
#------------------------------------------------------------------
# Copyright (c) 1998, 1999 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------
#
# This tcl script mimics the default SESSION app
#
#
# If DID is configured, just place the call to the dnis
# Otherwise, output dial-tone and collect digits from the
# caller against the dial-plan.
#
# Then place the call. If successful, connect it up, otherwise
# the caller should hear a busy or congested signal.

# The main routine just establishes the state machine and then exits.
# From then on the system drives the state machine depending on the
# events it receives and calls the appropriate tcl procedure


#---------------------------------
#   Example Script
#---------------------------------


proc init { } {
    global param

    set param(interruptPrompt) true
    set param(abortKey) *
    set param(terminationKey) #

}

proc act_Setup { } {
    global dest
    global beep

    set beep 0
    leg setupack leg_incoming

    if { [infotag get leg_isdid] } {
        set dest [infotag get leg_dnis]
        leg proceeding leg_incoming
        leg setup $dest callInfo leg_incoming
        fsm setstate PLACECALL
    } else {

        playtone leg_incoming tn_dial

        set param(dialPlan) true
        leg collectdigits leg_incoming param
      }


}

proc act_GotDest { } {
    global dest

    set status [infotag get evt_status]

    if {  $status == "cd_004" } {
        set dest [infotag get evt_dcdigits]
```

```
        leg proceeding leg_incoming
        leg setup $dest callInfo leg_incoming

    } else {
        puts "\nCall [infotag get con_all] got event $status while placing an outgoing
call"
        call close
    }
}


proc act_CallSetupDone { } {
    global beep

    set status [infotag get evt_status]

    if { $status == "CS_000"} {

        set  creditTimeLeft [infotag get leg_settlement_time leg_outgoing]

        if { ($creditTimeLeft == "unlimited") ||
             ($creditTimeLeft == "uninitialized") } {
            puts "\n Unlimited Time"
        } else {
            # start the timer for ...
            if { $creditTimeLeft < 10 } {
                set beep 1
                set delay $creditTimeLeft
            } else {
                set delay [expr $creditTimeLeft - 10]
            }
            timer start leg_timer $delay leg_incoming
        }
    } else {
        puts "Call [infotag get con_all] got event $status collecting destination"
        call close
    }
}

proc act_Timer { } {
    global beep
    global incoming
    global outgoing

    set incoming [infotag get leg_incoming]
    set outgoing [infotag get leg_outgoing]

    if { $beep == 0 } {
        #insert a beep ...to the caller
        connection destroy con_all
        set beep 1
    } else {
        media play leg_incoming flash:out_of_time.au
        fsm setstate CALLDISCONNECTED
    }
}

proc act_Destroy { } {
    media play leg_incoming flash:beep.au
}

proc act_Beeped { } {
    global incoming
    global outgoing
```

```
    connection create $incoming $outgoing
}

proc act_ConnectedAgain { } {
    timer start leg_timer 10 leg_incoming
}

proc act_Ignore { } {
# Dummy
    puts "Event Capture"
}

proc act_Cleanup { } {
    call close
}

init

#---------------------------------
#   State Machine
#---------------------------------
  set TopFSM(any_state,ev_disconnected) "act_Cleanup,same_state"
  set TopFSM(CALL_INIT,ev_setup_indication) "act_Setup,GETDEST"
  set TopFSM(GETDEST,ev_digitcollect_done) "act_GotDest,PLACECALL"
  set TopFSM(PLACECALL,ev_setup_done)  "act_CallSetupDone,CALLACTIVE"
  set TopFSM(CALLACTIVE,ev_leg_timer) "act_Timer,INSERTBEEP"
  set TopFSM(INSERTBEEP,ev_destroy_done) "act_Destroy,same_state"
  set TopFSM(INSERTBEEP,ev_media_done)  "act_Beeped,same_state"
  set TopFSM(INSERTBEEP,ev_create_done)    "act_ConnectedAgain,CALLACTIVE"
  set TopFSM(CALLACTIVE,ev_disconnected)   "act_Cleanup,CALLDISCONNECTED"
  set TopFSM(CALLDISCONNECTED,ev_disconnected) "act_Cleanup,same_state"
  set TopFSM(CALLDISCONNECTED,ev_media_done)  "act_Cleanup,same_state"
  set TopFSM(CALLDISCONNECTED,ev_media_done)  "act_Cleanup,same_state"
  set TopFSM(CALLDISCONNECTED,ev_disconnect_done) "act_Cleanup,same_state"
  set TopFSM(CALLDISCONNECTED,ev_leg_timer) "act_Cleanup,same_state"


  fsm define TopFSM  CALL_INIT
```

Using URLs in IVR Scripts

With IVR scripts, you use URLs to call the script and to call the audio files that the script plays. The VoIP system uses Cisco IOS File System (IFS) to read the files, so any IFS-supported URLs can be used, which includes TFTP, FTP, or a pointer to a device on the router.

Note There is a limit of 32 entries in Flash memory, so you may not be able to copy all your audio files into Flash memory.

URLs for Loading the IVR Script

The URL of the IVR script is a standard URL that points to the location of the script. Examples include:

- flash:myscript.tcl—The script called myscript.tcl is being loaded from Flash memory on the router.
- slot0:myscript.tcl—The script called myscript.tcl is being loaded from a device in slot 0 on the router.

- tftp://BigServer/myscripts/betterMouseTrap.tcl—The script called myscript.tcl is being loaded from a server called BigServer in a directory within the tftpboot directory called myscripts.

URLs for Loading Audio Files

URLs for audio files are different from those used to load IVR scripts. With URLs for audio files:

- For static prompts, you can use the IFS-supported URLs as described in the "URLs for Loading the IVR Script" section on page 7.
- For dynamic prompts, the URL is created by the software, using information from the parameters specified for the **media play** command and the language CLI configuration command.

Tips for Using Your Tcl IVR Script

This section provides some answers to frequently asked questions about using Tcl IVR scripts.

- How do I get information from my RADIUS server to the Tcl IVR script?

  After you have performed an authentication and authorization, you can use the **infotag get** command to obtain the credit amount, credit time, and cause codes maintained by the RADIUS server.

- What happens if my script encounters an error?

  When an error is encountered in the script, the call is cleared with a cause of TEMPORARY_FAILURE (41). If the IVR application has already accepted the incoming call, the caller hears silence. If the script has not accepted the incoming call, the caller might hear a fast busy signal.

  If the script exits with an error and IVR debugging is on (as described in the "Testing and Debugging Your Script" section on page 2), the location of the error in the script is displayed at the command line.

Media Inactive Call Detection

The Media Inactive Call Detection feature enhances Cisco IOS behavior for disconnecting a call when an inactive condition is detected. The former behavior automatically disconnected inactive calls. The current feature provides more control for managing these calls.

The Media Inactive Call Detection feature detects inactive (silent) H.323 or SIP call-legs on Cisco IOS-based gateways, and reports this situation to the Tcl IVR 2.0 application (which can disconnect the call). When the Media Inactive Call Detection feature is enabled, Cisco IOS software does not automatically disconnect detected inactive calls. Inactivity is defined as no RTP/RTCP packets for a configurable length of time.

The command-line interface enables system administrators to do the following:

- Use additional filters to display one or more calls detected and reported as inactive
- Manually release a call by entering the called number, the calling number, or the call ID

The Media Inactive Call Detection feature enables a system administrator to view all detected inactive calls that have been reported to the Tcl script. An internal error code (IEC) is generated when an inactive call is requested to be cleared via the CLI command.

The following sections provide configuration information for Media Inactive Call Detection:

- Prerequisites for Media Inactive Call Detection, page 9

Prerequisites for Media Inactive Call Detection

Note Before the following step is done, you should set the inactive timer. This can be done using the **timer receive-rtcp** command in the CLI or it can be set in the script itself using **infotag set media_timer_factor**.

The Media Inactive Call Detection feature requires Cisco IOS Release 12.3(4)T or later.

To enable the Media Inactive Call Detection feature, set the information tag **evt_feature_report** using **media_inactivity** type. For example:

infotag set evt_feature_report media inactivity

Restrictions for Media Inactive Call Detection

- The Media Inactive Call Detection feature does not change the existing behavior for the default session application and Tcl IVR 1.0 or the existing Tcl IVR 2.0 script behavior that does not request the new feature.
- This feature does not support MGCP call legs.
- The Media Inactive Call Detection feature works in IP only. This feature does not include PSTN inactive call detection.
- This feature supports RTP/RTCP media inactivity detection and notification only on H.323 and SIP basic calls.

Information about Media Inactive Call Detection

This section provides information about the configuration of Media Inactive Call Detection.

Improved Functionality of Media Inactive Call Detection

Legacy Functionality

This functionality is an enhancement to the pre-existing Media Inactivity Timer feature, which enables gateways to monitor and disconnect VoIP calls if no RTCP packets are received within a configurable time period.

The Media Inactivity Timer feature requires the configuration of the Cisco IOS **ip rtcp report interval** command and the **timer receive-rtcp** command to enable detection of RTCP packets by the gateway. When these commands are configured, the gateway uses RTCP report detection, rather than RTP packet detection, to determine whether calls on the gateway are still active or should be disconnected.

If no RTCP packets are received in the resulting time period, the call is disconnected.

The **ip rtcp report interval** command configures the RTCP reporting interval in milliseconds in the range of 1 to 65535. The **timer receive-rtcp** command configures the multiplier in the range of 2 to 1000. These values can be adjusted depending on network traffic conditions. Under normal conditions, a value of **5000** for the **ip rtcp report interval** and a value of **5** for the **timer receive-rtcp** are typical.

Current Functionality

The Media Inactive Call Detection feature offers the following:

- The **show call active** command indicates that a call has no RTP or RTCP inactivity.
- The **clear call** command offers options so that an inactive call can be released using the called number or calling number. The **clear call** command has also been enhanced to configure the Q.850 release cause code to be used when the call is released.

Modifications to Information Tags and Internal Error Codes

This feature includes modifications to two information tags, the addition of two information tags, and an internal error code:

- **evt_feature_report**
- **evt_feature_type**
- **evt_feature_param**
- **media_timer_factor**
- **media_inactivity_err**

For more detailed information, refer to the *Cisco IOS TCL and VoiceXML Application Guide* and the *Cisco IOS TCL Programming Guide*.

evt_feature_report

This existing tag has a new event name (**media_inactivity**) added for the Tcl script to request notification of media inactivity detection.

Description

To enable/disable certain feature events to be intercepted by the script

Syntax

Infotag set evt_feature_report {["no_"]*event_names***}**

Where event_name is a list of application event names that define what events should or should not be reported to application when call is active (connected). An event name with "no_" prefix means not to report it.

Mode

Write

Scope

ev_feature

Return Type

None

Direct Mapping

None

Event Names:

fax

modem

modem_phase

hookflash

onhook

offhook

media_inactivity

Example

The following example enables hookflash and disable fax and modem feature events to be received by the script:

infotag set evt_feature_report hookflash nofax no modem

The following example enables media_inactivity event to be received by the script:

infotag set evt_feature_report media_inactivity

evt_feature_type

This existing information tag adds two new event feature types representing media inactivity notification and media activity notification. Note that the script only needs to request for report type media_inactivity. However, after media inactivity is reported and the VoIP RTP starts receiving RTP/RTCP packets again, the event with type media_activity is automatically notified, indicating that the call is back alive.

Description

To return the feature type string when a feature event is received

Syntax

infotag get evt_feature_type

Mode

Read

Scope

ev_feature

Return Type

String

Direct Mapping

None

Event Names

fax

modem

modem_phase

hookflash

onhook

offhook

media_inactivity

media_activity

evt_feature_param

This is a new information tag added so that the Tcl application can pass along parameters related to the feature back to the script. The Media Inactive Call Detection feature uses this new tag to pass the information on whether RTCP packet has been received before the media inactive condition is met.

Description

To return a parameter related to a specific feature event

Syntax

infotag get evt_feature_param *parameter_name*

Mode

Read

Scope

ev_feature

Return Type

String

Direct Mapping

None

Event Parameter

media_inactivity_type—This parameter belongs to feature **media_inactivity**. The return string is:

- **no media received**—Media inactivity detected (no RTP or RTCP packets have been received for a configured amount of time). RTCP packet has been received before media inactivity condition is met.

– **no control info received**—Media inactivity detected (no RTP or RTCP packets have been received for a configured amount of time). No RTCP packet has been received before media inactivity condition is met.

Example

infotag get evt_feature_param media_inactivity_type

media_timer_factor

This new information tag gives the Tcl script the ability to overwrite the configured gateway **receive-rtcp** timer. This value is used to calculate the timeout value used to detect media inactivity.

Description

To set the timer receive-rtcp timer. This new value is used within the scope of the script. It does not change the gateway configuration.

Syntax:

infotag set media_timer_factor *timer_factor*

Mode

Write

Scope

None

Return Type

None

Direct Mapping

None

Timer factor:

An integer between 2 and 1000. This value is the multiple of RTCP report transmission interval—a value of 5 is recommended.

Example:

infotag set media_timer_factor 6

Note If the value specified is not between 2 and 1000, the script generates an error message.

media_inactivity_err

The internal error code used by the Tcl script for this feature is **media_inactivity_err** (common IEC error #8). This IEC is used to disconnect a call where media inactivity is detected and reported.

Configuring Media Inactive Call Detection

This feature is enabled using Tcl IVR 2.0.

This section describes the use of this feature to set parameters for determining call inactive status and then clearing specific calls. These new options are designed to work as filters.

SUMMARY STEPS

1. **enable**
2. Make a call using the Tcl IVR 2.0 script application. While the call is going, the commands in steps 3 and 4 can be used.
3. **show call active voice** [**brief**] [**called-number** *number* | **calling-number** *number* | **compact** | **echo-canceller** | **id** | **media-inactive** {**called-number** *number* | **calling-number** *number*}]
4. **clear call voice causecode** <**1-127**> {**id <1-FFFF>** | **media-inactive}**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`Make a call using the TCL IVR script application`	Starts a call. While the call is going, the commands in steps 3 and 4 can be used.

	Command or Action	Purpose

Step 3	**show call active voice** [**brief**] [**called-number** *number* | **calling-number** *number* | **compact** | **echo-canceller** | **id** | **media-inactive** {**called-number** *number* | **calling-number** *number*}]
	
	Example:
	`Router# show call active voice media-inactive calling-number 4085551234`
	
	`Router# show call active voice brief media-inactive called-number 4085554321`
		Displays calls that have no RTP or RTCP activity.
		• The **brief** keyword shows the brief version of active voice calls.
		• The **called-number** keyword shows only the call with the specified called number pattern.
		• The **calling-number** keyword shows only the call with the specified calling number pattern.
		• The **compact** keyword shows a compact version of the active voice calls.
		• The **echo-canceller** keyword shows echo canceller data for an active voice call.
		• The **id** keyword shows only the call with the specified ID.
		• The **media-inactive** keyword shows only the calls with media inactive being detected and notified.
		• The *number* argument is a sequence of digits representing a full, recognizable telephone number.

Step 4	**clear call voice causecode** <1-127> {**id** <1-FFFF> | **media-inactive**} **Example:** `Router# clear call voice causecode id 112B`	Clears calls that show media inactive and can clear a specific call. • The **causecode** keyword is a Q.850 disconnect cause code. The cause code can be specified as a number 1 through 127. • The **id** keyword can be used so that only the voice call with the specified ID is disconnected. • The **1-FFFF** range is a call identifier as shown in brief format. • The **media-inactive** keyword clears only calls with media inactivity detected and notified.

Output Examples for Media Inactive Call Detection

The examples provided in this section include the following:

- show call active voice brief Command: Example, page 15
- show running config Command: Example, page 17
- Sample Tcl IVR script, page 21

show call active voice brief Command: Example

The existing **show call active voice brief** command has additional media inactive detection data in the IP call leg information.

```
router# show call active voice brief
<ID>: <start>hs.<index> +<connect> pid:<peer_id> <dir> <addr> <state>
  dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes>
```

```
 IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms lost:<lost>/<early>/<late>
  delay:<last>/<min>/<max>ms <codec>
 media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>
           <----------- the above line is new -------------
   MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
   last <buf event time>s dur:<Min>/<Max>s
 FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 Tele <int>: tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l> i/o:<l>/<l> dBm
  MODEMRELAY info:<rcvd>/<sent>/<resent> xid:<rcvd>/<sent> total:<rcvd>/<sent>/<drops>
         speeds(bps): local <rx>/<tx> remote <rx>/<tx>
 Proxy <ip>:<audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
 bw: <req>/<act> codec: <audio>/<video>
  tx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
 rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>


Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2

11DF : 239062hs.1 +2 pid:1 Answer 4085254616 active
 dur 00:01:04 tx:2383/89775 rx:1187/23318
 Tele 3:D:13: tx:45900/2110/0ms g729r8 noise:-69 acom:45  i/0:-71/-29 dBm

11DF : 239684hs.1 +830 pid:1800877 Originate 18008770519 active
 dur 00:00:49 tx:1066/20965 rx:2378/47215
 IP 1.9.57.5:17750 rtt:5ms pl:38190/0ms lost:0/1/0 delay:50/50/70ms g729r8
 media inactive detected:y media cntrl recv:y timestamp: 12595
       <------------------ the above line is new ------------------
Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

Note For calls without media inactivity being detected, the display looks like the following:

```
...
11DF : 239062hs.1 +2 pid:1 Answer 4085254616 active
 dur 00:01:04 tx:2383/89775 rx:1187/23318
 Tele 3:D:13: tx:45900/2110/0ms g729r8 noise:-69 acom:45  i/0:-71/-29 dBm

11DF : 239684hs.1 +830 pid:1800877 Originate 18008770519 active
 dur 00:00:49 tx:1066/20965 rx:2378/47215
 IP 1.9.57.5:17750 rtt:5ms pl:38190/0ms lost:0/1/0 delay:50/50/70ms g729r8
 media inactive detected:y media cntrl recv:n/a timestamp: n/a
       <------------------ the above line is new ------------------
Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

The long display form of active call records generated by the **show call active voice** command is also modified to add the media inactive detected information.

```
Router_5300# show call active voice
```

```
Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2

 GENERIC:
SetupTime=239062 ms
Index=1
PeerAddress=4085254616
PeerSubAddress=
PeerId=1
...
TELE:
ConnectionId=[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
IncomingConnectionId=[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
TxDuration=51250 ms
VoiceTxDuration=2110 ms
...
 GENERIC:
SetupTime=239684 ms
Index=1
PeerAddress=18008770519
PeerSubAddress=
...
VOIP:
ConnectionId[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
IncomingConnectionId[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
RemoteIPAddress=1.9.57.5
...
TranslatedRedirectCalledNumber=
TranslatedRedirectCalledOctet=0x7F
MediaInactiveDetected=yes                <---- new
MediaInactiveTimestamp=12595             <---- new
MediaControlReceived=yes                 <---- new
Username=Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

Note For calls where no media inactivity is detected and notified, the above three fields are displayed as follows:

```
TranslatedRedirectCalledNumber=
TranslatedRedirectCalledOctet=0x7F
MediaInactiveDetected=no                 <---- new
MediaInactiveTimestamp=                  <---- new
MediaControlReceived=                    <---- new
Username=Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

show running config Command: Example

```
router# show running config
Building configuration...
This command has no effect on this line; use modem AT commands instead
This command has no effect on this line; use modem AT commands instead

Current configuration : 13850 bytes
```

```
!
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname "jc5400"
!
no boot startup-test
logging buffered 2000000 debugging
no logging console
enable secret 5 $1$afrj$LWwkVSLZ3cKak3OkHsAMt/
enable password lab
!
username 1111
username 2222 password 0 2222
username 123001 password 0 1001
username cisco
!
!
resource-pool disable
clock timezone GMT -8
tdm clock priority 1 6/0
spe default-firmware spe-firmware-1
aaa new-model
!
!
aaa authentication login h323 local group radius
aaa authentication login telnet none
aaa authorization exec h323 local group radius
aaa authorization exec telnet none
aaa accounting connection h323 start-stop group radius
aaa session-id common
ip subnet-zero
ip cef
ip ftp username dump
ip ftp password dump123
no ip domain lookup
ip host tftp-server1 10.1.1.211
ip host rtsp-server1 10.1.1.211
ip host radius-server1 10.1.1.211
ip host gwip-server1 10.1.1.211
ip host dump-server1 10.1.1.211
!
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
voice call carrier capacity active
!
!
!
voice cause-code
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
ivr prompt memory 16384
!
fax interface-type fax-mail
mta receive maximum-recipients 600
!
!
```

```
!
controller T1 6/0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
 no yellow generation
 no yellow detection
!
controller T1 6/1
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/2
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/3
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/4
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/5
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/6
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/7
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
gw-accounting aaa
!
!
!
interface FastEthernet0/0
 ip address 10.1.1.212 255.255.0.0
 no ip redirects
 no ip mroute-cache
```

```
 duplex auto
 speed auto
 no cdp enable
!
interface FastEthernet0/1
 no ip address
 no ip redirects
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
interface Serial0/0
 no ip address
 no ip mroute-cache
shutdown
 clockrate 2000000
 no cdp enable
!
interface Serial6/0
 no ip address
 shutdown
!
interface Serial0/1
 no ip address
 no ip mroute-cache
 shutdown
 clockrate 2000000
 no cdp enable
!
interface Serial6/0:23
 no ip address
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 isdn bchan-number-order ascending
 no keepalive
 no cdp enable
!
interface Group-Async0
 no ip address
 no ip mroute-cache
 group-range 1/00 1/107
!
ip classless
ip route 10.1.0.0 255.255.0.0 10.1.1.211
ip http server
!
ip pim bidir-enable
ip rtcp report interval 5000
!
!
no logging trap
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
!
radius-server host 10.1.1.211 auth-port 1645 acct-port 1646
radius-server key cisco
radius-server authorization permit missing Service-Type
radius-server vsa send accounting
radius-server vsa send authentication
!
call application voice testapp_JC tftp://10.1.1.211/Scripts/JC_app1.tcl
```

```
call rsvp-sync
!
voice-port 6/0:D
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1000 pots
 application testapp_JC
 incoming called-number 5551001
 direct-inward-dial
 port 6/0:D
!
!
dial-peer voice 3000 voip
 destination-pattern 5551001
 session target ipv4:10.1.1.213
 dtmf-relay h245-signal
codec g711ulaw
!
!
gateway
 timer receive-rtcp 5
!
sip-ua
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
 logging synchronous
line vty 0 4
 password lab
 authorization exec telnet
 login authentication telnet
line 1/00 1/107
 no flush-at-activation
 modem InOut
!
exception core-file jc5400_core
exception protocol ftp
exception dump 10.1.1.211
scheduler allocate 10000 400
end
```

Sample Tcl IVR script

The following is a sample script that is provided for reference purposes only.

silence_detect_demo.tcl

```
#-----------------------------------------------------------------------
# Copyright (c) 2003 by cisco Systems, Inc.
# All rights reserved.
#-----------------------------------------------------------------------
#
# This tcl demo script monitors Media Inactive Call. The Media Inactive Call
# Detection feature detects inactive (silent)H.323 or Sip call-legs on Cisco
```

```
# IOS based gateways, and reports this situation to the TCL IVR 2.0 application
# and TCL IVR application checks for the events and logs those events.
#
# This script is designed to place a call to the dnis if DID is configured.
# Otherwise, output dial-tone and collects digits from the caller against
# the dial-plan. If an inactive condition 'ev_feature' is detected then script
# begins to log the inactivity events.  However, if the VOIP RTP starts receiving
# RTP/RTCP packets again, then event with media_activity type will be automatically
# notified, indicating that the call is back alive.
#
#---------------------------------
#   Example Script
#---------------------------------

proc init { } {
    global param
    global timerFactor

    set param(interruptPrompt) true
    set param(abortKey) *
    set param(terminationKey) #
    set timerFactor 6
}

proc act_Setup { } {
    global dest

    if { [infotag get leg_isdid] } {
        set dest [infotag get leg_dnis]

        leg proceeding leg_incoming
        leg setup $dest callInfo leg_incoming
        fsm setstate PLACECALL
    } else {
        leg setupack leg_incoming
        playtone leg_incoming tn_dial

        set param(dialPlan) true
        leg collectdigits leg_incoming param
    }

}

proc act_GotDest { } {
     global dest

    set status [infotag get evt_status]
    if {  $status == "cd_004" } {
        set dest [infotag get evt_dcdigits]
        leg proceeding leg_incoming
        leg setup $dest callInfo leg_incoming
    } else {
        call close
    }
}

proc act_CallSetupDone { } {

    infotag set evt_feature_report media_inactivity
    infotag set media_timer_factor $timerFactor

    set status [infotag get evt_status]

    if { $status == "ls_000"} {
```

```
    } else {
        call close
    }
}

proc act_EvFeatureReceived { } {
   global timerFactor

   set featureType [infotag get evt_feature_type]

   if {  $featureType == "media_inactivity" } {
       log -s "media inactivity or silence is detected"

       set inactivity_type [infotag get evt_feature_param media_inactivity_type]
       if {  $inactivity_type == "no media received" } {
           log -s "media inactivity, RTCP packet was previously received"
       }
       if {  $inactivity_type == "no control info received" } {
           log -s "media inactivity, no RTCP packet was previously received "
       }

   } elseif { $featureType == "media_activity" } {
       log -s "media activity detected and call is back alive, VOIP RTP starts receiving
RTP/RTCP packets"

   } else {
       log -s "other Events have been detected"
   }
}

proc act_Cleanup { } {

    call close
}

init

#----------------------------------
#   State Machine
#----------------------------------
  set fsm(any_state,ev_disconnected)         "act_Cleanup          same_state"
  set fsm(CALL_INIT,ev_setup_indication)     "act_Setup            GETDEST"
  set fsm(GETDEST,ev_collectdigits_done)     "act_GotDest          PLACECALL"

  set fsm(PLACECALL,ev_setup_done)           "act_CallSetupDone    CALLACTIVE"
  set fsm(CALLACTIVE,ev_feature)             "act_EvFeatureReceived same_state"

  set fsm(CALLACTIVE,ev_disconnected)        "act_Cleanup          CALLDISCONNECT"
  set fsm(CALLDISCONNECT,ev_disconnected)    "act_Cleanup          same_state"
  set fsm(CALLDISCONNECT,ev_disconnect_done) "act_Cleanup          same_state"

  fsm define fsm CALL_INIT

# End the application
```

Troubleshooting Cisco VoiceXML

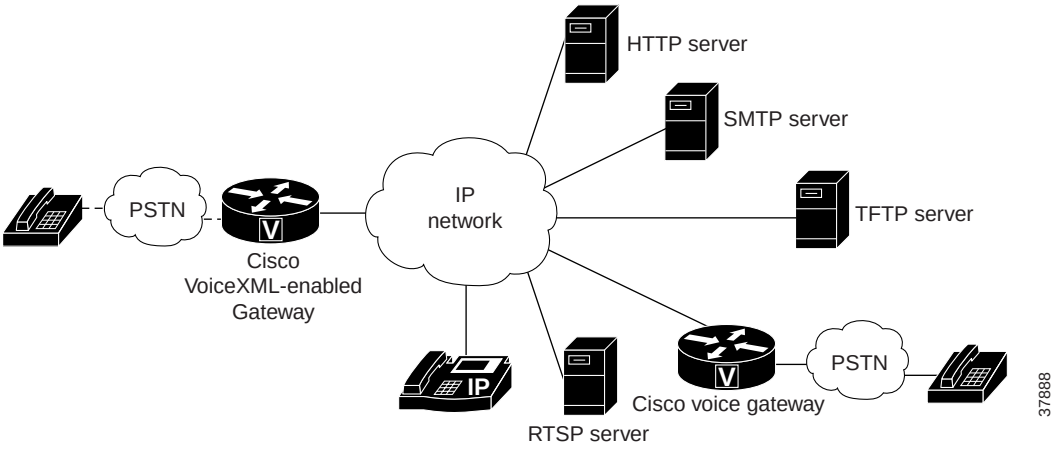
Applications written in Voice eXtensible Markup Language (VoiceXML) provide access through a voice browser to content and services over the telephone, just as Hypertext Markup Language (HTML) provides access through a web browser running on a PC. The universal accessibility of the telephone and its ease of use makes VoiceXML applications a powerful alternative to HTML for accessing the information and services of the World Wide Web.

The Cisco IOS VoiceXML feature provides a platform for interpreting VoiceXML documents. When a telephone call is made to the Cisco VoiceXML-enabled gateway, VoiceXML documents are downloaded from web servers, providing content and services to the caller, typically in the form of pre-recorded audio in an IVR application. Customers can access online business applications over the telephone, providing for example, stock quotes, sports scores, or bank balances.

VoiceXML brings the advantages of web-based development and content delivery to voice applications. It is similar to HTML in its simplicity and in its presentation of information. The Cisco IOS VoiceXML feature is based on the *W3C VoiceXML 2.0 Working Draft* and is designed to provide web developers great flexibility and ease in implementing VoiceXML applications.

Figure 44 shows components that can be configured as a part of a VoiceXML application installed on a Cisco voice gateway:

Figure 44 Cisco IOS VoiceXML Application Components



For information on developing a VoiceXML document for implementing an application on the Cisco voice gateway, refer to the *Cisco VoiceXML Programmer's Guide*.

This section describes some of the troubleshooting techniques for the Cisco VoiceXML features.

For a list of the latest troubleshooting FAQs, go to the developer support website here:

http://www.cisco.com/cgi-bin/dev_support/access_level/products.cgi?product=VOICE_XML_GATEWAY

Debugging Cisco VoiceXML Applications

To debug Cisco VoiceXML applications at the gateway level, refer to the *Cisco IOS TCL and VoiceXML Application Guide*.

This section describes troubleshooting at the script level. To troubleshoot Cisco VoiceXML scripts, enable the **debug vxml error** and **debug vxml puts** commands on the gateway.The **debug vxml error** command displays all errors on the console, and the **debug vxml puts** command prints debugging statements used with the <log> element in the VoiceXML document.

<cisco-debug>

<cisco-debug> is used to debug only a specific application. To disable debugging messages for all VoiceXML applications except the specific VoiceXML application you wish to debug, use the <cisco-debug> element in the VoiceXML document in conjunction with the **debug condition application voice** command.

Refer to the *Cisco IOS TCL and VoiceXML Application Guide* for information on debug commands.

Note Before you use Cisco IOS debug commands to debug a specific application, add <cisco-debug> to the VoiceXML document for the application you want to debug, .

For example:

SUMMARY STEPS

1. Turn on global debug.
2. Add the <cisco-debug enabled = “true”/> and <cisco-debug enabled = “false”/> elements around the specific part of the VoiceXML document where you want to see debugging messages.
3. Add conditional debugging to the specific application.

DETAILED STEPS

Step 1 Turn on global debug for the areas you want to debug. For example:

```
debug vxml application
debug vxml trace
```

Note If you do not proceed with step 2 and end your task with step 1, you see error messages for all the applications, irrespective of conditional debug being turned on or off.

Note The **debug condition application voice** command filters debugging output for only the **debug vxml** and **debug http client** commands. However, it does not filter output for the **debug vxml error**, **debug vxml background**, **debug http client error**, or **debug http client background** commands.

Step 2 Add the <cisco-debug enabled = “true”/> and <cisco-debug enabled = “false”/> elements around the specific part of the VoiceXML document where you want to see debugging messages. For example:

```
<?xml version="1.0"?>
                <vxml version="1.0" application="root.vxml">
```

```
    <form>
        <block>
            <cisco-debug enabled = "true"/>
            <prompt>
                <audio src="welcome.au" caching="fast"/>
            </prompt>
            <cisco-debug enabled = "false"/>
            <goto next="getExtension.vxml?"/>
        </block>
    </form>
</vxml>
```

Step 3 Add conditional debugging to the specific application you want to debug. For example:

Three applications named myapp1, myapp2, and myapp3, all of which can be loaded by using the **call application voice** command are shown below:

```
call application voice myapp1 http://server1/vxml/test1.vxml
call application voice myapp2 http://server2/vxml/test2.vxml
call application voice myapp3 http://server3/vxml/test3.vxml
```

To debug only one of the applications, for example myapp1, use the **debug condition application voice** command to disable debug messages for the other applications, myapp2 and myapp3.

```
debug condition application voice myapp1
```

Note Debugging for myapp1 is performed for only those debug areas that have been enabled in step 1 above. Debugging for the specific session must be enabled through the <cisco-debug> tag as shown in step 2 above.

Error Events

Enabling the **debug vxml error** command displays a list of possible error events on the console. For a list of error events, see the Events and Status Codes section.

Some of the possible errors generated with the **debug vxml error** command enabled are:

error.badfetch

Possible Causes	Suggested Actions
• The VoiceXML interpreter throws this event when there is a failure in retrieving external components in the application. These external components can be VoiceXML documents, prerecorded files, or grammar files. • A badfetch error usually occurs when there is an error in fetching an external document.	• Verify that the external documents, audio prompts, or grammar files are available at the specified location mentioned in the URL. • If the external components are stored on a HTTP server, enable the **debug http client error** command. • If the external components are stored on a RTSP server, search for error.badfetch.rtsp.xxx, where xxx is a RTSP response code. For values of RTSP response codes, refer to RFC 2326 available on the IETF website at http://www.ietf.org/.

error.semantic

Possible Causes	Suggested Actions
Logical errors such as referencing an undefined variable.	Verify that all variables referenced in the script are valid and defined.
Defining different grammar types in the same scope in the VoiceXML application.	Verify that only one grammar type is used at the time of recognizing user input.
Failure to define mandatory parameters in Cisco objects. For example, failure to define the account parameter in the authorize object results in a semantic error.	Verify that all mandatory parameters are defined in Cisco objects used in the script.

error.unsupported.format

Possible Causes	Suggested Actions
A resource format is not supported by the platform.	Verify that all formats used in the script are supported by the specific platforms being used.

JavaScript or ECMA Script

When the <script> element or ECMA expression is used in a VoiceXML document, enable the **debug java** command for debugging.

```
debug java ?
apm2- JavaScript APM2 Utility Debugging
error- JavaScript Error Debugging
interpreter- JavaScript Interpreter Debugging
```

Troubleshooting Speech Recognition and Synthesis

Cisco IOS Release 12.2(11)T and later support automatic speech recognition (ASR) and text-to-speech (TTS) capabilities for VoiceXML and Tcl applications on Cisco voice gateways.

The Speech Recognition and Synthesis feature provides interfaces to ASR and TTS media servers by using Media Resource Control Protocol (MRCP), an application-level protocol developed by Cisco and its ASR and TTS media server partners, Nuance Communications and SpeechWorks International. Client devices that are processing audio or video streams use MRCP to control media resources on external media servers, such as speech synthesizers for TTS and speech recognizers for ASR. The Cisco gateway, running a voice application, and the media servers providing speech recognition and speech synthesis, maintain a client/server relationship through an RTSP connection; the gateway is the RTSP client and the RTSP server is the streaming media server providing speech recognition and speech synthesis.

While doing speech recognition, the gateway creates a separate G.711 u-law RTP stream to the media server, enabling the gateway to simultaneously perform speech synthesis or play audio files using a different codec.

If speech recognition or synthesis is not working, Table 54 lists some possible causes and the actions that you can take.

Table 54 Speech Recognition or Synthesis Fails

Possible Causes	Suggested Actions
Server is not configured either on the Cisco gateway or in the VoiceXML document.	Verify that the server location is configured by using at least one of these methods: • Globally on the gateway by using the **ivr asr-server** or **ivr tts-server** command. Refer to the *Cisco IOS TCL and VoiceXML Application Guide*. • With the *com.cisco.asr-server* or *com.cisco.tts-server* property in the VoiceXML document. Refer to the *Cisco VoiceXML Programmer's Guide.*
Gateway cannot access external ASR or TTS server or server is not running.	Ping the external server to make sure that the gateway has connectivity.
RTSP or MRCP errors are occurring between the gateway and the media server.	See the "Troubleshooting ASR and TTS Server Functionality" section on page 28.

Troubleshooting ASR and TTS Server Functionality

SUMMARY STEPS

1. **debug vxml error** and **debug vxml event**
2. **debug mrcp error**
3. **debug rtsp error**, **debug rtsp session**, and **debug rtsp socket**

DETAILED STEPS

Step 1 Use the **debug vxml error** and **debug vxml event** commands to verify that the external media server is reachable and its location is configured on the gateway or in the VoiceXML document. In the following example, the application failed because the media server is not configured on the gateway or in the VoiceXML document.:

```
Router# debug vxml error
Router# debug vxml event

*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_vapp_tts:
tftp://demo/sample/banking.vxml at line 17: vapp_tts() fail with vapp error 1
**Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_event_proc:
*Jan  5 18:24:19.507:    <event>: event=error.badfetch status=0
*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_default_event_handler:  use default
event handler
*Jan  5 18:24:19.507: //62/36CA25A68036/VAPP:/vapp_session_exit_event_name: Exit Event
error.badfetch
*Jan  5 18:24:19.507: //62/36CA25A68036/VAPP:/vapp_session_exit_event_name: Exit Event
error.badfetch
*Jan  5 18:24:19.507: //62/36CA25A68036/VAPP:/vapp_session_exit_event_name: Exit Event
Name already set to error.badfetch
*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_vapp_terminate: vapp_status=0 ref_count
0
```

```
*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_vapp_terminate: vxml session
terminating with code=ERROR

     vapp status=VAPP_SUCCESS  vxml async status=VXML_ERROR_BAD_FETCH
```

In the following example, the application failed because the media server is either unreachable or is not running.

```
*Jan  5 18:36:44.451: //83/ECD9B163804B/VXML:/vxml_media_done: : media play failed to
setup with VAPP error=31,
                protocol_status_code=0
**Jan  5 18:36:44.451:   <event>: event=error.com.cisco.media.resource.unavailable
status=0
*Jan  5 18:36:44.451: //83/ECD9B163804B/VXML:/vxml_default_event_handler:  use default
event handler
*Jan  5 18:36:44.451: //83/ECD9B163804B/VAPP:/vapp_session_exit_event_name: Exit Event
error.com.cisco.media.resource.unavailable
*Jan  5 18:36:44.451: //83/ECD9B163804B/VAPP:/vapp_session_exit_event_name: Exit Event
error.com.cisco.media.resource.unavailable
*Jan  5 18:36:44.451: //83/ECD9B163804B/VAPP:/vapp_session_exit_event_name: Exit Event
Name already set to error.com.cisco.media.resource.unavailable
*Jan  5 18:36:44.451: //83/ECD9B163804B/VXML:/vxml_vapp_terminate: vapp_status=0 ref_count
0
```

Step 2 Use the **debug mrcp error** command to verify the connection between the gateway and the server. The following example shows the error when the RTSP connection to the server fails:

```
Router# debug mrcp error

*May  9 20:29:09.936:Connecting to 10.1.2.58:554 failed
```

The following error occurs when the response from the server is incorrect:

```
*May  9 20:29:09.936:Response from 10.1.2.58:554 failed
*May  9 20:29:09.936:MRCP/1.0 71 422 COMPLETE
```

The following error occurs when the recognize request comes out of sequence:

```
*May  9 20:29:09.936:act_idle_recognize:ignoring old recognize request
```

Step 3 Use the **debug rtsp error**, **debug rtsp session**, and **debug rtsp socket** commands to verify the RTSP connection with the media server, for example:

The following message displays if the RTSP connection fails:

```
*Sep 25 15:02:32.052: //-1//RTSP:/rtsplib_connect_to_svr: Socket Connect failed:
172.19.140.31:554
```

The following message displays if the RTSP client receives an incorrect response from the server:

```
*Sep 25 15:03:35.062: //-1//RTSP:/rtsp_process_single_svr_resp: Parse Server Response
failed, 172.19.140.31:554
```

The following message displays if the codec configured on the IP side is not G.711:

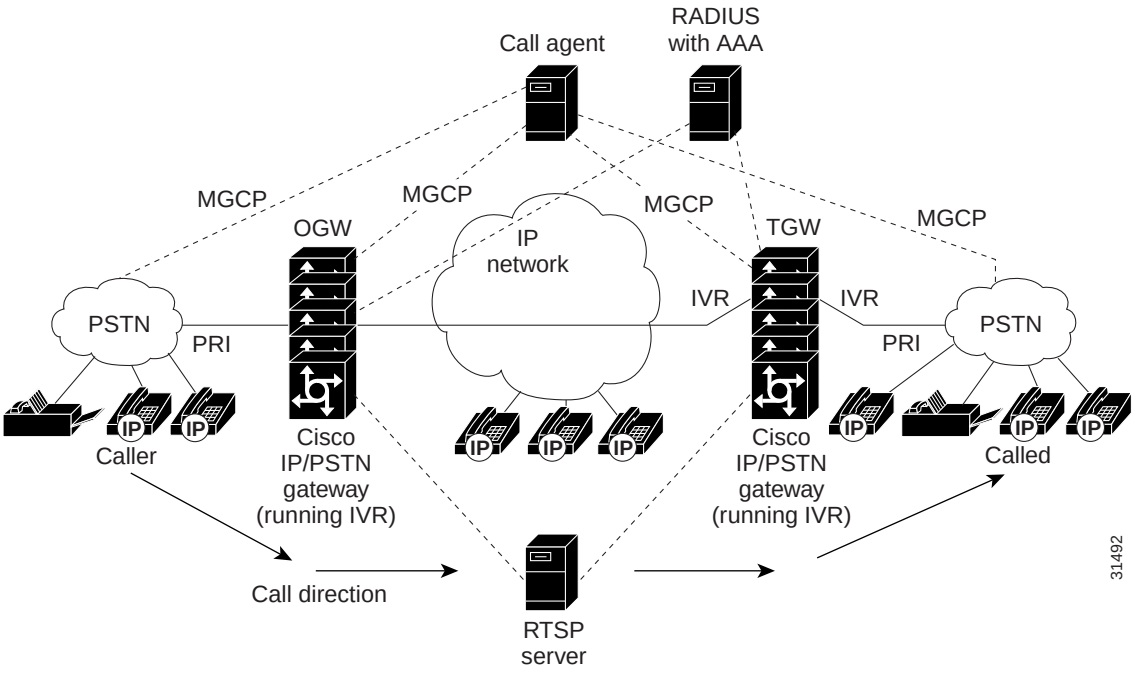
```
*Sep 25 15:05:15.765: //-1//RTSP:/rtsplib_rtp_associate_done: Association mismatch
```

MGCP Scripting Overview

MGCP scripting allows external call agents (CAs) to instruct the Cisco gateway to run a voice application on a PSTN or VoIP call leg. For example, you can request and collect the PIN and account number from a caller. These applications can be written in Tcl 2.0 or VoiceXML.

Figure 45 shows the MGCP CA controlling the application scripts.

Figure 45 MGCP Control of Voice Application Scripts



In the figure above, the RTSP server is configured to interact with gateways that have voice applications installed and running. The RADIUS server also interacts with the gateways to provide authentication, authorization, and accounting (AAA).

For instructions on how to configure your gateway for MGCP, refer to the *MGCP and Related Protocols Configuration Guide*.

Events and Status Codes

This section describes events received and status codes returned by Tcl IVR scripts. This chapter includes the following topics:

Events

The following events can be received by the Tcl IVR script. Any events received that are not included below are ignored.

Event	Description
ev_address_resolved	List of endpoint addresses.
ev_alert	An intermediate event generated by the **leg setup** or **leg setup_continue** commands to set up a call. If specified in the callinfo parameter, *notifyEvents*, the script receives an **ev_alert** message once the destination endpoint is successfully alerted. The script running in the transferee gateway could then disconnect the leg towards the transferring endpoint. If this event is an intercepted event, the application needs to use the **leg setup_continue** command to allow the system to continue with the setup.
ev_any_event	A special wildcard event that can be used in the state machine to represent any event that might be received by the script.
ev_authorize_done	Confirms the completion of the **aaa authorize** command. You can use the evt_status info-tag to determine the authorization status (whether it succeeded or failed).
ev_authenticate_done	Confirms the completion of the authentication command. You can use the evt_status info-tag to determine the authentication status (whether it succeeded or failed).
ev_call_timer0	Indicates that the call-level timer expired.
ev_collectdigits_done	Confirms the completion of the **leg collectdigits** command on the call leg. You can then use the evt_status info-tag to determine the status of the command completion. You can use the evt_dcdigits info-tag to retrieve the collected digits.
ev_connected	An intermediate event generated by the **leg setup** or **leg setup_continue** commands to set up a call. If the callinfo paramater, *notifyEvents*, is specified, the script receives an *ev_connected* message when the system receives a connect event from the destination switch. If this event is an intercepted event, the application needs to use the **leg setup_continue** command to allow the system to continue with the setup.
ev_consult_request	Indicates a call-transfer consultation-id request from an endpoint.
ev_consult_response	Indicates a response to the **leg consult request** command. For return codes, see Consult Status under Status Codes.
ev_consultation_done	Indicated the completion of a **leg consult response** command. For return codes, see Consult Response under Status Codes.
ev_create_done	Confirms the completion of the **connection create** command. You can use the evt_connection info-tag to determine the ID of the completed connection.
ev_destroy_done	Confirms the completion of the **connection destroy** command. You can use the evt_connection info-tag to determine the ID of the connection that was destroyed.

Event	Description
ev_digit_end	Indicates that a digit key is pressed and released. You can use the evt_digit info-tag to determine which digit was pressed. You can use the evt_digit_duration info-tag to determine how long (in seconds) the digit was pressed and to detect long pounds or long digits.
ev_disconnect_done	Indicates that the call leg has been cleared.
ev_disconnected	Indicates that one of the call legs needs to disconnect. On receiving this event, the script must issue a **leg disconnect** on that call leg. You can use the evt_legs info-tag to determine which call leg disconnected.
ev_disc_prog_ind	Indicates that a DISC/PI message is received at a call leg.
ev_facility	Indicates a response to a **leg facility** command.
ev_grab	Indicates that an application that called this script is requesting that the script return the call leg. The script receiving this event can clean up and return the leg with a **handoff return** command. Whether this is done is at the discretion of the script receiving the ev_grab event.
ev_hookflash	Indicates a hook flash (such as a quick onhook-offhook in the middle of a call), assuming that the underlying platform or interface supports hook flash detection.
ev_handoff	Indicates that the script received one or more call legs from another application. When the script receives this event, you can use the evt_legs and the evt_connections info-tags to obtain a list of the call legs and connection IDs that accompanied the ev_handoff event.
ev_leg_timer	Indicates that the leg timer expired. You can use the evt_legs info-tag to determine which leg timer expired.
ev_media_done	Indicates that the prompt playout either completed or failed. You can use the evt_status info-tag to determine the completion status.
ev_proceeding	An intermediate event generated by the **leg setup** or **leg setup_continue** commands to set up a call. If the callinfo paramater, *notifyEvents*, is specified, the script receives an *ev_proceeding* message when the system receives a proceeding event from the remote end. If this event is an intercepted event, the application needs to use the **leg setup_continue** command to allow the system to continue with the setup.
ev_progress	An intermediate event generated by the **leg setup** or **leg setup_continue** commands to set up a call. If the callinfo paramater, *notifyEvents*, is specified, the script receives an *ev_progress* message when the system receives a progress event from the destination switch. If this event is an intercepted event, the application needs to use the **leg setup_continue** command to allow the system to continue with the setup.

Event	Description
ev_returned	Indicates that a call leg that was sent to another application (using **handoff callappl**) has been returned. This event can be accompanied by one or more call legs that were created by the called application. When the script receives this event, you can use the evt_legs and the evt_connections info-tags to obtain a list of the call legs and connection IDs that accompanied the ev_returned event. You can use the evt_iscommand_done info-tag to verify that all of the call legs sent have been accounted for, meaning that the **handoff callappl** command is complete.
ev_setup_done	Indicates that the **leg setup** command has finished. You can then use the evt_status info-tag to determine the status of the command completion (whether the call was successfully set up or failed for some reason).
ev_setup_indication	Indicates that the system received a call. This event and the ev_handoff event are the events that initiate an execution instance of a script.
ev_transfer_request	Indicates a call transfer from an endpoint to the application.
ev_transfer_status	An intermediate event generated by the **leg setup** command. If specified in the callinfo parameter, *notifyEvents*, the script receives an **ev_trasfer_status** message. The ev_status information tag would then contain the status value of the call transfer.
ev_vxmldialog_done	Received when the VXML dialog completes. This could be because of a VXML dialog executing an <exit/> tag or interpretation completing the current document without a transition to another document. The dialog could also complete due to an interpretation failure or a document error. This completion status is also available through the evt_status info-tag.
ev_vxmldialog_event	Received by the Tcl IVR application when the VXML dialog initiated on a leg executes a sendevent object tag. The VXML subevent name is available through the evt_vxmlevent info-tag. All events thrown from the dialog markup are of the form vxml.dialog.*. All events generated by the system—perhaps as an indirect reaction to the VXML document executing a certain tag or throwing a certain event like the dialog completion event—are of the form vxml.session.*.

Status Codes

The evt_status info-tag returns a status code for the event received. This sections lists the possible status codes and their meaning.

Status codes are grouped according to function. The first two characters of the status code indicate the grouping.

- au—Authentication status
- ao—Authorization status
- cd—Digit collection status
- cr—Consult response
- cs—Consult status
- di— Disconnect cause
- fa—Facility

- ft—Feature type
- ls—Leg setup status
- ms—Media status
- ts—Transfer status
- vd—Voice dialog completion status

Authentication Status

Authentication status is reported in **au**_*xxx* format:

Value for *xxx*	Description
000	Authorization was successful.
001	Authorization error.
002	Authorization failed.

Authorization Status

Authorization status is reported in **ao**_*xxx* format:

Value for *xxx*	Description
000	Authorization was successful.
001	Authorization error.
002	Authorization failed.

Digit Collection Status

Digit collection status is reported in **cd**_*xxx* format:

Value for *xxx*	Description
001	The digit collection timed out, because no digits were pressed and not enough digits were collected for a match.
002	The digit collection was aborted, because the user pressed an abort key.
003	The digit collection failed, because the buffer overflowed and not enough digits were collected for a match.
004	The digit collection succeeded with a match to the dial plan.
005	The digit collection succeeded with a match to one of the patterns.
006	The digit collection failed because the number collected was invalid.
007	The digit collection was terminated because an ev_disconnected event was received on the call leg.
008	The digit collection was terminated because an ev_grab event was received on the call leg.
009	The digit collection successfully turned on digit reporting to the script.
010	The digit collection was terminated because of an unsupported or unknown feature or event.

Consult Response

Feature type is reported in **cr**_*xxx* format:

Value for *xxx*	Description
000	Success
001	Failed, invalid state
002	Failed, timeout
003	Failed, abandon
004	Failed, protocol error

Consult Status

Feature type is reported in **cs**_*xxx* format:

Value for *xxx*	Description
000	Consultation success, consult-id available
001	Consultation failed, request timeout
002	Consultation failed
003	Consultation failed, request rejected
004	Consultation failed, leg disconnected
005	Consultation failed, operation unsupported

Disconnect Cause

Disconnect causes use the format **di**_*xxx* where *xxx* is the Q931 cause code. Possible values are:

Value for *xxx*	Description
000	Uninitialized
001	Unassigned number
002	No route to the transit network
003	No route to the destination
004	Send information tone
005	Misdialed trunk prefix
006	Unacceptable channel
007	Call awarded
008	Preemption
009	Preemption reserved
016	Normal
017	Busy
018	No response from the user
019	No answer from the user

Value for *xxx*	Description
020	Subscriber is absent
021	Call rejected
022	Number has changed
026	Selected user is clearing
027	Destination is out of order
028	Invalid number
029	Facility rejected
030	Response to status inquiry
034	No circuit available
035	Requested VPCI VCI is not available
036	VPCI VCI assignment failure
037	Cell rate is not available
038	Network is out of order
039	Permanent frame mode is out of service
040	Permanent frame mode is operational
041	Temporary failure
042	Switch is congested
043	Access information has been discarded
044	No required circuit
045	No VPCI VCI is available
046	Precedence call blocked
047	No resource available
048	DSP error
049	QoS is not available
050	Facility is not subscribed
053	Outgoing calls barred
055	Incoming calls barred
057	Bearer capability is not authorized
058	Bearer capability is not available
062	Inconsistency in the information and class
063	Service or option not available
065	Bearer capability is not implemented
066	Change type is not implemented
069	Facility is not implemented
070	Restricted digital information only
079	Service is not implemented
081	Invalid call reference value

Value for *xxx*	Description
082	Channel does not exist
083	Call exists and call ID in use
084	Call ID in use
085	No call suspended
086	Call cleared
087	User is not in CUG
088	Incompatible destination
090	CUG does not exist
091	Invalid transit network
093	AAL parameters not supported
095	Invalid message
096	Mandatory information element (IE) is missing
097	Message type is not implemented
098	Message type is not compatible
099	IE is not implemented
100	Invalid IE contents
101	Message in incomplete call state
102	Recovery on timer expiration
103	Nonimplemented parameter was passed on
110	Unrecognized parameter message discarded
111	Protocol error
127	Internetworking error
128	Next node is unreachable
129	Holst Telephony Service Provider Module (HTSPM) is out of service
160	DTL transit is not my node ID

Facility

Leg setup requesting address resolution status is reported in **fa**_*xxx* format:

Value for *xxx*	Description
000	supplementary service request succeeded
003	supplementary service request unavailable
007	supplementary service was invoked in an invalid call state
009	supplementary service was invokes in a non-incoming call leg
010	supplementary service interaction is not allowed
050	MCID service is not subscribed
051	MCID request timed out
052	MCID is not configured for this interface

Feature Type

Feature type is reported in **ft**_*xxx* format:

Value for *xxx*	Description
001	Fax
002	Modem
003	Modem_phase
004	Hookflash
005	OnHook
006	OffHook

Leg Setup Status

Leg setup status is reported in **ls**_*xxx* format:

Value for *xxx*	Description
000	The call is active or was successful.
001	The outgoing call leg was looped.
002	The call setup timed out (meaning that the destination phone was alerting, but no one answered). The limit of this timeout can be specified in the **leg setup** command.
003	The call setup failed because of a lack of resources in the network.
004	The call setup failed because of an invalid number.
005	The call setup failed for reasons other than a lack of resources or an invalid number.
006	Unused; setup failure.
007	The destination was busy.
008	The incoming side of the call disconnected.
009	The outgoing side of the call disconnected.
010	The conferencing or connecting of the two call legs failed.
011	Supplementary services internal failure
012	Supplementary services failure
013	Supplementary services failure. Inbound call leg was disconnected.
014	The call was handed off to another application.
015	The call setup was terminated by an application request.
016	The outgoing called number was blocked.
026	Leg redirected
031	Transfer request acknowledge
032	Transfer target alerting (future SIP use)
033	Transfer target trying (future SIP use)
040	Transfer success
041	Transfer success with transfer-to party connected (SIP only)

Value for *xxx*	Description
042	Transfer success unacknowledged (SIP only)
050	Transfer fail
051	Transfer failed, bad request (SIP only)
052	Transfer failed, destination busy
053	Transfer failed, request cancelled
054	Transfer failed, internal error
055	Transfer failed, not implemented (SIP only)
056	Transfer failed, service unavailable or unsupported
057	Transfer failed, leg disconnected
058	Transfer failed, multiple choices (SIP only)
059	Transfer failed, timeout; no response to transfer request

Media Status

Media status is reported in **ms**_*xyy* format:

***x* indicates the command**		***yy* indicates the status of the command**	
Value for *x*	**Description**	**Value for *yy***	**Description**
0	Status for a **media play** command.	00	The command was successful and the prompt finished.[1]
1	Status for a media record command.	01	Failure
2	Status for a **media stop** command.	02	Unsupported feature or request
3	Status for a **media pause** command.	03	Invalid host or URL specified
4	Status for a **media resume** command.	04	Received disconnected
5	Status for a **media seek** command to forward.	05	The prompt was interrupted by a key press.
6	Status for a **media seek** command to rewind.		

1. Valid for the **media play** command only, because media_done events are not received for successful completion of other media commands.

Transfer Status

Transfer status is reported in **ts**_*xxx* format:

Value for *xxx*	Description
000	Generic transfer success
001	Transfer success, transfer-to party is alerting
002	Transfer success, transfer-to party is answered

Value for *xxx*	Description
003	Transfer finished; however, the result of the transfer is not guaranteed
004	Transfer request is accepted
005	Transferee is trying to reach transfer-to party
006	Transfer request is rejected by transferee
007	Invalid transfer number
008	Transfer-to party unreachable
009	Transfer-to party is busy

VoiceXML Dialog Completion Status

VoiceXML dialog completion status is reported in **vd_***xxx* format:

Value for *xxx*	Description
000	Normal completion because of the <exit> tag or execution reaching the end of the document.
001	Termination because of the default VXML event handling requiring VXML termination.
002	Terminated by the Tcl IVR application.
003	Internal failure.

Troubleshooting AAA and Billing Applications

To troubleshoot authentication, authorization, and accounting (AAA), billing, and settlement issues for voice services, refer to the following sections:

Troubleshooting AAA for Voice

This section contains the following topics:

For more information about AAA troubleshooting, go to the "Diagnosing and Troubleshooting AAA Operations" chapter in the *Cisco AAA Implementation Case Study* document.

Using Debug Commands for AAA Voice Troubleshooting

debug radius

The output below is from troubleshooting AAA redirect using called number for an incoming POTS dial peer.

```
!In this example, an incoming call is set up using dial-peer voice 1000 pots. Applying
voice-class aaa 1 to dial-peer voice 1000 redirects AAA requests to the server specified
for method list sanj_aaa1:10.6.20.70 auth-port 1698 acct-port 1699.

aaa group server radius sg1
 server 10.6.20.70 auth-port 1698 acct-port 1699
!
aaa group server radius sg6
 server 10.6.20.70 auth-port 1704 acct-port 1705
```


Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

```
!
aaa group server radius sg7
 server 10.6.20.70 auth-port 1720 acct-port 1721
!
aaa authentication login sanj_aaa1 group sg1
aaa authorization exec sanj_aaa1 group sg1
aaa accounting connection sanj_aaa1 start-stop group sg1
!
aaa authentication login sanj_aaa6 group sg6
aaa authorization exec sanj_aaa6 group sg6
aaa accounting connection sanj_aaa6 start-stop group sg6
!
aaa authentication login sanj_aaa7 group sg7
aaa authorization exec sanj_aaa7 group sg7
aaa accounting connection sanj_aaa7 start-stop group sg7
!
voice class aaa 1
 authentication method sanj_aaa1
 authorization method sanj_aaa1
 accounting method sanj_aaa1
 accounting template cdr1
!
voice class aaa 2
 authentication method sanj_aaa6
 authorization method sanj_aaa6
 accounting method sanj_aaa6
 accounting template cdr2
!
voice class aaa 3
 authentication method sanj_aaa7
 authorization method sanj_aaa7
 accounting method sanj_aaa7
!
dial-peer voice 1000 pots
 application plain_debit
 incoming called-number 12345
 voice-class aaa 1
 port 0:D
!
dial-peer voice 1001 pots
 application plain_debit
 incoming called-number 12346
 voice-class aaa 2
 port 1:D
```

```
debug radius
Radius protocol debugging is on
Radius packet hex dump debugging is off
Radius packet protocol debugging is on
debug isdn q931
ISDN Q931 packets debugging is on

00:17:55: ISDN Se0:23: RX <-  SETUP pd = 8  callref = 0x009D
00:17:55:         Bearer Capability i = 0x8090A2
00:17:55:         Channel ID i = 0xE1808397
00:17:55:         Calling Party Number i = 0x0080, '4081234567', Plan:Unknown,
Type:Unknown
00:17:55:         Called Party Number i = 0xE9, '12345', Plan:Private, Type:Abbreviated
00:17:55: RADIUS/ENCODE(0000000C): Unsupported AAA attribute timezone
00:17:55: RADIUS(0000000C): Encoding nas-port...Only port-type avlbl
00:17:55: RADIUS(0000000C): sending
00:17:55: RADIUS: Send to unknown id 4 10.6.20.70:1699, Accounting-Request, len 262
00:17:55: RADIUS:  authenticator 10 41 58 99 4C F2 B1 CD - 44 3E E3 60 5D 10 C3 A9
```

```
00:17:55: RADIUS:  Acct-Session-Id     [44]  10  "0000000C"
00:17:55: RADIUS:  Vendor, Cisco       [26]  56
00:17:55: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=B8FE8B7F BF1711D3 800CE483
89ADC43B"
00:17:55: RADIUS:  Vendor, Cisco       [26]  31
00:17:55: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
00:17:55: RADIUS:  Vendor, Cisco       [26]  65
00:17:55: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=B8FE8B7F BF1711D3
800CE483 89ADC43B"
00:17:55: RADIUS:  User-Name           [1]   12  "4081234567"
00:17:55: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
00:17:55: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:17:55: RADIUS:  Vendor, Cisco       [26]  19
00:17:55: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
00:17:55: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:17:55: RADIUS:  Called-Station-Id   [30]  7   "12345"
00:17:55: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:17:55: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:17:55: RADIUS:  Delay-Time          [41]  6   0
00:17:55: ISDN Se0:23: TX ->  CALL_PROC pd = 8  callref = 0x809D
00:17:55:         Channel ID i = 0xA98397
00:17:55: ISDN Se0:23: TX ->  CONNECT pd = 8  callref = 0x809D
00:17:55: RADIUS: Received from id 4 10.6.20.70:1699, Accounting-response, len 20
00:17:55: RADIUS:  authenticator DC CD BA E8 7E 02 EA D1 - 12 67 DC 57 3C 73 56 75
00:17:55: ISDN Se0:23: RX <-  CONNECT_ACK pd = 8  callref = 0x009D
00:17:55: ISDN Se0:23: CALL_PROGRESS: CALL_CONNECTED call id 0x63, bchan 22, dsl 0
00:17:55: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4081234567
00:18:01: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4081234567
00:18:06: RADIUS(0000000C): Encoding nas-port...Only port-type avlbl
00:18:06: RADIUS/ENCODE(0000000C): acct_session_id: 12
00:18:06: RADIUS(0000000C): sending
00:18:06: RADIUS: Send to unknown id 3 10.6.20.70:1698, Access-Request, len 199
00:18:06: RADIUS:  authenticator 4B 2C 8C D7 12 54 45 3D - 51 44 30 05 C3 9B 44 B1
00:18:06: RADIUS:  User-Name           [1]   8   "777777"
00:18:06: RADIUS:  User-Password       [2]   18  *
00:18:06: RADIUS:  Vendor, Cisco       [26]  56
00:18:06: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=61A46F2C 00000003 62E66E40
62E3A5C8"
00:18:06: RADIUS:  Vendor, Cisco       [26]  36
00:18:06: RADIUS:   Cisco AVpair       [1]   30  "h323-ivr-out=transactionID:3"
00:18:06: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:18:06: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:18:06: RADIUS:  Vendor, Cisco       [26]  19
00:18:06: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
00:18:06: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:18:06: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:18:06: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:18:06: RADIUS: Received from id 3 10.6.20.70:1698, Access-Accept, len 200
00:18:06: RADIUS:  authenticator 9C AA 9E 4C 64 02 13 3A - 72 8C 3F D9 72 D0 3B 06
00:18:06: RADIUS:  Vendor, Cisco       [26]  27
00:18:06: RADIUS:   Cisco AVpair       [1]   21  "h323-ivr-in=sanjose"
00:18:06: RADIUS:  Vendor, Cisco       [26]  34
00:18:06: RADIUS:   Cisco AVpair       [1]   28  "h323-credit-amount=7777.77"
00:18:06: RADIUS:  Vendor, Cisco       [26]  26
00:18:06: RADIUS:   Cisco AVpair       [1]   20  "h323-return-code=0"
00:18:06: RADIUS:  Vendor, Cisco       [26]  30
00:18:06: RADIUS:   h323-credit-time   [102] 24  "h323-credit-time=54329"
00:18:06: RADIUS:  Vendor, Cisco       [26]  33
00:18:06: RADIUS:   h323-billing-model [109] 27  "h323-billing-model=prepay"
00:18:06: RADIUS:  Vendor, Cisco       [26]  24
00:18:06: RADIUS:   h323-currency      [110] 18  "h323-currency=US"
00:18:06: RADIUS:  Idle-Timeout        [28]  6   30
00:18:06: RADIUS: Received from id C
00:18:27: ISDN Se0:23: RX <-  DISCONNECT pd = 8  callref = 0x009D
```

```
00:18:27:         Cause i = 0x8290 - Normal call clearing
00:18:27: %ISDN-6-DISCONNECT: Interface Serial0:22  disconnected from 4081234567 , call
lasted 32 seconds
00:18:27: ISDN Se0:23: TX ->  RELEASE pd = 8  callref = 0x809D
00:18:27: ISDN Se0:23: RX <-  RELEASE_COMP pd = 8  callref = 0x009D
00:18:27: RADIUS/ENCODE(0000000C): Unsupported AAA attribute timezone
00:18:27: RADIUS(0000000C): Encoding nas-port...Only port-type avlbl
00:18:27: RADIUS(0000000C): sending
00:18:27: RADIUS: Send to unknown id 5 10.6.20.70:1699, Accounting-Request, len 327
00:18:27: RADIUS:  authenticator 2D 65 1C 38 6D 5B B3 DD - C8 57 D6 02 B4 4F E4 4E
00:18:27: RADIUS:  Acct-Session-Id     [44]  10  "0000000C"
00:18:27: RADIUS:  Vendor, Cisco       [26]  56
00:18:27: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=B8FE8B7F BF1711D3 800CE483
89ADC43B"
00:18:27: RADIUS:  Vendor, Cisco       [26]  31
00:18:27: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
00:18:27: RADIUS:  Vendor, Cisco       [26]  65
00:18:27: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=B8FE8B7F BF1711D3
800CE483 89ADC43B"
00:18:27: RADIUS:  Acct-Input-Octets   [42]  6   0
00:18:27: RADIUS:  Acct-Output-Octets  [43]  6   148000
00:18:27: RADIUS:  Acct-Input-Packets  [47]  6   0
00:18:27: RADIUS:  Acct-Output-Packets [48]  6   925
00:18:27: RADIUS:  Acct-Session-Time   [46]  6   32
00:18:27: RADIUS:  Vendor, Cisco       [26]  35
00:18:27: RADIUS:   Cisco AVpair       [1]   29  "h323-ivr-out=Tariff:Unknown"
00:18:27: RADIUS:  User-Name           [1]   12  "4081234567"
00:18:27: RADIUS:  Acct-Status-Type    [40]  6   Stop                     [2]
00:18:27: RADIUS:  NAS-Port-Type       [61]  6   Async                    [0]
00:18:27: RADIUS:  Vendor, Cisco       [26]  19
00:18:27: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
00:18:27: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:18:27: RADIUS:  Called-Station-Id   [30]  7   "12345"
00:18:27: RADIUS:  Service-Type        [6]   6   Login                    [1]
00:18:27: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:18:27: RADIUS:  Delay-Time          [41]  6   0
00:18:27: RADIUS: Received from id 5 10.6.20.70:1699, Accounting-response, len 20
00:18:27: RADIUS:  authenticator E5 B1 ED 3B AD A8 5B 5C - 49 83 63 BA DF 02 B2 00
```

An incoming call is set up using dial-peer voice 1001 pots. dial-peer voice 1001 has voice-class aaa 2 applied which should redirect AAA requests to the server specified for method list sanj_aaa6: 10.6.20.70 auth-port 1708 acct-port 1709.

```
00:30:05: ISDN Se1:23: RX <-  SETUP pd = 8  callref = 0x0004
00:30:05:         Bearer Capability i = 0x8090A2
00:30:05:         Channel ID i = 0xE1808397
00:30:05:         Calling Party Number i = 0x0080, '4081234567', Plan:Unknown,
Type:Unknown
00:30:05:         Called Party Number i = 0xE9, '12346', Plan:Private, Type:Abbreviated
00:30:05: RADIUS/ENCODE(0000000E): Unsupported AAA attribute timezone
00:30:05: RADIUS(0000000E): Encoding nas-port...Only port-type avlbl
00:30:05: RADIUS(0000000E): sending
00:30:05: RADIUS: Send to unknown id 6 10.6.20.70:1709, Accounting-Request, len 262
00:30:05: RADIUS:  authenticator 2F 3A 09 3D 6B C4 10 D2 - F6 68 D6 F4 36 35 C3 DE
00:30:05: RADIUS:  Acct-Session-Id     [44]  10  "0000000E"
00:30:05: RADIUS:  Vendor, Cisco       [26]  56
00:30:05: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=6C29BC16 BF1911D3 8010E483
89ADC43B"
00:30:05: RADIUS:  Vendor, Cisco       [26]  31
00:30:05: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
00:30:05: RADIUS:  Vendor, Cisco       [26]  65
00:30:05: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=6C29BC16 BF1911D3
8010E483 89ADC43B"
```

```
00:30:05: RADIUS:  User-Name           [1]   12  "4081234567"
00:30:05: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
00:30:05: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:30:05: RADIUS:  Vendor, Cisco       [26]  19
00:30:05: RADIUS:   cisco-nas-port     [2]   13  "ISDN 1:D:23"
00:30:05: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:30:05: RADIUS:  Called-Station-Id   [30]  7   "12346"
00:30:05: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:30:05: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:30:05: RADIUS:  Delay-Time          [41]  6   0
00:30:05: ISDN Se1:23: TX ->  CALL_PROC pd = 8  callref = 0x8004
00:30:05:         Channel ID i = 0xA98397
00:30:05: ISDN Se1:23: TX ->  CONNECT pd = 8  callref = 0x8004
00:30:05: ISDN Se1:23: RX <-  CONNECT_ACK pd = 8  callref = 0x0004
00:30:05: ISDN Se1:23: CALL_PROGRESS: CALL_CONNECTED call id 0x64, bchan 22, dsl 1
00:30:05: %ISDN-6-CONNECT: Interface Serial1:22 is now connected to 4081234567
00:30:06: RADIUS: Received from id 6 10.6.20.70:1709, Accounting-response, len 20
00:30:06: RADIUS:  authenticator E1 AD 70 9F DC 09 29 32 - 74 47 96 9F 3F 77 27 82
00:30:11: %ISDN-6-CONNECT: Interface Serial1:22 is now connected to 4081234567
00:30:19: RADIUS(0000000E): Encoding nas-port...Only port-type avlbl
00:30:19: RADIUS/ENCODE(0000000E): acct_session_id: 14
00:30:19: RADIUS(0000000E): sending
00:30:19: RADIUS: Send to unknown id 4 10.6.20.70:1708, Access-Request, len 199
00:30:19: RADIUS:  authenticator CE 16 21 8D A5 59 56 9F - B7 E9 CA 5C EC C5 89 A0
00:30:19: RADIUS:  User-Name           [1]   8   "777777"
00:30:19: RADIUS:  User-Password       [2]   18  *
00:30:19: RADIUS:  Vendor, Cisco       [26]  56
00:30:19: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=61A46F2C 00000003 62E66E40
634A0A64"
00:30:19: RADIUS:  Vendor, Cisco       [26]  36
00:30:19: RADIUS:   Cisco AVpair       [1]   30  "h323-ivr-out=transactionID:4"
00:30:19: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:30:19: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:30:19: RADIUS:  Vendor, Cisco       [26]  19
00:30:19: RADIUS:   cisco-nas-port     [2]   13  "ISDN 1:D:23"
00:30:19: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:30:19: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:30:19: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:30:20: RADIUS: Received from id 4 10.6.20.70:1708, Access-Accept, len 173
00:30:20: RADIUS:  authenticator FF 0D 40 72 0D 80 12 26 - 44 13 D5 0E C4 BB 71 BE
00:30:20: RADIUS:  Vendor, Cisco       [26]  34
00:30:20: RADIUS:   Cisco AVpair       [1]   28  "h323-credit-amount=7777.77"
00:30:20: RADIUS:  Vendor, Cisco       [26]  26
00:30:20: RADIUS:   Cisco AVpair       [1]   20  "h323-return-code=0"
00:30:20: RADIUS:  Vendor, Cisco       [26]  30
00:30:20: RADIUS:   h323-credit-time   [102] 24  "h323-credit-time=54329"
00:30:20: RADIUS:  Vendor, Cisco       [26]  33
00:30:20: RADIUS:   h323-billing-model [109] 27  "h323-billing-model=prepay"
00:30:20: RADIUS:  Vendor, Cisco       [26]  24
00:30:20: RADIUS:   h323-currency      [110] 18  "h323-currency=US"
00:30:20: RADIUS:  Idle-Timeout        [28]  6   30
00:30:20: RADIUS: Received from id E
00:30:43: ISDN Se1:23: RX <-  DISCONNECT pd = 8  callref = 0x0004
00:30:43:         Cause i = 0x8290 - Normal call clearing
00:30:43: %ISDN-6-DISCONNECT: Interface Serial1:22  disconnected from 4081234567 , call
lasted 37 seconds
00:30:43: ISDN Se1:23: TX ->  RELEASE pd = 8  callref = 0x8004
00:30:43: ISDN Se1:23: RX <-  RELEASE_COMP pd = 8  callref = 0x0004
00:30:43: RADIUS/ENCODE(0000000E): Unsupported AAA attribute timezone
00:30:43: RADIUS(0000000E): Encoding nas-port...Only port-type avlbl
00:30:43: RADIUS(0000000E): sending
00:30:43: RADIUS: Send to unknown id 7 10.6.20.70:1709, Accounting-Request, len 327
00:30:43: RADIUS:  authenticator 99 5A B4 45 67 C0 F4 91 - 9B 4B C3 1D 7E DE 7D D1
00:30:43: RADIUS:  Acct-Session-Id     [44]  10  "0000000E"
```

```
00:30:43: RADIUS:  Vendor, Cisco       [26]  56
00:30:43: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=6C29BC16 BF1911D3 8010E483
89ADC43B"
00:30:43: RADIUS:  Vendor, Cisco       [26]  31
00:30:43: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
00:30:43: RADIUS:  Vendor, Cisco       [26]  65
00:30:43: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=6C29BC16 BF1911D3
8010E483 89ADC43B"
00:30:43: RADIUS:  Acct-Input-Octets   [42]  6   0
00:30:43: RADIUS:  Acct-Output-Octets  [43]  6   161920
00:30:43: RADIUS:  Acct-Input-Packets  [47]  6   0
00:30:43: RADIUS:  Acct-Output-Packets [48]  6   1012
00:30:43: RADIUS:  Acct-Session-Time   [46]  6   37
00:30:43: RADIUS:  Vendor, Cisco       [26]  35
00:30:43: RADIUS:   Cisco AVpair       [1]   29  "h323-ivr-out=Tariff:Unknown"
00:30:43: RADIUS:  User-Name           [1]   12  "4081234567"
00:30:43: RADIUS:  Acct-Status-Type    [40]  6   Stop                      [2]
00:30:43: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:30:43: RADIUS:  Vendor, Cisco       [26]  19
00:30:43: RADIUS:   cisco-nas-port     [2]   13  "ISDN 1:D:23"
00:30:43: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:30:43: RADIUS:  Called-Station-Id   [30]  7   "12346"
00:30:43: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:30:43: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:30:43: RADIUS:  Delay-Time          [41]  6   0
00:30:43: RADIUS: Received from id 7 10.6.20.70:1709, Accounting-response, len 20
00:30:43: RADIUS:  authenticator 78 80 AB D1 82 75 ED ED - E4 1F 12 25 D8 83 F9 6
```

!**voice class aaa 3** is applied to **dial-peer voice 1000 pots** and a call is made. **voice class aaa 3** uses server 10.6.20.70 with auth port 1720 and acct port 1721. The radius daemon has not started. AAA accounting and AAA authorization requests are sent to the appropriate server but no acknowledgement is recieved. Retries are attempted.

```
00:37:03: %SYS-5-CONFIG_I: Configured from console by console
00:37:11: ISDN Se0:23: RX <-  SETUP pd = 8  callref = 0x009E
00:37:11:         Bearer Capability i = 0x8090A2
00:37:11:         Channel ID i = 0xE1808397
00:37:11:         Calling Party Number i = 0x0080, '4081234567', Plan:Unknown,
Type:Unknown
00:37:11:         Called Party Number i = 0xE9, '12345', Plan:Private, Type:Abbreviated
00:37:11: RADIUS/ENCODE(00000010): Unsupported AAA attribute timezone
00:37:11: RADIUS(00000010): Encoding nas-port...Only port-type avlbl
00:37:11: RADIUS(00000010): sending
00:37:11: RADIUS: Send to unknown id 8 10.6.20.70:1721, Accounting-Request, len 414
00:37:11: RADIUS:  authenticator EC F7 FD AB ED 0D 26 BF - F0 A4 D2 88 91 1E D9 22
00:37:11: RADIUS:  Acct-Session-Id     [44]  10  "00000010"
00:37:11: RADIUS:  Vendor, Cisco       [26]  56
00:37:11: RADIUS:   h323-setup-time    [25]  50  "h323-setup-time=*00:37:09.095 UTC Sat
Jan 1 2000"
00:37:11: RADIUS:  Vendor, Cisco       [26]  34
00:37:11: RADIUS:   h323-gw-id         [33]  28  "h323-gw-id=router."
00:37:11: RADIUS:  Vendor, Cisco       [26]  56
00:37:11: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=69EAABEB BF1A11D3 8014E483
89ADC43B"
00:37:11: RADIUS:  Vendor, Cisco       [26]  31
00:37:11: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
00:37:11: RADIUS:  Vendor, Cisco       [26]  32
00:37:11: RADIUS:   h323-call-type     [27]  26  "h323-call-type=Telephony"
00:37:11: RADIUS:  Vendor, Cisco       [26]  65
00:37:11: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=69EAABEB BF1A11D3
8014E483 89ADC43B"
```

```
00:37:11: RADIUS:  Vendor, Cisco       [26]  30
00:37:11: RADIUS:   Cisco AVpair       [1]   24  "subscriber=RegularLine"
00:37:11: RADIUS:  User-Name           [1]   12  "4081234567"
00:37:11: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
00:37:11: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:37:11: RADIUS:  Vendor, Cisco       [26]  19
00:37:11: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
00:37:11: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:37:11: RADIUS:  Called-Station-Id   [30]  7   "12345"
00:37:11: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:37:11: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:37:11: RADIUS:  Delay-Time          [41]  6   0
00:37:11: ISDN Se0:23: TX ->  CALL_PROC pd = 8  callref = 0x809E
00:37:11:         Channel ID i = 0xA98397
00:37:11: ISDN Se0:23: TX ->  CONNECT pd = 8  callref = 0x809E
00:37:11: ISDN Se0:23: RX <-  CONNECT_ACK pd = 8  callref = 0x009E
00:37:11: ISDN Se0:23: CALL_PROGRESS: CALL_CONNECTED call id 0x65, bchan 22, dsl 0
00:37:11: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4081234567
00:37:16: RADIUS: Retransmit id 8
00:37:16: RADIUS: acct-delay-time for 4021D9EC (at 4021DB84) now 5
00:37:17: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4081234567
00:37:21: RADIUS: Retransmit id 1
00:37:21: RADIUS: acct-delay-time for 4021D9EC (at 4021DB84) now 10
00:37:26: RADIUS: Retransmit id 2
00:37:26: RADIUS: acct-delay-time for 4021D9EC (at 4021DB84) now 15
00:37:31: RADIUS: Tried all servers.
00:37:31: RADIUS: No valid server found. Trying any viable server
00:37:31: RADIUS: Tried all servers.
00:37:31: RADIUS: No response for id 3
00:37:31: RADIUS/DECODE: parse response no app start; FAIL
00:37:31: RADIUS/DECODE: parse response; FAIL
00:37:35: RADIUS(00000010): Encoding nas-port...Only port-type avlbl
00:37:35: RADIUS/ENCODE(00000010): acct_session_id: 16
00:37:35: RADIUS(00000010): sending
00:37:35: RADIUS: Send to unknown id 5 10.6.20.70:1720, Access-Request, len 199
00:37:35: RADIUS:  authenticator 4B 6E 67 9F D4 1E 73 37 - 45 D3 CD 7C 70 FD C7 12
00:37:35: RADIUS:  User-Name           [1]   8   "777777"
00:37:35: RADIUS:  User-Password       [2]   18  *
00:37:35: RADIUS:  Vendor, Cisco       [26]  56
00:37:35: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=61A46F2C 00000003 62E66E40
634A0A64"
00:37:35: RADIUS:  Vendor, Cisco       [26]  36
00:37:35: RADIUS:   Cisco AVpair       [1]   30  "h323-ivr-out=transactionID:5"
00:37:35: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:37:35: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
00:37:35: RADIUS:  Vendor, Cisco       [26]  19
00:37:35: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
00:37:35: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:37:35: RADIUS:  Service-Type        [6]   6   Login                     [1]
00:37:35: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:37:40: RADIUS: Retransmit id 5
00:37:45: RADIUS: Retransmit id 5
00:37:50: RADIUS: Retransmit id 5
00:37:55: RADIUS: Tried all servers.
00:37:55: RADIUS: No valid server found. Trying any viable server
00:37:55: RADIUS: Tried all servers.
00:37:55: RADIUS: No response for id 5
00:37:55: RADIUS/DECODE: parse response no app start; FAIL
00:37:55: RADIUS/DECODE: parse response; FAIL
00:38:00: %ISDN-6-DISCONNECT: Interface Serial0:22  disconnected from 4081234567 , call
lasted 48 seconds
00:38:00: ISDN Se0:23: TX ->  DISCONNECT pd = 8  callref = 0x809E
00:38:00:         Cause i = 0x8090 - Normal call clearing
00:38:00: RADIUS/ENCODE(00000010): Unsupported AAA attribute timezone
```

```
00:38:00: RADIUS(00000010): Encoding nas-port...Only port-type avlbl
00:38:00: RADIUS(00000010): sending
00:38:00: RADIUS: Send to unknown id 9 10.6.20.70:1721, Accounting-Request, len 660
00:38:00: RADIUS:  authenticator C5 79 B7 D3 92 75 37 D0 - E7 5C 5B 84 99 6E 97 17
00:38:00: RADIUS:  Acct-Session-Id     [44]  10  "00000010"
00:38:00: RADIUS:  Vendor, Cisco       [26]  56
00:38:00: RADIUS:   h323-setup-time    [25]  50  "h323-setup-time=*00:37:09.095 UTC Sat
Jan 1 2000"
00:38:00: RADIUS:  Vendor, Cisco       [26]  34
00:38:00: RADIUS:   h323-gw-id         [33]  28  "h323-gw-id=router."
00:38:00: RADIUS:  Vendor, Cisco       [26]  56
00:38:00: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=69EAABEB BF1A11D3 8014E483
89ADC43B"
00:38:00: RADIUS:  Vendor, Cisco       [26]  31
00:38:00: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
00:38:00: RADIUS:  Vendor, Cisco       [26]  32
00:38:00: RADIUS:   h323-call-type     [27]  26  "h323-call-type=Telephony"
00:38:00: RADIUS:  Vendor, Cisco       [26]  65
00:38:00: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=69EAABEB BF1A11D3
8014E483 89ADC43B"
00:38:00: RADIUS:  Vendor, Cisco       [26]  30
00:38:00: RADIUS:   Cisco AVpair       [1]   24  "subscriber=RegularLine"
00:38:00: RADIUS:  Acct-Input-Octets   [42]  6   0
00:38:00: RADIUS:  Acct-Output-Octets  [43]  6   112160
00:38:00: RADIUS:  Acct-Input-Packets  [47]  6   0
00:38:00: RADIUS:  Acct-Output-Packets [48]  6   701
00:38:00: RADIUS:  Acct-Session-Time   [46]  6   49
00:38:00: RADIUS:  Vendor, Cisco       [26]  58
00:38:00: RADIUS:   h323-connect-time  [28]  52  "h323-connect-time=*00:37:09.109 UTC Sat
Jan 1 2000"
00:38:00: RADIUS:  Vendor, Cisco       [26]  61
00:38:00: RADIUS:   h323-disconnect-tim[29]  55  "h323-disconnect-time=*00:37:57.739 UTC
Sat Jan 1 2000"
00:38:00: RADIUS:  Vendor, Cisco       [26]  34
00:38:00: RADIUS:   h323-disconnect-cau[30]  28  "h323-disconnect-cause=10  "
00:38:00: RADIUS:  Vendor, Cisco       [26]  35
00:38:00: RADIUS:   Cisco AVpair       [1]   29  "h323-ivr-out=Tariff:Unknown"
00:38:00: RADIUS:  Vendor, Cisco       [26]  28
00:38:00: RADIUS:   h323-voice-quality [31]  22  "h323-voice-quality=0"
00:38:00: RADIUS:  User-Name           [1]   12  "4081234567"
00:38:00: RADIUS:  Acct-Status-Type    [40]  6   Stop                     [2]
00:38:00: RADIUS:  NAS-Port-Type       [61]  6   Async                    [0]
00:38:00: RADIUS:  Vendor, Cisco       [26]  19
00:38:00: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
00:38:00: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
00:38:00: RADIUS:  Called-Station-Id   [30]  7   "12345"
00:38:00: RADIUS:  Service-Type        [6]   6   Login                    [1]
00:38:00: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
00:38:00: RADIUS:  Delay-Time          [41]  6   0
00:38:00: ISDN Se0:23: RX <-  RELEASE pd = 8  callref = 0x009E
00:38:00: ISDN Se0:23: TX ->  RELEASE_COMP pd = 8  callref = 0x809E
00:38:05: RADIUS: Retransmit id 9
00:38:05: RADIUS: acct-delay-time for 4021D9EC (at 4021DC7A) now 5
00:38:10: RADIUS: Retransmit id 4
00:38:10: RADIUS: acct-delay-time for 4021D9EC (at 4021DC7A) now 10
00:38:15: RADIUS: Retransmit id 5
00:38:15: RADIUS: acct-delay-time for 4021D9EC (at 4021DC7A) now 15
00:38:20: RADIUS: Tried all servers.
00:38:20: RADIUS: No valid server found. Trying any viable server
00:38:20: RADIUS: Tried all servers.
00:38:20: RADIUS: No response for id 6
00:38:20: RADIUS/DECODE: parse response no app start; FAIL
00:38:20: RADIUS/DECODE: parse response; FAIL
```

debug radius accounting

In the output below, cdr1 includes h323-call-origin but does not include VSA h323-gw-id. cdr2 includes h323-gw-id but does not include h323-call-origin.

```
show call accounting-template voice cdr1
CDR template cdr1 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
    The last load was successful.

    attr:  h323-call-origin (56)

 Totally 1 attrs defined.


show call accounting-template voice cdr2
CDR template cdr2 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr2.cdr
    The last load was successful.

    attr:  h323-gw-id (65)

 Totally 1 attrs defined.

!The output below is from a call that uses cdr1.cdr which allows only h323-call-origin.

debug radius accounting
Radius protocol debugging is on
Radius packet hex dump debugging is off
Radius packet protocol (authentication) debugging is off
Radius packet protocol (accounting) debugging is on
02:41:32: RADIUS/ENCODE(00000023): Unsupported AAA attribute timezone
02:41:32: RADIUS(00000023): Encoding nas-port...Only port-type avlbl
02:41:32: RADIUS(00000023): sending
02:41:32: RADIUS: Send to unknown id 26 10.6.20.70:1699, Accounting-Request, len
262
02:41:32: RADIUS:  authenticator 84 6E A0 C0 0F 27 79 03 - 59 96 FC 6C F4 17 05
4D
02:41:32: RADIUS:  Acct-Session-Id     [44]  10  "00000023"
02:41:32: RADIUS:  Vendor, Cisco       [26]  56
02:41:32: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=C925CD59 BF2B11D3
8038E483 89ADC43B"
02:41:32: RADIUS:  Vendor, Cisco       [26]  31
02:41:32: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
02:41:32: RADIUS:  Vendor, Cisco       [26]  65
02:41:32: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=C925CD59
BF2B11D3 8038E483 89ADC43B"
02:41:32: RADIUS:  User-Name           [1]   12  "4081234567"
02:41:32: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
02:41:32: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
02:41:32: RADIUS:  Vendor, Cisco       [26]  19
02:41:32: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
02:41:32: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
02:41:32: RADIUS:  Called-Station-Id   [30]  7   "12345"
02:41:32: RADIUS:  Service-Type        [6]   6   Login                     [1]
02:41:32: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
02:41:32: RADIUS:  Delay-Time          [41]  6   0
02:41:32: RADIUS: Received from id 26 10.6.20.70:1699, Accounting-response, len
20
02:41:32: RADIUS:  authenticator 90 AD C8 09 60 D7 26 01 - DE E0 BC DC C1 F8 CA
2F
02:41:32: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4081234567
```

```
02:41:38: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4081234567
02:41:52: RADIUS(00000023): Encoding nas-port...Only port-type avlbl
02:41:59: %ISDN-6-DISCONNECT: Interface Serial0:22  disconnected from 4081234567
, call lasted 26 seconds
02:41:59: RADIUS/ENCODE(00000023): Unsupported AAA attribute timezone
02:41:59: RADIUS(00000023): Encoding nas-port...Only port-type avlbl
02:41:59: RADIUS(00000023): sending
02:41:59: RADIUS: Send to unknown id 27 10.6.20.70:1699, Accounting-Request, len
327
02:41:59: RADIUS:  authenticator 13 B7 10 EE 1C 55 7A D2 - 0F 4A A5 2F 1F 85 0E
3A
02:41:59: RADIUS:  Acct-Session-Id     [44]  10  "00000023"
02:41:59: RADIUS:  Vendor, Cisco       [26]  56
02:41:59: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=C925CD59 BF2B11D3
8038E483 89ADC43B"
02:41:59: RADIUS:  Vendor, Cisco       [26]  31
02:41:59: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
02:41:59: RADIUS:  Vendor, Cisco       [26]  65
02:41:59: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=C925CD59
BF2B11D3 8038E483 89ADC43B"
02:41:59: RADIUS:  Acct-Input-Octets   [42]  6   0
02:41:59: RADIUS:  Acct-Output-Octets  [43]  6   121600
02:41:59: RADIUS:  Acct-Input-Packets  [47]  6   0
02:41:59: RADIUS:  Acct-Output-Packets [48]  6   760
02:41:59: RADIUS:  Acct-Session-Time   [46]  6   27
02:41:59: RADIUS:  Vendor, Cisco       [26]  35
02:41:59: RADIUS:   Cisco AVpair       [1]   29  "h323-ivr-out=Tariff:Unknown"
02:41:59: RADIUS:  User-Name           [1]   12  "4081234567"
02:41:59: RADIUS:  Acct-Status-Type    [40]  6   Stop                     [2]
02:41:59: RADIUS:  NAS-Port-Type       [61]  6   Async                    [0]
02:41:59: RADIUS:  Vendor, Cisco       [26]  19
02:41:59: RADIUS:   cisco-nas-port     [2]   13  "ISDN 0:D:23"
02:41:59: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
02:41:59: RADIUS:  Called-Station-Id   [30]  7   "12345"
02:41:59: RADIUS:  Service-Type        [6]   6   Login                    [1]
02:41:59: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
02:41:59: RADIUS:  Delay-Time          [41]  6   0
02:41:59: RADIUS: Received from id 27 10.6.20.70:1699, Accounting-response, len
20
02:41:59: RADIUS:  authenticator 7F B2 88 3A 4A 96 05 C6 - D5 81 19 D8 25 3B 4D CB

!The output below is from the show debug command.
show debug
Radius protocol debugging is on
Radius packet protocol (accounting) debugging is on

!The output below is from a call that uses cdr2 which allows h323-gw-id, but does not
allow h323-call-origin.

ADIUS/ENCODE(00000025): Unsupported AAA attribute timezone
02:51:35: RADIUS(00000025): Encoding nas-port...Only port-type avlbl
02:51:35: RADIUS(00000025): sending
02:51:35: RADIUS: Send to unknown id 28 10.6.20.70:1709, Accounting-Request, len
265
02:51:35: RADIUS:  authenticator 15 F0 7E AB 75 07 10 70 - 5E 3C 54 78 09 18 83
E5
02:51:35: RADIUS:  Acct-Session-Id     [44]  10  "00000025"
02:51:35: RADIUS:  Vendor, Cisco       [26]  34
02:51:35: RADIUS:   h323-gw-id         [33]  28  "h323-gw-id=router."
02:51:35: RADIUS:  Vendor, Cisco       [26]  56
02:51:35: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=306F55DD BF2D11D3
803CE483 89ADC43B"
02:51:35: RADIUS:  Vendor, Cisco       [26]  65
02:51:35: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=306F55DD
```

```
BF2D11D3 803CE483 89ADC43B"
02:51:35: RADIUS:  User-Name           [1]   12  "4081234567"
02:51:35: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
02:51:35: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
02:51:35: RADIUS:  Vendor, Cisco       [26]  19
02:51:35: RADIUS:   cisco-nas-port     [2]   13  "ISDN 1:D:23"
02:51:35: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
02:51:35: RADIUS:  Called-Station-Id   [30]  7   "12346"
02:51:35: RADIUS:  Service-Type        [6]   6   Login                     [1]
02:51:35: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
02:51:35: RADIUS:  Delay-Time          [41]  6   0
02:51:35: %ISDN-6-CONNECT: Interface Serial1:22 is now connected to 4081234567
02:51:35: RADIUS: Received from id 28 10.6.20.70:1709, Accounting-response, len
20
02:51:35: RADIUS:  authenticator D3 D8 59 42 2B 48 96 8D - 5E 2E D8 61 9A 9D 0D
5F
02:51:41: %ISDN-6-CONNECT: Interface Serial1:22 is now connected to 4081234567
02:51:43: %ISDN-6-DISCONNECT: Interface Serial1:22  disconnected from 4081234567
, call lasted 8 seconds
02:51:43: RADIUS/ENCODE(00000025): Unsupported AAA attribute timezone
02:51:43: RADIUS(00000025): Encoding nas-port...Only port-type avlbl
02:51:43: RADIUS(00000025): sending
02:51:43: RADIUS: Send to unknown id 29 10.6.20.70:1709, Accounting-Request, len
330
02:51:43: RADIUS:  authenticator 55 35 AB CC 20 64 69 4B - 3F EE 79 04 11 E8 AE
4F
02:51:43: RADIUS:  Acct-Session-Id     [44]  10  "00000025"
02:51:43: RADIUS:  Vendor, Cisco       [26]  34
02:51:43: RADIUS:   h323-gw-id         [33]  28  "h323-gw-id=router."
02:51:43: RADIUS:  Vendor, Cisco       [26]  56
02:51:43: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=306F55DD BF2D11D3
803CE483 89ADC43B"
02:51:43: RADIUS:  Vendor, Cisco       [26]  65
02:51:43: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=306F55DD
BF2D11D3 803CE483 89ADC43B"
02:51:43: RADIUS:  Acct-Input-Octets   [42]  6   0
02:51:43: RADIUS:  Acct-Output-Octets  [43]  6   50240
02:51:43: RADIUS:  Acct-Input-Packets  [47]  6   0
02:51:43: RADIUS:  Acct-Output-Packets [48]  6   314
02:51:43: RADIUS:  Acct-Session-Time   [46]  6   8
02:51:43: RADIUS:  Vendor, Cisco       [26]  35
02:51:43: RADIUS:   Cisco AVpair       [1]   29  "h323-ivr-out=Tariff:Unknown"
02:51:43: RADIUS:  User-Name           [1]   12  "4081234567"
02:51:43: RADIUS:  Acct-Status-Type    [40]  6   Stop                      [2]
02:51:43: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
02:51:43: RADIUS:  Vendor, Cisco       [26]  19
02:51:43: RADIUS:   cisco-nas-port     [2]   13  "ISDN 1:D:23"
02:51:43: RADIUS:  Calling-Station-Id  [31]  12  "4081234567"
02:51:43: RADIUS:  Called-Station-Id   [30]  7   "12346"
02:51:43: RADIUS:  Service-Type        [6]   6   Login                     [1]
02:51:43: RADIUS:  NAS-IP-Address      [4]   6   10.5.20.100
02:51:43: RADIUS:  Delay-Time          [41]  6   0
02:51:43: RADIUS: Received from id 29 10.6.20.70:1709, Accounting-response, len
20
02:51:43: RADIUS:  authenticator 45 31 ED 45 F4 06 ED 54 - 5E 6F 83 64 4D 2D 34
90
```

Using show Commands for AAA Voice Troubleshooting

show call accounting voice summary

The **show call accounting voice summary** command shows the status of all accounting templates that are defined, loaded.

```
show call accounting voice summary
name         url                                        last_load  is_running
=========================================================================
cdr1         tftp://10.255.255.255/johndoe/sanjose/ success    is running
cdr2         tftp://10.255.255.255/johndoe/sanjose/ success    is running
```

show call accounting-template voice

The **show call accounting-template voice** *template name* command shows the VSAs that are contained in the accounting template.

```
show call accounting-template voice cdr1
CDR template cdr1 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
The last load was successful.

    attr:  h323-call-origin (56)

 Totally 1 attrs defined.

show call accounting-template voice cdr2
CDR template cdr2 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr2.cdr
    The last load was successful.

    attr:  h323-call-origin (56)

 Totally 1 attrs defined.


!The output below results from defining a template that does not exist or that cannot be
reached.

router(config)#$://10.255.255.255/johndoe/sanjose/cdr/cdr4000.cdr
Reading cdr template cdr10 fail, put it on retry queue.

01:15:46: hifs ifs could not open file

!The output below is for a template with an invalid VSA.
show call accounting-template voice cdr1
CDR template cdr1 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
    The last load was successful.

    attr:  h323-call-origin (56)

 Totally 1 attrs defined.

!Template cdr1.cdr is modified on the tftp server to enable an invalid VSA ( for example
h323-call-origin) to be put into the template.
```

```
call accounting-template voice reload cdr1
Loading johndoe/sanjose/cdr/cdr1.cdr from 10.255.255.255 (via Ethernet0): !
[OK - 88/4096 bytes]
cam: Fail to reload cdr template cdr1, unloading ...

02:27:29: hifs ifs file read succeeded. size=88,
url=tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
02:27:29: Error: attr name invalid-vsa-h323-call-origin (0) is not valid in line 3.

sh call accounting-template voice cdr1
CDR template cdr1 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
    Last load returned errno=8, Exec format error

    attr:  h323-call-origin (56)

 Totally 1 attrs defined.


The template has been rejected, and previous template still applied.
```

show call aaa attributes

The **show call aaa attributes** command displays the VSAs that are supported. Mandatory VSAs that are sent to the accounting server are not displayed.

```
show call aaa attributes
AAA ATTRIBUTE LIST:
    Name=disc-cause-ext                Format=Enum
    Name=Acct-Status-Type              Format=Enum
    Name=acl                           Format=Ulong
    Name=addr                          Format=IP v4 Address
    .....      .....
    Name=gw-password                   Format=Binary
    Name=h323-billing-model            Format=String
    Name=h323-call-origin              Format=String
    .....      .....

!Use the show call accounting-template voice summary command to check if a template is
loaded and running.
!The output below shows two templates successfully loaded and running, and a template that
failed to load.
show call accounting-template voice summary
name         url                                    last_load  is_running
=========================================================================
cdr1         tftp://10.255.255.255/johndoe/sanjose/ success    is running
cdr2         tftp://10.255.255.255/johndoe/sanjose/ success    is running
cdr10        tftp://10.255.255.255/johndoe/sanjose/ fail       is not running

!The output below shows reloading template cdr1 after modifying it.
!Initially, the original template cdr1 is loaded as shown:
show call accounting-template voice cdr1
CDR template cdr1 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
    The last load was successful.

    attr:  h323-call-origin (56)

 Totally 1 attrs defined.
```

```
!Additional VSAs are added to modify cdr1 on the tftp server as shown:

call accounting
call accounting-template voice reload cdr1
Loading johndoe/sanjose/cdr/cdr1.cdr from 10.255.255.255 (via Ethernet0): !
[OK - 1848/3072 bytes]

cam: Reload cdr template cdr1 success.
01:35:58: hifs ifs file read succeeded. size=1848,
url=tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
show call accounting-template voice cdr1
CDR template cdr1 is running
    url: tftp://10.255.255.255/johndoe/sanjose/cdr/cdr1.cdr
    The last load was successful.
    attr:  h323-call-origin (56)
    attr:  h323-call-type (57)
    attr:  h323-connect-time (59)
    attr:  h323-disconnect-cause (63)
    attr:  h323-disconnect-time (64)
    attr:  h323-gw-id (65)
    attr:  h323-remote-address (73)
    attr:  h323-remote-id (74)
    attr:  h323-setup-time (76)
    attr:  h323-voice-quality (78)
    attr:  subscriber (79)
    attr:  in-portgrp-id (80)
    attr:  out-portgrp-id (81)
    attr:  charged-units (82)
    attr:  disconnect-text (83)
    attr:  info-type (84)
    attr:  logical-if-index (85)
    attr:  peer-address (86)
    attr:  peer-id (87)
    attr:  peer-if-index (88)
    attr:  acom-level (89)
    attr:  tx-duration (90)
    attr:  voice-tx-duration (91)
    attr:  fax-tx-duration (92)
    attr:  noise-level (94)
    attr:  codec-bytes (95)
    attr:  coder-type-rate (96)
    attr:  early-packets (97)
    attr:  late-packets (98)
    attr:  lost-packets (99)
    attr:  gapfill-with-interpolation (100)
    attr:  gapfill-with-prediction (101)
    attr:  gapfill-with-redundancy (102)
    attr:  gapfill-with-silence (103)
    attr:  lowater-playout-delay (104)
    attr:  hiwater-playout-delay (105)
    attr:  ontime-rv-playout (106)
    attr:  receive-delay (107)
    attr:  round-trip-delay (108)
    attr:  remote-udp-port (109)
    attr:  session-protocol (110)
    attr:  vad-enable (111)

 Totally 42 attrs defined.
```

Accounting Server Connectivity Failure and Recovery Detection

The Accounting Server Connectivity Failure and Recovery Detection feature provides the scriptable option to reject new calls entering the VoIP network and tear down all existing calls upon detecting connectivity failure to the method list that is associated with the RADIUS-based accounting server(s).

Prerequisites for Accounting Server Connectivity Failure and Recovery Detection

The following tasks are prerequisites to configuring the features:

- Establish a working IP network. For more information about configuring IP, refer to the *Cisco IOS IP Configuration Guide*, Release 12.3.
- Configure VoIP. For more information about configuring VoIP, refer to the Voice Configuration Library.
- Configure a TFTP sever to perform storage and retrieval of the audio files, which are required by the Debit Card gateway or other features requiring Tool Command Language Interactive Voice Response (Tcl IVR) scripts and audio files.
- Program and configure the interface between the RADIUS server and the Cisco voice gateway to operate with VSAs.
- Create the accounting method list **default** that includes all RADIUS servers using the **aaa accounting connection default start-stop group radius** command in global configuration mode. This method list is required by the probe accounting records the Accounting Server Connectivity Failure and Recovery Detection feature uses to determine the state of connectivity to the method list.

Restrictions for Accounting Server Connectivity Failure and Recovery Detection

The Accounting Server Connectivity Failure and Recovery Detection feature is applicable only to the RADIUS accounting protocol. It is not applicable to any other protocols or servers, such as RADIUS access protocol, TACACS, or DIAMETER.

If both voice and dial calls need to be done on the same gateway, different accounting servers must be configured for each type of call.

Information About Accounting Server Connectivity Failure and Recovery Detection

To configure Accounting Server Connectivity Failure and Recovery Detection, you need to understand the following concept:

- Global Accounting Script, page 16

Global Accounting Script

The Accounting Server Connectivity Failure and Recovery Detection feature uses a configurable, legless Global Accounting Script (GAS) to control algorithms that determine the state of connectivity to the method list that is associated with the accounting server. The Accounting Server Connectivity Failure and Recovery Detection feature has two major functional components:

- Tcl legless GAS—Controls and drives the detection, recovery, and probe algorithms.
- Application Tcl scripts—Performs the call treatments to incoming calls and existing calls when notified that the method list is unreachable.

The GAS is a Tcl script with configurable parameters that users can customize for their own network requirements. Users can configure one GAS for each method list, or one GAS script for multiple method lists.

Because the RADIUS accounting protocol is User Datagram Protocol (UDP)-based, it is connectionless, and there is no guaranteed connectivity with the RADIUS server. The Accounting Server Connectivity Failure and Recovery Detection feature uses the acknowledgment of accounting requests from the method list to detect connectivity.

The GAS determines the state transition of the method list and updates the AAA system with the latest method list status. If the method list is unreachable, the AAA system locates all the active calls associated with the unreachable method list and informs the application script instances of the server unreachable event. The application script applies the appropriate treatments for this new event to the existing calls. For incoming calls, the application script checks the method list status and applies the appropriate treatment. For example, the application can clear the existing calls and reject new incoming calls for this method list. When the method list becomes reachable, the application script instances are notified, and they can take the appropriate action.

How to Configure Accounting Server Connectivity Failure and Recovery Detection

This section contains procedures identified as either required or optional.

- Configuring the GAS, page 17 (optional)
- Loading the GAS, page 18 (required)
- Starting the GAS, page 19 (optional)
- Starting the GAS in Global Configuration Mode, page 20 (optional)
- Verifying the GAS, page 20 (optional)
- Troubleshooting Accounting Server Connectivity Failure and Recovery Detection, page 21 (optional)

Configuring the GAS

Perform this task to configure the GAS.

How the GAS Application Verifies Configuration Parameters

The GAS application reads in the configured method list and uses it to identify the configuration parameters associated with that method list. Configuration parameters associated with a method list are either mandatory or optional.

If a mandatory configuration parameter does not exist, the GAS application displays the following message:

```
TCL GAS: >> Mandatory Parameter <parameter name from configuration avpair> does not exist
```

If a mandatory configuration parameter has an invalid type or value, the GAS application displays the following message:

```
TCL GAS:  >> Mandatory Parameter <exact parameter from configuration avpair> invalid value
```

At least one method list must be configured with the GAS application.

If the GAS application fails to read any of the mandatory configuration parameters, it fails and display this message:

```
TCL GAS:>>>> GasManager.Start  Exit Failure <<<<
```

SUMMARY STEPS

1. Download the GAS file.
2. Configure the GAS application and mandatory parameters.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Download the GAS file.	Download the GAS file at Technical Support Software Download for Cisco Tool Command Language Software at http://www.cisco.com/cgi-bin/tablebuild.pl/tclware. The GAS file contains the GAS and a ReadMe file.
Step 2	Configure the GAS application and mandatory parameters.	Refer to the ReadMe file for script-specific information about configuration parameters, including which are mandatory and which are optional. • You must configure at least the mandatory parameters before loading the GAS. • You must configure the GAS for at least one method list. • For information on the changes to the Tcl application programming interface (API) included in the Accounting Server Connectivity Failure and Recovery Detection feature, refer to the *TCL IVR Version 2.0 Programmer's Guide*. **Note** The GAS should be configured only by changing the value of its configuration parameters. The Technical Assistance Center (TAC) cannot support this feature, or the system on which it is loaded, if any changes are made to the script itself. Any changes to the GAS are supported by the Cisco Developer Support Program only, which requires a signed Developer Support Agreement. For more information, see the "Obtaining Technical Assistance" section on page xxvi.

Loading the GAS

Perform this task to load the GAS Tcl script.

Prerequisites

You must configure any script-specific parameters before loading the GAS.

SUMMARY STEPS

1. **enable**
2. **call application voice load** *application-name*

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	**`call application voice load`** *`application-name`* **Example:** `Router# call application voice load GAS`	Reloads the selected Tcl script from the URL.

Starting the GAS

You can start the GAS in privileged EXEC mode or global configuration mode. If it is started in privileged EXEC mode, it must be restarted every time the router is rebooted. If it is started in global configuration mode, it is saved in NVRAM and is started automatically when the router is rebooted.

Choose one of the following optional tasks to start the GAS.

- Starting the GAS in Privileged EXEC Mode, page 19 (optional)
- Starting the GAS in Global Configuration Mode, page 20 (optional)

Starting the GAS in Privileged EXEC Mode

Perform this task to start the GAS in privileged EXEC mode:

SUMMARY STEPS

1. **enable**
2. **call application session start** *instance-name application-name*

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	**`call application session start`** *`instance-name application-name`* **Example:** `Router# call application session start session_1 GAS`	Starts an instance of an application.

Starting the GAS in Global Configuration Mode

Perform this task to start the GAS in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call application session start** *instance-name application-name*

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

Step 3	`call application session start` *instance-name application-name* **Example:** `Router(config)# call application session start session_1 GAS`	Starts a new instance (session) of a Tcl IVR 2.0 application.

Verifying the GAS

Perform this task to verify that the GAS has been configured and is working correctly.

SUMMARY STEPS

1. **enable**
2. **show running-config**
3. **show voice accounting method**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show running-config** **Example:** `Router# show running-config`	(Optional) Displays the contents of the currently running configuration file. • Use the **show running-config** command to verify that the GAS parameters have the correct values.
Step 3	**show voice accounting method** [*method-list-name*] **Example:** `Router# show voice accounting method`	(Optional) Displays connectivity status information for a specified accounting method list or all the accounting method lists. • Use the *method-list-name* argument to specify a single method list, or omit this argument to display information for all method lists.

Troubleshooting Accounting Server Connectivity Failure and Recovery Detection

The Accounting Server Connectivity Failure and Recovery Detection feature uses a privileged EXEC mode command to enable diagnostic output concerning various events relating to gateway accounting Application Subscribe/Notify Layer (ASNL) to be displayed on a console. The **debug voice aaa asnl** command is intended only for troubleshooting purposes, because the volume of output generated by the software can result in severe performance degradation on the router.

The following procedure minimizes the load on the router created by the **debug voice aaa asnl** commands, because the console port is no longer generating character-by-character processor interrupts. If you cannot connect to a console directly, you can run this procedure via a terminal server. If you must break the Telnet connection, however, you may not be able to reconnect because the router may be unable to respond due to the processor load of generating the **debug voice aaa asnl** output.

Perform the following task to minimize the impact of using the **debug voice aaa asnl** command.

SUMMARY STEPS

1. Attach a console directly to a router running Cisco IOS Release 12.3(8)T or a later release.
2. **enable**
3. **configure terminal**
4. **no logging console**
5. Use Telnet to access a router port and repeat Steps 2 and 3.
6. **terminal monitor**
7. **end**
8. **debug voice aaa asnl**
9. **configure terminal**
10. **no terminal monitor**

11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	Attach a console directly to a router running Cisco IOS Release 12.2(8)T or a later release.	—
Step 2	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**no logging console** **Example:** `Router(config)# no logging console`	Disables all logging to the console terminal. • To reenable logging to the console, use the **logging console** command in global configuration mode.
Step 5	Use Telnet to access a router port and repeat Steps 2 and 3, then continue with step 6.	Enters global configuration mode in a recursive Telnet session, which allows the output to be redirected away from the console port.
Step 6	**terminal monitor** **Example:** `Router(config)# terminal monitor`	Enables logging output on the virtual terminal.
Step 7	**end** **Example:** `Router(config)# end`	Exits to privileged EXEC mode.
Step 8	**debug voice aaa asnl** **Example:** `Router# debug voice aaa asnl`	Displays debugging messages for gateway AAA ASNL. • Enter the **no debug voice aaa asnl** command when you are finished.
Step 9	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose

Step 10	`no terminal monitor`
	
	Example:
	`Router(config)# no terminal monitor`
		Disables logging on the virtual terminal.

Step 11	**end** **Example:** `Router(config)# end`	Exits to privileged EXEC mode.

Configuration Examples for Accounting Server Connectivity Failure and Recovery Detection

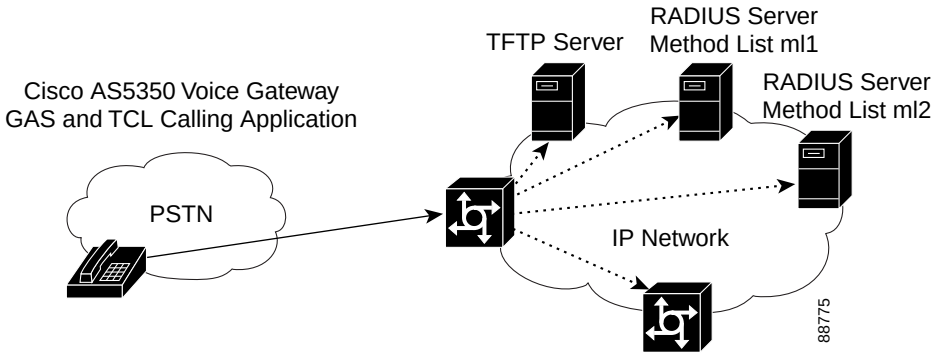
This section provides configuration examples to match the identified configuration tasks in the previous section:

- Configuring the GAS: Example, page 23
- Loading the GAS: Example, page 25
- Starting the GAS: Example, page 25
- Verifying the GAS: Example, page 25

Configuring the GAS: Example

Figure 46 shows the topology used in this example.

Figure 46 Example Topology



In this example, the GAS is configured for two method lists: ml1 and ml2.

```
call application voice GAS tftp://192.255.254.253/app_GAS.2.0.0.0.tcl
call application voice GAS method-list ml1;ml2
call application voice GAS gas-active-timer-ml1 30
call application voice GAS detect-failure-responses-ml1 2
call application voice GAS recovery-responses-ml1 2
call application voice GAS probe-retry-timer-ml1 5
call application voice GAS report-accounting-failed-ml1 false
call application voice GAS probe-user-name-ml1 johndoe
call application voice GAS acct-inactivity-period-ml1 120
```

```
call application voice GAS send-accounting-on-ml1 true
call application voice GAS use-gas-debugs-ml1 true

call application voice GAS gas-active-timer-ml2 30
call application voice GAS detect-failure-responses-ml2 2
call application voice GAS recovery-responses-ml2 5
call application voice GAS probe-retry-timer-ml2 10
call application voice GAS report-accounting-failed-ml2 true
call application voice GAS acct-inactivity-period-ml2 90
call application voice GAS send-accounting-on-ml2 false
call application voice GAS use-gas-debugs-ml2 false
```

The ml1 method list is configured with the following parameter values:

Parameter Value	Description
gas-active-timer = 30	Generate a syslog message indicating that the GAS application is active every 30 minutes.
detect-failure-responses-ml1 = 2	After receiving two consecutive failed responses, the method list is declared unreachable.
recovery-responses-ml1 = 2	After receiving two consecutive success responses, the method list is declared reachable again.
probe-retry-timer-ml1 = 5	When the method list is in an unreachable state, send probe accounting records every 5 seconds.
report-accounting-failed-ml1 = false	The application script should not be notified when accounting fails before the method list is marked as unreachable.
probe-user-name-ml1 = johndoe	Use johndoe as the user name field in the probe accounting records.
acct-inactivity-period-ml1 = 120	When the method list is in a reachable state and 120 seconds of inactivity have passed after the last accounting record was sent to the method list, a probe accounting record is sent to determine connectivity to the method list.
send-accounting-on-ml1 = true	A message is sent when the method list transitions from unreachable to reachable. This behavior should be synchronized with the calling application script. If the application script tears down the existing calls when it is notified that the method list status transitioned from reachable to unreachable, this parameter should be set to true; otherwise, it should be set to false.
use-gas-debugs-ml1 = true	Debug output is controlled by the **debug voip ivr** commands and displayed to the terminal. If the use-gas-debugs parameter was set to false, only the value of the use-gas-debugs parameters for the ml1 and ml2 method lists would be displayed. The default value is true.

Loading the GAS: Example

In the following example, the GAS named GAS is loaded:

```
enable
call application voice load GAS
```

Starting the GAS: Example

In the following example, the GAS named GAS is started in instance session_1:

```
enable
call application session start session_1 GAS
```

Verifying the GAS: Example

In the following examples, the output is displayed for each command in the task.

Sample Output from the show running-config Command

In the following example, method lists ml1 and ml2 are defined, and the GAS named GAS is configured:

```
Router# show running-config

Current configuration :4419 bytes
!
version 12.2
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname as5300-2
!
logging buffered warnings
enable password cisco
!
resource-pool disable
clock timezone gmt 15 40
!
aaa new-model
!
aaa group server radius ml1
 server 10.8.159.105 auth-port 1645 acct-port 1646
aaa group server radius ml2
 server 10.9.57.101 auth-port 1715 acct-port 1716
!
aaa accounting update newinfo
aaa accounting connection default start-stop group radius
aaa accounting connection h323 start-stop group radius
aaa accounting connection ml1 start-stop group ml1
aaa accounting connection ml2 start-stop group ml2

aaa session-id common
ip subnet-zero
!
isdn switch-type primary-5ess
!
mta receive maximum-recipients 0
no memory check-interval
!
!
```

```
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
 clock source line secondary 1
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 2
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 3
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
gw-accounting aaa
 method ml1
!
interface Loopback1
 no ip address
 no ip route-cache
 no ip mroute-cache
!
interface Ethernet0
 ip address 10.8.156.2 255.255.0.0
!
!
interface Serial0:23
 no ip address
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 fair-queue 64 256 0
 no cdp enable
!
interface Serial1:23
 no ip address
 isdn switch-type primary-5ess
 no cdp enable
!
interface Serial2:23
 no ip address
 isdn switch-type primary-5ess
 no cdp enable
!
interface Serial3:23
 no ip address
 isdn switch-type primary-5ess
 no cdp enable
!
interface FastEthernet0
 ip address 172.19.141.84 255.255.0.0
 ip directed-broadcast
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
```

```
!
ip default-gateway 10.8.0.1
ip classless
ip route 10.7.0.0 255.255.0.0 10.8.0.1
ip route 10.8.0.1 255.255.255.255 Ethernet0
ip route 10.9.0.0 255.255.0.0 10.8.0.1
ip route 192.255.254.253 255.255.255.255 10.8.0.1
ip route 192.255.254.254 255.255.255.255 10.8.0.1
no ip http server
!
access-list 101 permit ip any any
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
radius-server host 10.8.159.105 auth-port 1645 acct-port 1646
radius-server host 10.9.57.101 auth-port 1715 acct-port 1716
radius-server key cisco
radius-server authorization permit missing Service-Type
radius-server vsa send accounting
radius-server vsa send authentication
call rsvp-sync
!
call application voice GAS tftp://192.255.254.253/app_GAS.2.0.0.0.tcl
call application voice GAS method-list ml1;ml2
call application voice GAS gas-active-timer-ml1 5
call application voice GAS detect-failure-responses-ml1 2
call application voice GAS recovery-responses-ml1 2
call application voice GAS probe-retry-timer-ml1 5
call application voice GAS report-accounting-failed-ml1 false
call application voice GAS probe-user-name-ml1 johndoe
call application voice GAS acct-inactivity-period-ml1 120
call application voice GAS send-accounting-on-ml1 true
call application voice GAS use-gas-debugs-ml1 true
!
call application voice GAS gas-active-timer-ml2 5
call application voice GAS detect-failure-responses-ml2 2
call application voice GAS recovery-responses-ml2 5
call application voice GAS probe-retry-timer-ml2 10
call application voice GAS report-accounting-failed-ml2 true
call application voice GAS acct-inactivity-period-ml2 90
call application voice GAS send-accounting-on-ml2 false
call application voice GAS use-gas-debugs-ml2 false
!
call application voice calling_app tftp://192.255.254.253/app_session_rw.tcl
!
call application session start G1 GAS
!
voice-port 0:D
!
voice-port 1:D
!
voice-port 2:D
!
voice-port 3:D
!
!
mgcp profile default
!
dial-peer cor custom
!
!
dial-peer voice 1 pots
 application calling_app
 incoming called-number 25170
```

```
 port 0:D
!
dial-peer voice 1800877 voip
 destination-pattern 1800877....
 session target ipv4:10.8.156.3
!
alias exec osperr debug voip sett err
alias exec h225 debug cch323 h225
alias exec h245 debyg cch323 h245
alias exec ivr debyg voip ivr
alias exec ccapi debub voip ccapi in
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0
 password lab
line vty 1 4
!
end
```

Sample Output from the show voice accounting method Command

The following example displays the status history for the ml1 method list:

```
Router# show voice accounting method

Accounting Method List [ml1]
=====================
Current Status:
---------------
unreachable               [21:52:39 gmt Dec 4 2002]
last record sent time     [23:14:59 gmt Dec 4 2002]
total probe sent out      [84]

Status History:
---------------
(2) unreachable           [21:52:39 gmt Dec 4 2002]
(1) reachable             [21:46:19 gmt Dec 4 2002]
                 SUCCESS                          FAILURE
Record  [Received   | Notified ] [Received   |  Notified  | Reported ]
Type    [from server| to client] [from server|  to client | to call  ]
------  [-----------|----------] [-----------|------------|----------]
START   [     0     |     0    ] [     0     |      0     |     0    ]
UPDATE  [     0     |     0    ] [     0     |      0     |     0    ]
STOP    [     0     |     0    ] [     84    |      84    |     0    ]
ACCT_ON [     0     |     0    ] [     0     |      0     |     0    ]
------  [-----------|----------] [-----------|------------|----------]
TOTAL   [     0     |     0    ] [     84    |      84    |     0    ]
```

Troubleshooting Enhanced Billing Support for SIP Gateways

To troubleshoot the Enhanced Billing Support for SIP Gateways feature, perform the following steps:

- Make sure that you can make a voice call.
- Use the **debug ccsip all** command to enable all SIP debugging capabilities, or use one of the following SIP debug commands:
  - **debug ccsip calls**

  - **debug ccsip error**
  - **debug ccsip events**
  - **debug ccsip messages**
- In addition, **debug ccsip events** and **debug ccsip all** include new output specific to the Enhanced Billing Support for SIP Gateways feature. The example shows how the Proxy-Authorization header is broken down into a decoded user name and password.

```
CCSIP SPI: SIP Call Events tracing is enabled
21:03:21: sippmh_parse_proxy_auth: Challenge is 'Basic'.
21:03:21: sippmh_parse_proxy_auth: Base64 user-pass string is
'MTIzNDU2Nzg5MDEyMzQ1Njou'.
21:03:21: sip_process_proxy_auth: Decoded user-pass string is '1234567890123456:.'.
21:03:21: sip_process_proxy_auth: Username is '1234567890123456'.
21:03:21: sip_process_proxy_auth: Pass is '.'.
21:03:21: sipSPIAddBillingInfoToCcb: sipCallId for billing records =
10872472-173611CC-81E9C73D-F836C2B6@172.18.192.19421:03:21: ****Adding to UAS Request
table
```

Troubleshooting Settlement

The following section is provided to assist in determining if your OSP network is set up correctly. The problems listed have been reported as the most common errors made when configuring settlement in a network. Each section describes a problem and a solution.

Settlement Database Not Set Up Properly

Problem

Calls are routed through a settlement server, but the originating gateway gets no response or a negative response.

Solution

Check with the settlement provider to make sure that the router is properly registered with that provider. Router registration with settlement provider is normally done outside of OSP.

Tcl IVR Script Not Called

Problem

Tcl IVR script is not used on the originating gateway or terminating gateway.

Solution

Configure a Tcl IVR script for the dial peer using the **application** *application name* command.

Note Tcl/IVR scripts are required for settlement, and classic IVR 1.0 does not support settlement.

  - Use the **show call application voice summary** to list all the available scripts on the router.
  - Default is classic SESSION application, which cannot do settlement.

- The fax_hop_on.tcl script does not work with settlement.

The following example shows the available scripts on the router.

```
router# show call application voice summary
name                description

session             Basic app to do DID, or supply dialtone.
fax_hop_on          Script to talk to a fax redialer
clid_authen         Authenticate with (ani, dnis)
clid_authen_collect Authenticate with (ani, dnis), collect if that fails
clid_authen_npw     Authenticate with (ani, NULL)
clid_authen_col_npw Authenticate with (ani, NULL), collect if that fails
clid_col_npw_3      Authenticate with (ani, NULL), and 3 tries collecting
clid_col_npw_npw    Authenticate with (ani, NULL) and 3 tries without pw
SESSION             Default system session application
```

No Destination Pattern Set

Problem

The originating gateway inbound POTS dial peer has no destination pattern set.

Solution

Because some PBX devices do not pass along the calling number in the setup message, the router uses the destination-pattern number or answer-address as an alternative, and a calling number is a required field for settlement.

No Session Target Settlement Set on Originating Gateway

Problem

The originating gateway outbound VoIP dial peer has no session target settlement.

The router could make successful calls, but not through a settlement server. The session target specification dictates how the router resolves the terminating gateway address for a particular called number.

Solution

Configure the **session target settlement** *provider-number* command.

No VoIP Inbound Dial Peer on Terminating Gateway

Problem

The terminating gateway has no VoIP inbound dial peer. Because the settlement token in the incoming setup message from the originating gateway cannot be validate, the terminating gateway rejects the call.

Solution

Create an inbound dial peer with the **session target settlement** command.

No Application Attribute on Terminating Gateway

Problem

The terminating gateway has an inbound dial peer configured, but with no **application** command. The default session application (SESSION) processes the call, but it does not support settlement.

Solution

The default session application (SESSION) does not support the settlement feature. Therefore, you must configure the **application** command in the inbound dial peer.

Terminating Gateway Not Synchronized with Settlement Server

Problem

The terminating gateway clock is not synchronized with the settlement server. The terminating gateway rejects the call because it is too soon or too late to use the settlement token in the incoming setup message.

Solution

Use the **ntp** or **clock set** command to synchronize the clocks between the terminating gateway and the settlement server.

Settlement Provider Not Running

Problem

The settlement provider on the originating gateway or terminating gateway is not running. No settlement transaction processing is allowed unless the provider is running.

Solution

Enable settlement using the **no shutdown** command in settlement configuration mode. Use the **show settlement** command to verify the provider status.

Router and Server Not Using SSL to Communicate

Problem

The router cannot use SSL to communicate with the server because the server URL should be “https,” not “http.”

Solution

Configure a secured URL using “https.”

Problem

The router cannot use SSL to communicate with the server because the certificates of the server or router were not properly obtained.

Solution

Check the certificate enrollment process for both the server and the router.

Multiple Dial Peers Have Random Order

Problem

The originating gateway has multiple dial peers for the same called number, and settlement is never used. The order for rotary dial peers is random unless a dial peer preference is specified. The dial peer with lower preference is chosen first.

Solution

Define dial peer preference by using the **preference** command.

H.323 Setup Connection Timeout

Problem

The originating gateway cannot successfully set up a call with the first terminating gateway that is returned from the OSP server. The problem occurs when a gateway attempts to set up the call with the terminating gateways in the order they are received. If for some reason the H.323 call setup is not successful, there is a 15-second timeout by default before the next terminating gateway on the list is contacted.

Solution

The H.323 call setup timeout can be tuned using the **h225 timeout** command.

For example:

```
voice class h323 1
 h225 timeout tcp etablish <value 0 to 30 seconds>

 dial-peer voice 919 voip
 application session
 destination-pattern 919555....
 voice-class codec 1
 voice-class h323 1
 session target settlement
```

Problem Isolation

To isolate problems with settlement, perform the following tasks:

- Check the originating gateway and terminating gateway configuration for dial peers, settlement providers, and certificates.
- Check the network between the originating gateway, terminating gateway, and the server. Ping each device to make sure that the machines are running.
- Verify that IP calls can be made successfully. If so, the problem is specific to settlement.
- Turn on **debug voip ivr settlement** on the originating gateway to see if the Tcl IVR script initiates a settlement request to the server.

- Use the **debug voip settlement network** command on the originating gateway to capture the HTTP requests sent to the server and the response from the server. If the originating gateway gets no response from the server, contact the settlement provider.
- Turn on **debug voip settlement misc** to see the list of TOWs returned from the server. If this list is incorrect, contact the settlement provider.
- If the terminating gateway rejects the settlement token because it is too soon or too late to use it, synchronize the terminating gateway clock with the server.

Troubleshooting Fax

Troubleshooting Fax Applications

This chapter describes T.37 store and forward fax and T.38 fax gateway troubleshooting concepts. The applications are T.37 store and forward fax, T.38 fax relay for VoIP H.323, fax relay packet loss concealment, and T.37/T.38 fax gateways. The applications enable the Cisco gateways to send and receive faxes across packet-based networks, using modems or voice feature cards (VFCs).

Before beginning these troubleshooting procedures, check to ensure your fax machine and voice network are working. Try the following steps to help isolate the problem:

1. Plug the fax machine into a regular analog line and test the fax call while bypassing the voice network. If the problem persists, it could be the fax machine.
2. If the fax machine has a handset, connect the fax machine to the voice network, pick the handset up, and try to make a voice call. If the voice call works, you have verified that your voice network is operational. A voice call must be successful before a fax call can succeed.

This chapter contains the following topics:

For information about configuring fax features, refer to the *Cisco Fax Services over IP Application Guide*.

Fax Call Flow

The following fax call flows are described:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Fax Pass-Through and Fax Pass-Through with Upspeed

Fax pass-through is the simplest technique for sending fax over IP networks, but it is not the default, nor is it the most desirable method of supporting fax over IP. T.38 fax relay provides a more reliable and error-free method of sending faxes over an IP network, but some third-party H.323 and SIP implementations do not support T.38 fax relay. These same implementations often support fax pass-through.

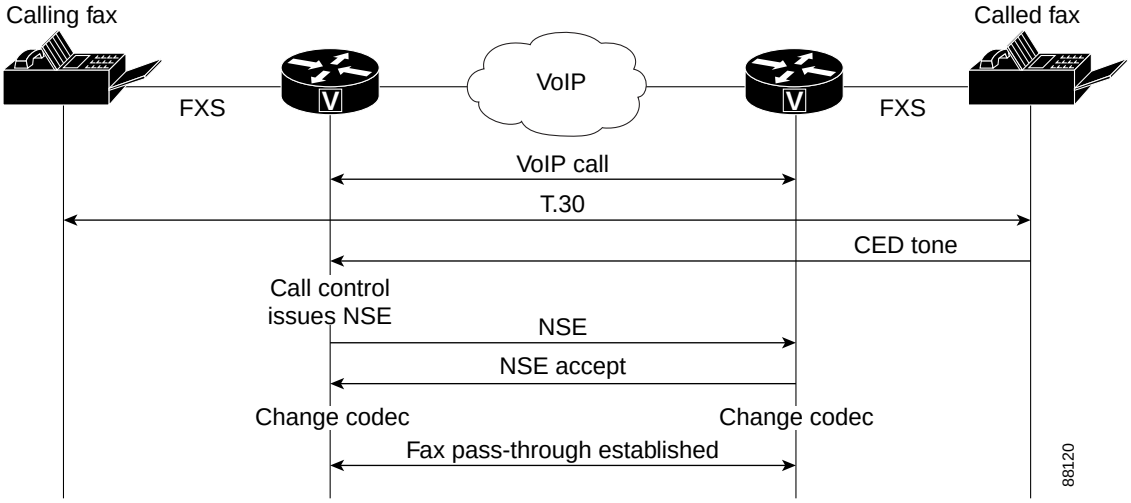
In fax pass-through mode, gateways do not distinguish between a fax call and a voice call. Fax communication between the two fax machines is carried in its entirety in-band over a voice call. Fax upspeed is similar to pass-through in the sense that the fax call is carried in-band over the voice call. The difference is that when you use upspeed, the gateways to some extent react to the fax call. Although relay mechanisms are not employed, with upspeed the gateways do recognize a CED fax tone and automatically change the voice codec to G.711 if necessary (thus the designation *upspeed*). Turn off echo cancellation (EC) and voice activity detection (VAD) for the duration of the call.

When a DSP is put into voice mode at the beginning of a VoIP call, the DSP is informed by the call control stack whether the control protocol can support pass-through or not. If pass-through is supported, the following events occur during a fax call:

1. For the duration of the call, the DSP listens for the 2100-Hz CED tone to detect a fax or modem on the line.
2. If the CED tone is heard, an internal event is generated to alert the call control stack that a fax or modem changeover is required.
3. The call control stack on the originating gateway instructs the DSP to send an NSE to the terminating gateway, informing the terminating gateway of the request to carry out a codec change.
4. If the terminating gateway supports NSEs, it responds to the originating gateway instruction and loads the new codec. The fax machines are able to communicate on an end-to-end basis with no further intervention by the voice gateways.

Fax pass-through call flow is shown in Figure 47.

Figure 47 Fax Pass-Through and Fax Upspeed Call Flow



Cisco Fax Relay

Cisco fax relay is the oldest method of supporting fax on Cisco IOS gateways and has been supported since Cisco IOS Release 11.3. Cisco fax relay uses Real-Time Transport Protocol (RTP) as the method of transport. In Cisco fax relay mode, gateways terminate T.30 fax signaling by spoofing a virtual fax machine to the locally attached fax machine. The gateways use a Cisco-proprietary fax-relay RTP-based protocol to communicate between them.

Cisco fax relay is the default operation and, in the absence of any explicit CLI on the dial peer, is used when a fax transmission is detected. If voice calls are being completed successfully between two routers, fax calls should also work. Events that occur during a Cisco fax relay call fall into the following call phases:

- Cisco Fax Relay Fax Setup Phase
- Cisco Fax Relay Data Transfer Phase

Cisco Fax Relay Fax Setup Phase

When a DSP is put into voice mode at the beginning of a VoIP call, the DSP is informed by the call control stack whether fax relay is supported and if it is supported, whether it is Cisco fax relay or T.38 fax relay. If Cisco fax relay is supported, the following events occur:

- Initially a VoIP call is established as if it were a normal speech call. Call control procedures are followed and the DSP is put into voice mode, after which human speech is expected to be received and processed.

Note If a fax answer or calling tone (CED or CNG) is heard at any time during the call, the DSP does not interfere with the speech processing. It allows the tone to continue across the VoIP call leg.

- A normal fax machine, after generating a CED or hearing a CNG, sends a DIS message with the capabilities of the fax machine.

  The DSP in the Cisco IOS gateway attached to the fax machine that generated the DIS message (normally the terminating gateway) detects the HDLC flag sequence at the start of the DIS message and initiates fax relay switchover.

  The DSP also triggers an internal event to notify the call control stack that fax switchover is required. The call control stack then instructs the DSP to change the RTP payload type to 96 and to send this payload type to the originating gateway.

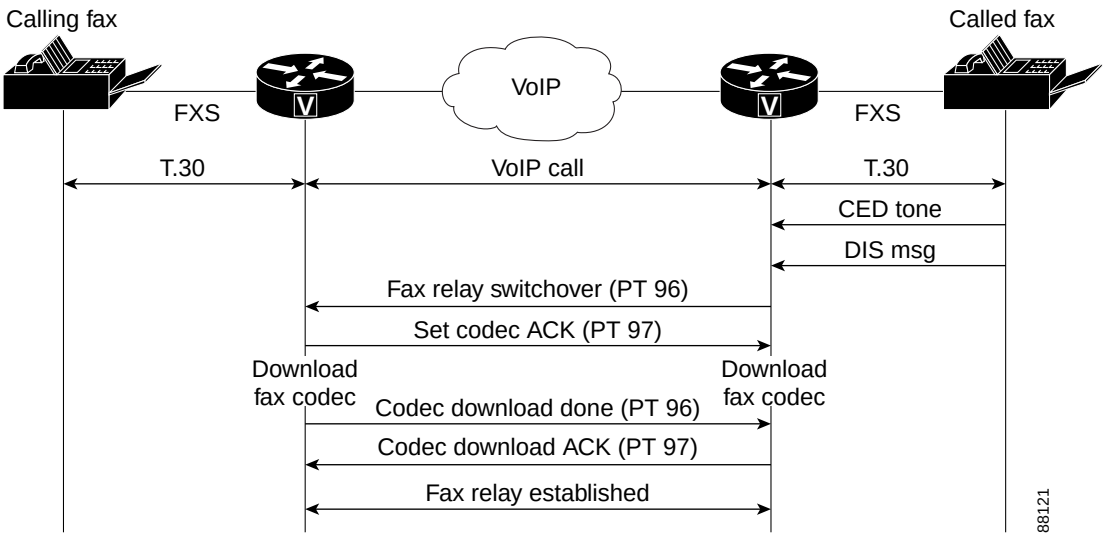
- When the DSP on the originating gateway receives an RTP packet with payload type set to 96, it triggers an event informing its own call control stack that a fax changeover has been requested by the remote gateway.

  The originating gateway then sends an RTP packet to the terminating gateway with payload type 97 to indicate that the originating gateway has started the fax changeover. When the terminating gateway receives the payload type 97 packet, the packet serves as an acknowledgement. The terminating gateway starts the fax codec download and is ready for fax relay.

- Once the originating gateway has completed the codec download, it sends RTP packets with payload type 96 to the terminating gateway. The terminating gateway responds with an RTP packet with payload type 97, and fax relay can begin between the two gateways. As part of the fax codec download, other parameters such as VAD, jitter buffers, and echo cancellation are changed to suit the different characteristics of a fax call.

Cisco fax relay fax setup is shown in Figure 48.

Figure 48 Cisco Fax Relay Fax Setup Call Flow



Cisco Fax Relay Data Transfer Phase

During fax relay operation, the T.30 analog fax signals are received from the PSTN or from a directly attached fax machine. The T.30 fax signals are demodulated by a DSP on the gateway and then packetized and sent across the VoIP network as data. The terminating gateway decodes the data stream and remodulates the T.30 analog fax signals to be sent to the PSTN or to a destination fax machine.

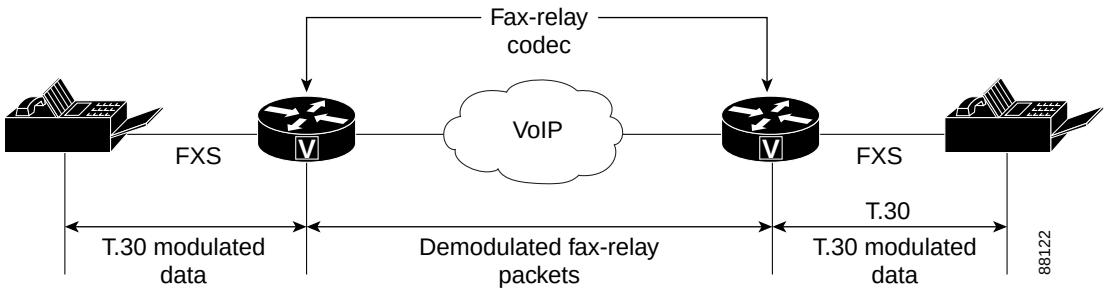
The messages that are demodulated and remodulated are predominantly the phase B, phase D, and phase E messages of a T.30 transaction. Most of the messages are passed across without any interference, but certain messages are modified according to the constraints of the VoIP network.

During phase B, fax machines interrogate each other's capabilities. They expect to communicate with each other across a 64-kbps PSTN circuit, and they attempt to make best use of the available bandwidth and circuit quality of a 64-kbps voice path. However, in a VoIP network, the fax machines do not have a 64-kbps PSTN circuit available. The bandwidth per call is probably less than 64 kbps, and the circuit is not considered a clear circuit.

Because transmission paths in VoIP networks are more limited than in the PSTN, Cisco IOS CLI is used to adjust fax settings on the VoIP dial peer. The adjusted fax settings restrict the facilities that are available to fax machines across the VoIP call leg and are also used to modify values in DIS and NSF messages that are received from fax machines.

The call flow of the Cisco fax relay data transfer phase is shown in Figure 49.

Figure 49 Cisco Fax Relay Data Transfer Call Flow



T.38 Fax Relay

T.38 provides an ITU-T standards-based method and protocols for fax relay. Data is packetized and encapsulated according to the T.38 standard. The encoding of the packet headers and the mechanism to switch from VoIP mode to fax relay mode are clearly defined in the specification. Annexes to the basic specification include details for operation under Session Initiation Protocol (SIP) and H.323 call control protocols.

T.38 uses data redundancy to compensate for packet loss. During T.38 call establishment, voice gateways indicate the level of packet redundancy that they incorporate in their transmission of facsimile User Datagram Packet Transport Layer packets (UDPTLs). The level of redundancy (the number of times that the packet is repeated) can be configured on Cisco IOS gateways.

There is work under way to implement T.38 fax switchover independently of the call control mechanisms. This is referred to as “bearer level signaling” and makes use of Named Signaling Events (NSEs).

The following sections address call-control-initiated switchover mechanisms:

- H.323 T.38 Fax Relay
- SIP T.38 Fax Relay
- MGCP T.38 Fax Relay

H.323 T.38 Fax Relay

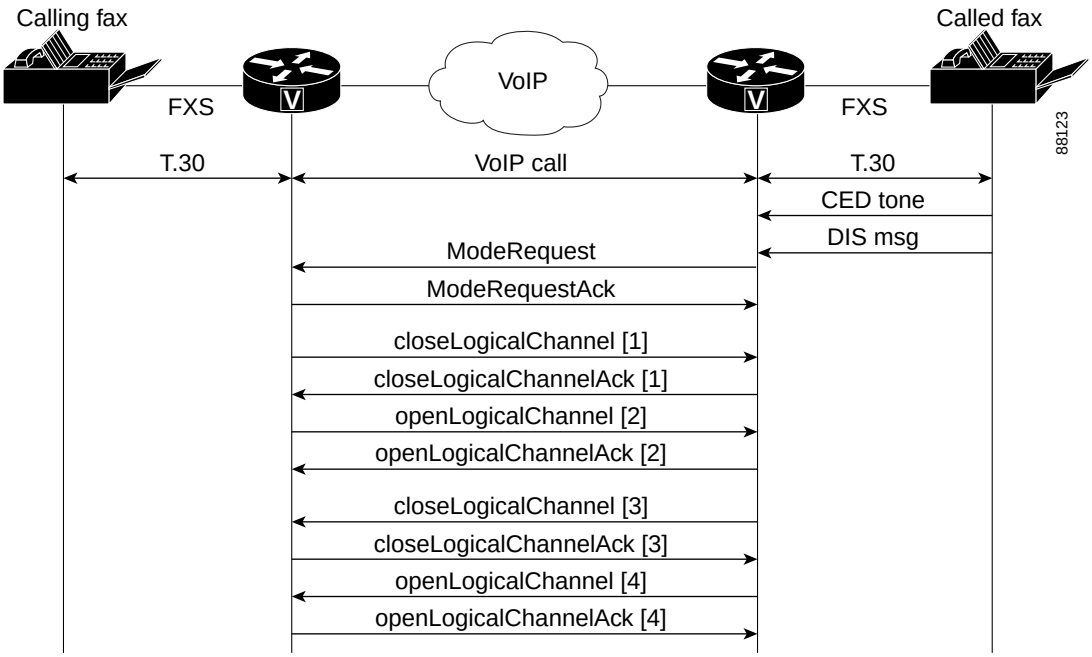
The T.38 Annex B standard defines the mechanism that is used to switch over from voice mode to T.38 fax mode during a call. The ability to support T.38 must be indicated during the initial VoIP call setup. If the DSP on the gateway is capable of supporting T.38 mode, this information is indicated during the H.245 negotiation procedures as part of the regular H.323 VoIP call setup.

Once the VoIP call setup is completed, the DSP continues to listen for a fax tone. When it hears one, the DSP signals the receipt of fax tone to the call control layer, which then initiates fax changeover as specified in the T.38 Annex B procedures. The H.245 message flow shown in Figure 50 contains the following events:

1. The detecting terminating gateway sends a ModeRequest message to the originating gateway, and the originating gateway responds with a ModeRequestAck.
2. The originating gateway sends a closeLogicalChannel message to close its VoIP UDP port, and the terminating gateway responds with a closeLogicalChannelAck while it closes the VoIP port.

3. The originating gateway sends an openLogicalChannel message that indicates to which port to send the T.38 UDP information on the originating gateway, and the terminating gateway responds with an openLogicalChannelAck.
4. The terminating gateway sends a closeLogicalChannel message to close its VoIP UDP port, and the originating gateway responds with a closeLogicalChannelAck.
5. Finally the terminating gateway sends an openLogicalChannel message that indicates to which port to send the T.38 UDP stream, and the originating gateway responds with an openLogicalChannelAck.
6. T.38-encoded UDP packets flow back and forth. At the end of the fax transmission, either gateway can initiate another ModeRequest message to return to VoIP mode.

Figure 50 H.323 T.38 Fax Relay Call Flow



SIP T.38 Fax Relay

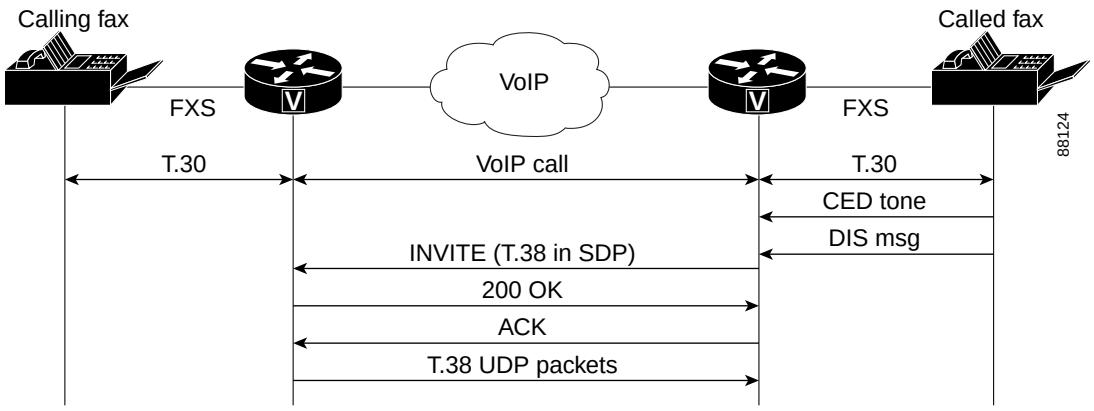
When the call control protocol is SIP, T.38 Annex D procedures are used for the changeover from VoIP to fax mode during a call. Initially, a normal VoIP call is established through the use of SIP INVITEs. The DSP needs to be informed that it can support T.38 mode while it is put into voice mode. During the call, when the DSP detects fax HDLC flags, it signals the detection of the flags to the call control layer, and the call control layer initiates a SIP INVITE mid-call to signal the desire to change the media stream.

The SIP T.38 fax relay call flow shown in Figure 51 contains the following events:

1. The terminating gateway detects a fax V.21 flag sequence and sends an INVITE with T.38 details in the SDP field to the originating gateway or to the SIP proxy server, depending on the network topology.
2. The originating gateway receives the INVITE message and sends back a 200 OK message.
3. The terminating gateway acknowledges the 200 OK message and sends an ACK message direct to the originating gateway.

4. The originating gateway starts sending T.38 UDP packets instead of VoIP UDP packets across the same ports.
5. At the end of the fax transmission, another INVITE message can be sent to return to VoIP mode.

Figure 51 SIP T.38 Fax Relay Call Flow



MGCP T.38 Fax Relay

MGCP T.38 fax relay conforms to ITU-T T.38, Procedures for Real-Time Group 3 Facsimile Communication over IP Networks, which determines procedures for real-time facsimile communication in various gateway control protocol (XGCP) applications.

MGCP-based T.38 fax relay has the following call flow:

1. A call is initially established as a voice call.
2. The gateways advertise capabilities in an SDP exchange during connection establishment.
3. If both gateways do not support T.38 fax relay, fax pass-through is used for fax transmission. If both gateways support T.38, they attempt to switch to T.38 upon fax tone detection. The existing audio channel is used for T.38 fax relay, and the existing connection port is reused to minimize delay. If failure occurs at some point during the switch to T.38, the call reverts to the original settings it had as a voice call. If this failure occurs, a fallback to fax pass-through is not supported.
4. Upon completion of the fax image transfer, the connection remains established and reverts to a voice call using the previously designated codec, unless the call agent instructs the gateways to do otherwise.

A fax relay MGCP event allows the gateway to notify the call agent of the status (start, stop, or failure) of T.38 processing for the connection. This event is sent in both call-agent-controlled and gateway-controlled mode.

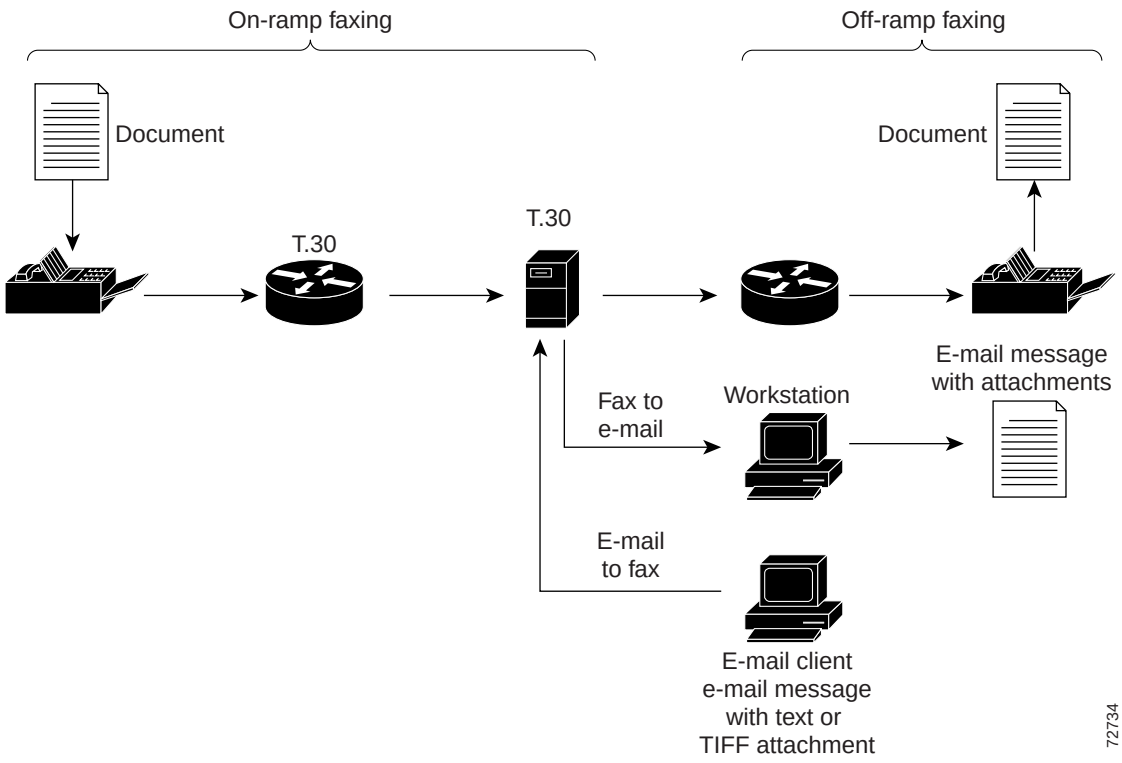
T.37 Store-and-Forward Fax

T.37 provides an ITU-T standards-based method for store-and-forward fax operation. The fax transmission process is divided into distinct sending and receiving phases with the potential to store the fax between sending and receiving, if necessary.

Cisco recommends that you dedicate a mail server to fax mail and avoid the conflicting configuration requirements of traditional e-mail and fax-mail servers. Optimize each mail server for its individual functions—for example, fax messages should usually retry transmissions every 5 minutes whereas normal e-mail should retry every 30 minutes, and fax messages should give up after 3 to 4 hours whereas normal e-mail should not give up for 4 to 5 days.

A topology for T.37 store-and-forward fax is shown in Figure 52.

Figure 52 T.37 Store-and-Forward Fax Topology



For more information about T.37 store and forward fax, refer to the "Configuring T.37 Store-and-Forward Fax" chapter in the *Cisco Fax Services over IP Application Guide*.

Fax Relay

To troubleshoot T.38 fax relay, perform the following steps:

- Make sure that you can make a voice call.
- Make sure that the desired fax protocol was set using the **fax protocol** command on both the originating and terminating gateways.
- Make sure that the fax protocol is configured as T.38 at the global configuration level or at the dial-peer configuration level for both the originating and terminating gateways.
- Use the **show call active** {**voice** | **fax**} command to display information for the active call table.
- Use the **show dial-peer voice** command to display configuration information for dial peers.

- For H.323 gateways, use the **debug cch323 all** command to enable all H.323 debugging capabilities, or use one of the following commands to debug problems while making the call:
  - **debug cch323 error**
  - **debug cch323 h225**
  - **debug cch323 h245**
  - **debug cch323 RAS**
  - **debug cch323 session**
  - **debug voip ccapi inout**
  - **debug vtsp session**
- For SIP gateways, use the **debug ccsip all** command to enable all SIP debugging capabilities, or use one of the following SIP debug commands:
  - **debug ccsip calls**
  - **debug ccsip error**
  - **debug ccsip events**
  - **debug ccsip info**
  - **debug ccsip media**
  - **debug ccsip messages**
  - **debug ccsip states**

For further details, see the following sections:

Identifying and Isolating the Problem

The first step to take when you troubleshoot any fax relay issue is to reduce the problem to its simplest form. Many issues arise in situations where multiple fax machines are not able to pass fax traffic. It is easiest to isolate two fax machines that are having problems and concentrate on a simple topology. Determine how these machines are connected to one another and resolve the issue between this pair first. In addition, you should draw a complete picture of the topology and determine how the fax machines are interconnected.

Troubleshooting one issue at a time minimizes confusion and allows for methodical troubleshooting. It is also possible that the solution for this problem resolves other fax relay problems in the network. Most fax relay problems result from poor voice configuration or network design. These lead to basic connectivity problems and physical line or packet loss and jitter problems.

After you have identified and isolated the problem, you need to verify the basic voice configuration and monitor the health of the network.

Checking Basic Connectivity

Basic fax connectivity issues can be the result of the following factors:

- Normal Voice Connectivity Problems, page 10
- Configuration Problems Related to Dial Peers, page 10
- Other Basic Connectivity Problems Not Relating to Dial Peers, page 13
- Fax Connectivity Problems Across the PSTN, page 13

Normal Voice Connectivity Problems

Confirm that normal voice calls can be completed before investigating fax connectivity. If there is no telephone attached, unplug the fax machine and connect a regular telephone. If normal voice calls do not connect, the issue might be network-related and you should troubleshoot the problem as a normal voice connectivity issue before proceeding with fax troubleshooting.

Configuration Problems Related to Dial Peers

Some configuration problems are associated with dial peers, for example:

- Wrong Dial Peer Being Matched, page 10
- Incorrectly Configured Dial Peers on One or Both Sides, page 12

Wrong Dial Peer Being Matched

After ensuring that voice calls can be successfully completed in both directions through the voice network, issue the **show call active voice brief** command and note the dial peers that are being matched with each voice call.

Note When you have VoIP trunks, you should be able to see all the call legs with the **show call active voice brief** command. In some versions of Cisco IOS Software Release 12.2, there is a bug in the **show call active** command. As a result, a fax call that comes through a VoIP trunk no longer appears. When you issue a **show call active fax brief** command, the call is listed. For more information about this bug, see Cisco bug IDs CSCdx50212 and CSCdv02561 (registered customers only).

Note Ensure that the configured dial peer is the peer that is being matched.

In the command output shown below, you can see that the outbound VoIP call leg is using peer ID 100.

```
Router# show call active voice brief
```

```
<snip>

Total call-legs: 2
1218 : 51710253hs.1 +415 pid:400 Answer 400 active
dur 00:01:08 tx:3411/68220 rx:3410/68200
Tele 3/0/0:43: TX:68200/6820/0ms g729r8 noise:0 acom:2
 i/0:-51/-44 dBm

1218 : 51710396hs.1 +272 pid:100 Originate 100 active
dur 00:01:09 TX:3466/69320 rx:3467/69340
IP 2.1.1.2:17092 rtt:56ms pl:64730/0ms lost:0/1/0
 delay:69/69/70ms
 g729r8

Total call-legs: 2
```

A common cause of fax relay problems is that the correctly configured dial peer is not the one being matched. It is also common that there is no particular incoming VoIP dial peer configured on the terminating gateway, and Cisco IOS Software selects the first appropriate (including default) VoIP dial peer as the incoming dial peer. Hence the parameters for this incoming dial peer might not match those of the outbound dial peer on the originating gateway.

It is not always necessary to have identical configurations on the outgoing and incoming VoIP dial peers. When you have a fax relay problem, though, make sure you have a dedicated incoming VoIP dial peer on the terminating router and that its configuration matches the configuration of the outgoing VoIP dial peer on the originating router. The following configuration for ISDN-connected routers is an example of specific, matching VoIP dial peers for the destination pattern "5..." outbound on the originating gateway and inbound on the terminating gateway.

Originating Gateway	Terminating Gateway
!--- Incoming POTS peer: Dial-peer voice 1 pots Incoming called number. Direct-inward-dial Port 1/0:15 !--- Outgoing VoIP peer: Dial-peer voice 2 voip Destination-pattern 5… Session target ipv4:1.1.1.1 Fax rate 144400 fax protocol t38 ls-redundancy 0 hs-redundancy 0	!--- Outgoing POTS peer : Dial-peer voice 10 pots Destination-pattern 5… No digit-strip Port 2/0:15 !--- Incoming VoIP peer: Dial-peer voice 20 voip Incoming called-number 5… Fax rate 14400 fax protocol t38 Ls-redundancy 0 Hs-redundancy 0

You can also check dial-peer matching by issuing the **debug voip ccapi inout** command. The debug output from this command (as shown below) includes an ssaSetupPeer message that lists all the dial-peers matching the called number. A ccCallSetupRequest message follows with the outbound peer option indicating the outgoing VoIP dial peer selected. When multiple VoIP dial peers are configured for the same destination, it is possible that the initial call setup could fail and another dial peer could be tried. In this case, another ccCallSetupRequest appears in the debug.

debug voip ccapi inout—Originating Gateway

```
ms-3640-13b# show call active voice brief
```

```
<snip>

Total call-legs: 2
1218 : 51710253hs.1 +415 pid:400 Answer 400 active
dur 00:01:08 tx:3411/68220 rx:3410/68200
Tele 3/0/0:43: TX:68200/6820/0ms g729r8 noise:0 acom:2
 i/0:-51/-44 dBm

1218 : 51710396hs.1 +272 pid:100 Originate 100 active
dur 00:01:09 TX:3466/69320 rx:3467/69340
IP 2.1.1.2:17092 rtt:56ms pl:64730/0ms lost:0/1/0
 delay:69/69/70ms
 g729r8

Total call-legs: 2
```

debug voip ccapi inout —Originating Gateway

```
.Jun 4 21:06:43.461: ssaSetupPeer cid(19)
 peer list: tag(400) called number (5074)

.Jun 4 21:06:43.461: ccCallSetupRequest
 (Inbound call = 0x13, outbound peer =100,
  dest=, params=0x62F1CC70 mode=0, *callID=0x62F1CFD8,
  prog_ind = 0)
```

On the terminating voice gateway, the first line of the **debug voip ccapi inout** call trace (as shown below) is a **cc_api_call_setup_ind message** with a **peer_tag** option that refers to the incoming VoIP dial peer on the terminating gateway.

debug voip ccapi inout—Terminating Gateway

```
.Jun 4 21:06:43.461: cc_API_call_setup_ind
 (vdbPtr=0x62F07650,
 callInfo={called=5074,called_oct3=0x80,
 calling=5075, calling_oct3=0x0,>calling_oct3a=0x83,
 calling_xlated=false,
 subscriber_type_str=Unknown,fdest=1,
 peer_tag=400, prog_ind=0},callID=0x635F72D0)
```

Incorrectly Configured Dial Peers on One or Both Sides

After you confirm that the correct dial peer is being matched (in this case dial-peer 100 for the originating gateway and dial peer 400 for the terminating router), confirm in the configuration that the dial peer is configured correctly for fax. Here are some common errors to check for on both sides of the call:

- Fax relay is disabled (that is, the **fax rate disable** command has been issued on the dial peer) while a low bandwidth codec has been in use.
- The dial peer on one voice gateway is configured for Cisco fax relay but the other voice gateway is a Cisco AS5350/AS5400. Cisco AS5350/AS5400 universal gateways support only T.38 so the negotiation fails.
- The default dial peer is being used inbound on the terminating gateway, and default parameters do not match those for the outbound dial peer on the originating gateway.
- Incorrect compand type.

The companding type for the US is µ-law, for Europe and Asia it is a-law. You can issue the **show voice call** command to see which value is currently configured. If you are on a BRI or E1 port, the companding type on the router does not match the one on the connected device, and calls sometimes fail and sometimes connect, but the voice becomes heavily distorted so that the person becomes unrecognizable and a high low-pitch noise level appears.

In Cisco IOS Release 12.2(3) the **compand-type** command is missing on the BRI ports, leaving the companding type to the default value. For more information about this bug, see Cisco bug IDs CSCdv00152 and CSCdv01861 (registered customers only).

Other Basic Connectivity Problems Not Relating to Dial Peers

Here are some basic connectivity issues that are related to dial peers:

- Cisco IOS software incompatibilities on gateway pairs. It is not always required that Cisco IOS software releases match, but you should check releases when problems occur.
- Compressed Real-Time Transport Protocol (cRTP) has several known problems. Fixes are available for these problems and it makes sense to disable cRTP when problems occur. You can then decide whether a Cisco IOS software upgrade is an appropriate course of action.
- On Cisco AS5300 voice gateways, ensure that the VCWare and Cisco IOS software are compatible.

Fax Connectivity Problems Across the PSTN

If voice calls work in both directions but fax calls are failing in at least one direction, check that normal faxing between these two machines works across the PSTN. In other words, ensure that the fax machines successfully transmit faxes to each other using the PSTN without traversing the VoX network. If they do not, the fax machines may have problems that need to be addressed before you consider fax relay problems.

Checking for Slips and Other Errors on Digital Interfaces

If there are any T1 or E1 digital connections used by the routers performing fax relay, make sure that they are error free. Fax relay is very sensitive to errors on digital interfaces, especially slips. The errors may not be noticeable on voice calls but they can cause faxes to fail.

show controller T1(E1) 1/0 Command

```
Router# show contr t1 1/0
T1 1/0 is up.
Applique type is Channelized T1
Cablelength is long gain36 0db
No alarms detected.
alarm-trigger is not set
Version info Firmware: 20010805, FPGA: 15
Framing is ESF, Line Code is B8ZS, Clock Source is Line.
Data in current interval (132 seconds elapsed):
0 Line Code Violations, 0 Path Code Violations
0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs,
0 Degraded Mins
0 Errored Secs, 0 Bursty Err Secs, 0 Severely Err Secs,
 0 Unavail Secs
```

The T1 or E1 controllers at both originating and terminating gateways should be error free, as shown above. If errors occur, repeat the **show controller** command several times during the call to see if the number of errors increases. The most common problem of slips is a synchronization problem resulting in clocking errors.

In packet voice networks, confirming that the router is clocking from the line is usually sufficient. If it is not, ensure the **clock source line** command is entered at the controller level. However, in VoATM or TDM networks where a clocking hierarchy is established and the routers may need to pass clock through the network, additional considerations are needed.

On Cisco modular acccess routers with AIM voice cards installed, the controller shows “controlled slips” unless you add the **network-clock-participate** and **network-clock-select** commands.

On the Cisco 7200VXR platform, the **frame-clock-select** command is required for the voice cards. This command is particularly important for the Cisco 7200VXR voice gateways because, by default, the internal TDM bus is not driven by the local oscillator. Since the E1 trunks are normally synchronized to the telephony network, the result is hidden clocking errors and intermittent fax transmission problems. More detail is available in Cisco bug ID CSCdv10359 (registered customers only).

Checking Fax Interface Type

On some platforms, including the Cisco 3660, Cisco AS5300, Cisco AS5350, Cisco AS5400, and Cisco AS5800 platforms, the router defaults to fax interface-type modem. The **fax interface-type modem** global configuration command forces fax calls to a modem (usually for T.37 Store and Forward fax) and not to a DSP. For Cisco fax relay to work, the fax call must be sent to a DSP, which means it must be configured by the issuing of the **fax interface-type vfc** command.

```
Router(config)# fax interface-type ?
modem Use modem card
vfc   Use Voice Feature Card

Router(config)# fax interface-type vfc
You must reload the router
```

Make sure you reload the router; otherwise the command does not take effect. Fax calls fail on the universal gateways listed above with Cisco fax relay (or T.38), so this is an important command to check.

The **fax interface-type vfc** command was not necessary in Cisco IOS software releases prior to Release 12.2. The problem is commonly seen when one of the voice gateways listed above is upgraded to Cisco IOS Release 12.2 or later.

Ensuring That the Fax Codec is Loaded During the Fax Call

Each fax machine displays the remote fax machine ID on its LCD screen at the completion of the fax negotiation phase. It is unlikely that the fax machines could complete negotiation if the fax codec has not been successfully downloaded. On the other hand if no remote fax machine ID is displayed, further debugging in this area is appropriate.

There are two ways to make sure that the voice gateways detect the fax transmission and successfully load the fax codec:

- Issue the **debug vtsp all** command and the **debug voip ccapi inout** call trace.
- Issue the **show voice trace** command. Show commands are less resource intensive on the router than debug commands and are preferable in production networks. Shown below is sample output from a **show voice trace** command on an ISDN interface:

```
BrisVG200gwy01# show voice trace 1/0:15
1/0:15 1
1/0:15 2
1/0:15 3
1/0:15 4
1/0:15 5
1/0:15 6
1/0:15 7
1/0:15 8
1/0:15 9
1/0:15 10 State Transitions: timestamp (state, event) -> ...
63513.792 (S_SETUP_REQUEST, E_TSP_PROCEEDING) ->
63515.264 (S_SETUP_REQ_PROC, E_TSP_ALERT) ->
63515.264 (S_SETUP_REQ_PROC, E_CC_BRIDGE) ->
63515.332 (S_SETUP_REQ_PROC, E_CC_CAPS_IND) ->
63515.332 (S_SETUP_REQ_PROC, E_CC_CAPS_ACK) ->
63515.348 (S_SETUP_REQ_PROC, E_CC_CAPS_IND) ->
63515.348 (S_SETUP_REQ_PROC, E_CC_CAPS_ACK) ->
63515.356 (S_SETUP_REQ_PROC, E_CC_CAPS_IND) ->
63515.356 (S_SETUP_REQ_PROC, E_CC_CAPS_ACK) ->
63518.656 (S_SETUP_REQ_PROC, E_CC_REQ_PACK_STAT) ->
63518.660 (S_SETUP_REQ_PROC, E_DSP_GET_VP_DELAY) ->
63518.660 (S_SETUP_REQ_PROC, E_DSP_GET_VP_ERROR) ->
63518.660 (S_SETUP_REQ_PROC, E_DSP_GET_RX) ->
63518.660 (S_SETUP_REQ_PROC, E_DSP_GET_TX) ->
63521.028 (S_SETUP_REQ_PROC, E_CC_REQ_PACK_STAT) ->
63521.028 (S_SETUP_REQ_PROC, E_DSP_GET_VP_DELAY) ->
63521.028 (S_SETUP_REQ_PROC, E_DSP_GET_VP_ERROR) ->
63521.028 (S_SETUP_REQ_PROC, E_DSP_GET_RX) ->
63521.028 (S_SETUP_REQ_PROC, E_DSP_GET_TX) ->
63524.128 (S_SETUP_REQ_PROC, E_TSP_CONNECT) ->

!--- Fax tone detected:

63529.352 (S_CONNECT, E_DSP_TONE_DETECT) ->
63529.356 (S_LFAX_WAIT_ACK, E_PH_CODEC_ACK) ->

!--- Fax codec being downloaded to DSPs:
```

Disabling Fax Relay and Change Codec for Pass-Through

In the previous troubleshooting steps you established that voice calls work, faxes work through PSTN, and all digital interfaces in the fax relay path are free from errors. The current step determines whether faxes can go through with fax relay disabled. For the VoIP/VoATM/VoFR dial peers, enter the following to disable the fax rate:

```
Router(config)# voice-port 2/0:15
Router(config-voiceport)# no echo-cancel enable
Router(config)# dial-p voice 3
Router(config-dial-peer)# fax rate disable
Router(config-dial-peer)# codec g711ulaw
Router(config-dial-peer)# no vad
```

Make sure these commands are entered on both gateways. These commands disable fax relay, disable echo cancellation, and force the call to use a high-bandwidth codec without VAD. The router then samples the tones as it would for a normal voice call, and with the high-bandwidth codec (G.711), the most precise sample possible is captured. The tone to be replayed on the other side is as accurate as possible. The caveat to this step is that since G.711 is a 64- kbps bandwidth codec, each call consumes up to 80 kbps (for VoIP) when additional transport protocol overhead is added.

If this test is positive, two things have been accomplished. First, if per-call bandwidth consumption is not a major issue for the network, there is now a potential fax pass-through workaround for the fax relay problem. Second, and more significantly, if bandwidth consumption is an issue, the problem has been isolated to the fax relay software and you should open a TAC case.

If this test fails, whatever is causing the fax calls to fail with fax relay is also probably causing the failures with this test. The problem might be that the network might have a large amount of jitter or packet loss.

Checking for Packet Loss on the Voice Network

Her is the easiest and most accurate way to determine if there is a packet loss:

1. Disable VAD on the VoIP dial peers.
2. Make a voice call between the same ports where the fax machines are connected. (Fax machines may be able to serve as ordinary phones or you might need to connect the handsets to the same ports where the fax machines are connected.)
3. When the call is connected, there are two main steps to take:
   a. Issue the **show voice dsp** command. You can see in the output that one of the DSP channels has the configured codec loaded. Usually the column "TX/RX-PAK CNT" shows that the transmit and receive packet counters are equal, meaning that no packets are being lost. If the counters are not equal, packets might be getting lost. Type the **show voice dsp** command several times at 30-second intervals to determine if the difference increases and packets are being lost.
   b. Issue the **show voice call summary** command to see which port (and timeslot if applicable) is allocated to the voice call. Type **terminal monitor** and issue the **show voice call** command with the voice port (and timeslot if applicable) to get the detailed DSP statistics. In the "***DSP VOICE VP_ERROR STATISTICS***" section of the output, look for the counters. They are usually 0 or below 20. If the any of the counters have higher than 20, investigate the packet loss.

If the network appears to be lossy, it is not reasonable to expect fax relay to work reliably. It is possible to disable ECM, but further investigation is probably needed to ensure QoS is provisioned end-to-end so that the voice and fax relay traffic has priority and is never lost during the congestion.

Disabling Fax Relay ECM (Cisco Proprietary VoIP Only)

For networks with packet loss and lots of jitter, disabling ECM can improve fax relay calls. Issue the command **fax-relay ecm disable** to turn off ECM, so that a larger amount of jitter and packet loss can be tolerated.

Issuing the **fax-relay ecm disable** command improves fax relay performance in lossy networks, but this command is also recommended for basic troubleshooting. Even if there is not a noticeable jitter problem in the network, this command can sometimes help determine fax relay problems. This command is available under VoFR and VoATM dial peers but currently works only for VoIP.

Note This command also activates the packet loss concealment feature.

```
Router(config-dial-peer)# dial-peer voice 3
Router(config-dial-peer)# fax-relay ECM disable
```

Enabling T.38 Packet Redundancy (T.38 VoIP Only)

If T.38 for VoIP is being used as the fax relay protocol, you can turn on the T.38 packet redundancy feature by configuring the following command under the appropriate dial peers on both gateways:

```
Router(config-dial-peer)# fax protocol t38
 Ls-redundancy X Hs-redundancy Y
```

where X > 0 and Y = 0 (only make changes to Ls-redundancy)

If Cisco proprietary fax relay is in use, an alternative or additional option to disabling ECM is to change fax relay protocol to T.38 so that the T.38 packet redundancy feature can be tested. This feature might alleviate failure due to packet loss. However, T.38 packet redundancy significantly increases bandwidth usage, and it is preferable to eliminate the packet loss altogether, rather than alleviating the failures it causes.

Setting the Fax NSF Command to All Zeroes

The **fax nsf** command can be helpful for any fax machine that alters the NSF field during fax negotiation for proprietary encodings. This command allows the router doing fax relay to override the settings made by the fax machines trying to implement proprietary encodings. Before the **fax nsf** command was available, fax relay failed for such fax machines. Typically the **fax nsf** command is used to set the NSF field to all zeroes, forcing a standard fax negotiation from both sides. Using this command has been successful with certain brands of fax machines such as Harris and Lanier, and it is recommended when fax relay is failing.

```
Router(config-dial-peer)# fax NSF 000000
```

Achieving the Final Stages of Resolution

If the preceding troubleshooting steps do not resolve the fax relay issue, the problem might require more advanced troubleshooting. Listed below are additional steps to try before you open a case with the Cisco Technical Assistance Center (TAC). See the "Obtaining Technical Assistance" section on page xxvi for information about contacting TAC.

- Learn the brands and models of the fax machines that are failing, and investigate those brands and models for known issues.

  Sometimes there are CARE cases or bugs that address problems for a certain brand of fax machine. For example, a search in the Bug Toolkit (registered customers only) for a Pitney Bowes fax shows a bug with Pitney Bowes fax machines and Cisco fax relay (CSCdu78373 (registered customers only)). This bug is not in the Cisco IOS software but is an incompatibility with the Pitney Bowes proprietary fax signaling protocol. A problem arises when the fax devices on each side of a connection are Pitney Bowes 9920s or 9930s. The workaround is to disable the proprietary protocol on the fax machines or to disable fax relay and use a higher bandwidth codec.

- Use search tools to look for known fax problems in the Cisco IOS software release where the problem is occurring.

  In the previous step, searches were made for a specific fax brand in the hope of identifying a known issue between a certain fax brand and the Cisco fax relay code. The next step is to perform a generic search, because there could be a fax relay bug in the Cisco IOS software release installed.

For example, if fax relay using VoFR is not working in Cisco IOS Software Release 12.1(2)T, you can search for bugs using the Bug Toolkit on Cisco.com. In this example, you would use the following values:

- Major version: 12.1
- Revision: 2
- Feature/component: VoFR
- Keyword: fax

One of the bugs is Cisco bug ID CSCdr65984 (registered customers only), entitled “fax doesn't work for vofr.” This bug causes all fax relays to fail for VoFR, and an upgrade is needed to a Cisco IOS software release in which this bug is no longer present.

- Eliminate hardware faults.

  In some cases it is easier to isolate the problem by excluding potential problem sources one by one. You can do this by replacing different hardware parts and using alternative IP connections between the gateways.

  When extra hardware is available, the following steps can help.

  - Use different ports on the routers—If your configuration involves two gateways connected to the PBXs or PSTN with E1 or T1 and if you have the FXS ports available, try to connect the fax machines directly to the FXS ports on the voice gateways. This procedure helps you further isolate the problem by excluding the possibility of the E1 cards failing, problems on the telephony side, and E1 synchronization or cable problems.
  - Try different hardware—If you have another voice gateway with FXS ports available, try to connect it directly with the Ethernet crossover cable to each of the voice gateways and send a fax using the fax machine connected to the FXS port. This procedure helps determine if there are problems in the VoX network involving queuing, fragmentation, or prioritization.
  - Use debug commands on the router to determine what is happening—See the following section for details about debug commands that are useful for troubleshooting fax relay problems.

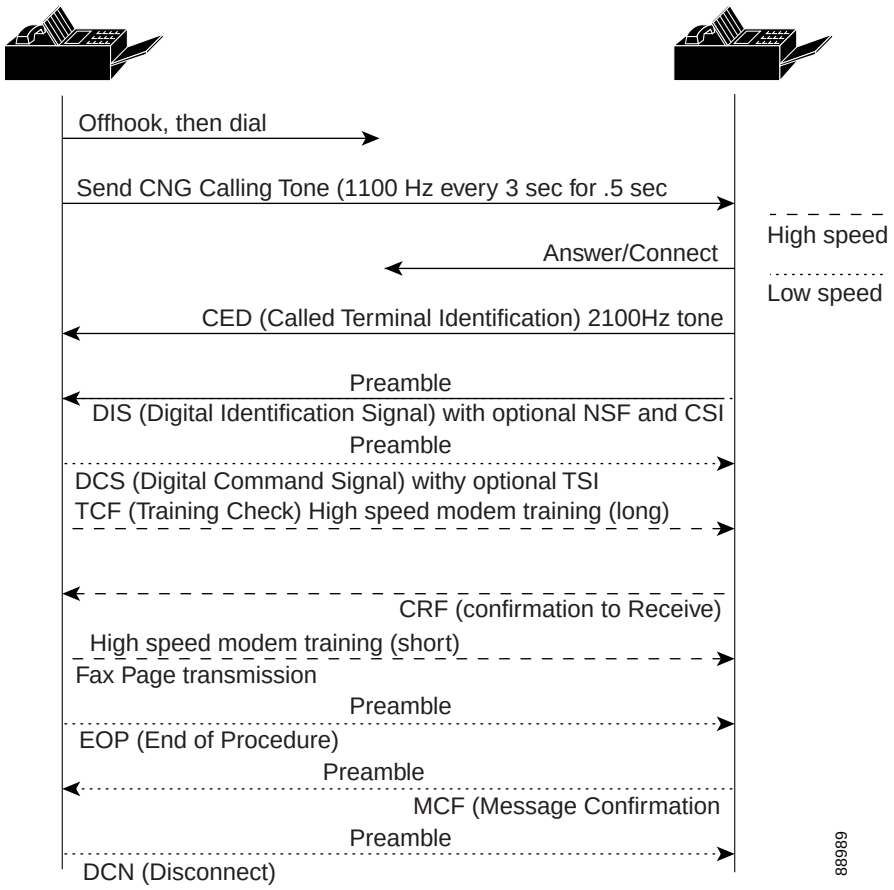
Debugging Fax Relay

The following topics are associated with fax relay debugging:

T.30 Messages

The debugs can be difficult to understand if you are not familiar with the messaging that occurs during a typical fax transmission. Figure 53 is a graphical representation of the basic T.30 transactions that occur for a single page fax transmission.

Figure 53 T.30 Transactions



Describing the details of these transactions is beyond the scope of this document, but listed below are definitions of the basic transactions that are seen during fax relay. The list is alphabetical for quick reference and includes messages that you are likely to see when you are debugging Cisco fax relay. For more in-depth information on this messaging or for information on messages that are not listed below, see the T.30 specification.

- **CED (called terminal identification)**—A 2100 Hz signal that is transmitted by the terminating fax device upon answering a fax call. This signal temporarily disables echo cancellers that are present on the connection to prepare the line for data transmission.
- **CFR (confirmation to receive)**—A response confirming that the previous messaging and training has been completed and that fax page transmission can begin.
- **CNG (calling tone)**—An 1100-Hz tone that is on for 0.5 seconds and then off for 3 seconds. This signal identifies the fax terminal as a nonspeech device. The signal also indicates that the initiating fax terminal is awaiting the DIS signal from the terminating fax terminal.
- **CRP (command repeat)**—A response that indicates that the previous command was received in error and needs to be repeated. (Optional)
- **CSI (called subscriber identification)** —Can be used to provide the specific identity of the called fax terminal through its international telephone number. (Optional)
- **DCN (disconnect)**—Ends the fax call and requires no response.
- **DIS (digital identification signal)** — Identifies the capabilities of the called fax terminal.

- **DTC (digital transmit command)** —The response to the capabilities identified by the DIS signal. Here is where the calling fax terminal matches its capabilities with those provided in the called fax terminal's DIS message.
- **EOM (end of message)** — Indicates the end of a complete page of fax information.
- **EOP (end of procedure)**— Indicates the end of a complete page of fax information and signals that no further pages are to be sent. The other device can proceed to the disconnect phase of the fax call.
- **FTT (failure to train)** — Used to reject a training signal and request a retrain (retrains usually occur at lower modulation speeds).
- **MCF (message confirmation)** —Indicates that a message has been satisfactorily received.
- **MPS (multipage signal)**—Indicates the end of a complete page of fax information and signals that the receiver is ready for additional pages.
- **NSF (nonstandard facilities)**— Can be used to identify specific capabilities or requirements that are not covered by the T-series specifications. (Optional)
- **RTN (retrain negative)**— Indicates that a previous message has not been satisfactorily received. Retraining is needed to proceed (usually at a lower modulation speed).
- **RTP (retrain positive)**— Indicates that a complete message has been received and that additional messages may follow after retraining.
- **TCF (training check)**— Sent through the higher-speed T.4 modulation system (versus the 300-kbps V.21 modulation used for the previous T.30 signaling) to verify training and indicate the acceptability of sending fax pages at this transmission rate.
- **TSI (transmitting subscriber identification)**— Identifies the transmitting (calling) fax terminal. (Optional)

Fax Relay Debug Commands

Useful fax relay debug commands include:

- **debug fax relay t30 all**
- **debug vtsp all**
- **debug vtsp vofr subframe 3**
- Additional Debug Commands

debug fax relay t30 all

The debug for Cisco fax relay is enabled with the **debug fax relay t30 all** command.

```
Router# debug fax relay t30 all
Debugging fax relay t30
```

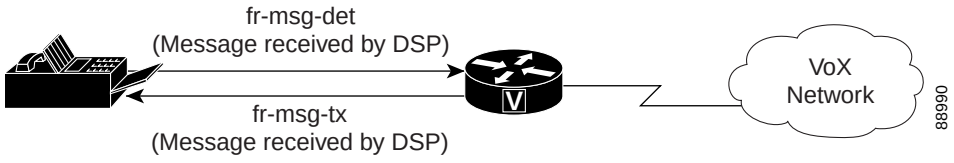
Shown below is a copy of a debug from a failed fax relay session. This is a debug from the originating fax gateway running Cisco IOS Release 12.2(7a).

```
Router#
Dec 5 07:49:13.073: 1/2:62 1281347052 fr-entered (10ms)
Dec 5 07:49:17.985: 1/2:62 1281351950 fr-msg-det CRP
Dec 5 07:49:20.105: 1/2:62 1281354070 Fr-MSG-TX NSF
Dec 5 07:49:20.655: 1/2:62 1281354620 Fr-MSG-TX good crc,
 19 bytes
Dec 5 07:49:20.720: 1/2:62 1281354680 Fr-MSG-TX DIS
DEC 5 07:49:22.350: 1/2:62 1281356310 fr-msg-det TSI
```

```
DEC 5 07:49:23.045: 1/2:62 1281357000 fr-msg-det DCS
DEC 5 07:49:27.346: 1/2:62 1281361290 Fr-MSG-TX FTT
DEC 5 07:49:28.836: 1/2:62 1281362780 fr-msg-det TSI
DEC 5 07:49:29.531: 1/2:62 1281363470 fr-msg-det DCS
DEC 5 07:49:29.740: 1/2:62 1281363680 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:30.362: 1/2:62 1281364300 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:30.804: 1/2:62 1281364740 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:30.852: 1/2:62 1281364790 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:33.868: 1/2:62 1281367800 Fr-MSG-TX FTT
DEC 5 07:49:35.414: 1/2:62 1281369340 fr-msg-det TSI
DEC 5 07:49:36.113: 1/2:62 1281370040 fr-msg-det DCS
DEC 5 07:49:36.515: 1/2:62 1281370440 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:36.908: 1/2:62 1281370830 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:37.559: 1/2:62 1281371480 fr-msg-det bad crc,
0 bytes
DEC 5 07:49:37.784: 1/2:62 1281371700 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:37.900: 1/2:62 1281371820 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:40.133: 1/2:62 1281374050 Fr-MSG-TX FTT
DEC 5 07:49:41.888: 1/2:62 1281375800 fr-msg-det TSI
DEC 5 07:49:42.583: 1/2:62 1281376490 fr-msg-det DCS
DEC 5 07:49:43.173: 1/2:62 1281377080 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:44.937: 1/2:62 1281378840 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:45.386: 1/2:62 1281379290 fr-msg-det bad crc,
 0 bytes
DEC 5 07:49:46.941: 1/2:62 1281380840 Fr-MSG-TX FTT
DEC 5 07:49:48.503: 1/2:62 1281382400 fr-msg-det DCN
DEC 5 07:49:50.631: 1/2:62 1281384520 fr-end-dcn
```

This debug shows the T.30 events that are taking place in the DSP during fax relay. Remember that the debugs are taking place from the perspective of the DSP interacting with the fax device. Any Fr-MSG-TX or transmit message is being transmitted from the DSP to the connected fax device. Any message that the DSP says that it detects (an fr-msg-det message, is a message that it received from the connected fax device. Figure 54 illustrates the directional flow of the DSP messages when the **debug fax relay t30 all** command is issued.

Figure 54 DSP Message Flow



From the failed fax transaction shown in the debug above, you can see several badcyclic redundancy check (CRC) messages followed by a failure to train (FTT) message from the far side. From the debugs it looks like the problem involves the training signal. The bad crc errors and the FTT message returned from the other side indicate that the signal is corrupted or incompatible with the Cisco fax relay protocol. This debug is taken from a fax relay problem that occurs with a Lexmark Optra fax machine. The

Lexmark is V.34-capable and attempts to connect at V.34 rates. V.34 is not supported in Cisco fax relay and the training errors shown here occur. See Cisco bug ID CSCdv89496 (registered customers only) for more details.

debug vtsp all

There are also other debug commands that might be useful for troubleshooting fax relay problems. These debugs might not be as easy to read or provide as much information as the T.30 debugs described above, but they can still be useful.

Voice Telephony Service Provider (VTSP) is an architecture for the interface between the Cisco IOS call control and a DSP endpoint connected to standard telephony equipment such as a PBX, fax, or central office via analog or digital interfaces.

For VoIP T.38 or fax relay, **debug vtsp all** can provide useful state information from the router. This debug command can be used to determine if the fax codec has been downloaded into the DSP.

debug vtsp vofr subframe 3

Another fax relay debug command that is helpful for fax using VoFR and VoATM is **debug vtsp vofr subframe 3** . This command outputs FRF.11 frames that have an Annex D fax relay payload type. There is a significant amount of output from this command even with just one fax relay call, and the hexadecimal must be decoded (the FRF.11 specification is helpful for hexadecimal decoding).

Additional Debug Commands

To debug T.38 capabilities exchange issues, use the **debug cch323 h245** command.

To debug DSP message exchanges between applications and the DSP, use the following debug commands:

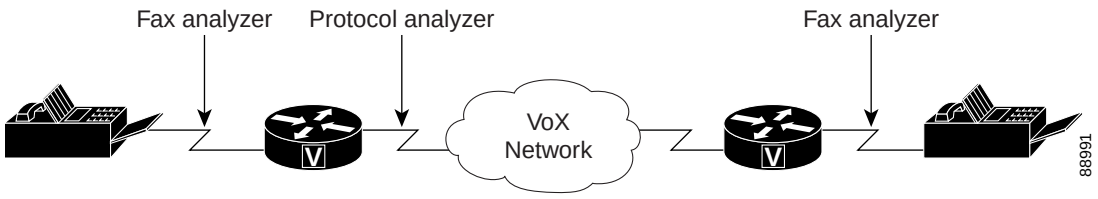
- **debug voip ccapi inout**
- **debug hpi all** (on the Cisco 5300/2600/3600 and all other voice platforms using the TI c54x DSPs)
- **debug nextport vsmgr detail** (on the NextPort DSP platforms (Cisco 5400, 5850))

Fax Analyzers

Sometimes it is necessary to go beyond the debugging capabilities of the Cisco voice gateways to resolve fax relay problems. You can use tools such as protocol analyzers and fax analyzers to see what is occurring during fax relay operation. Fax analyzers such as the Genoa ChannelProbe/FaxProbe by QualityLogic or the HP Telegra can be positioned between the fax device and the Cisco gateway to capture what is occurring. Protocol analyzers such as Sniffer and Domino can be helpful when you need to view the fax relay packets that are being exchanged between the routers.

The ability to resolve a complex problem sometimes requires using a combination of equipment—an analyzer to capture the fax traffic at each fax machine and a protocol analyzer to capture the fax relay packets. A single fax call is placed to reproduce the problem, and then the information is captured from the attached devices for analysis. Figure 55 shows where this test equipment is placed in the network.

Figure 55 Test Equipment Placement



Most of the fax analyzers have adequate help screens and documentation. The T.30 specification is also very helpful. For the protocol analyzers, decoding can be a little more difficult because sometimes the encodings are proprietary or the analyzer software does not have the specific decode needed. For fax relay using VoFR and VoATM, Cisco gateways use standards-based Annex D from the FRF11 specification. If the protocol analyzer cannot decode the frame, the frame can be manually decoded through use of this specification. With fax relay and VoIP, a Cisco proprietary format is used for the fax relay packets.

With fax analyzer and protocol analyzer information, you should be able to resolve fax relay problems. Few fax relay problems reach this point, and when they do, escalation and DE resources should already be involved for further assistance.

For more information about troubleshooting fax relay, refer to the *Fax Relay Troubleshooting Guide*, document 20227.

Fax Detection

Use the following tips to resolve problems that keep fax detection from working correctly:

- On the router that you are using for the fax detection application, make sure that you have installed at least the minimum version of Cisco IOS software that is listed in the *Cicso Fax Services over IP* guide.
- Before configuring fax detection, make sure that your voice application is functional by putting a series of calls through.
- Before configuring fax detection, make sure that your fax application is functional by sending a series of faxes.
- After configuring fax detection, issue the **debug voip ivr script** command to display debug information from the fax detection script. Put through a series of voice calls and fax calls to ensure correct operation. The debug output that is displayed when you put calls through is indispensable for diagnosing failing calls and finding the source of a problem. It is the only way to verify that parameters are set to the values that you want and that they are actually taking effect. Mistakes such as typing errors in command-line interface (CLI) parameters (for example, typing “moode” for “mode”) are not recognized as errors by Cisco IOS. They are accepted without complaint when typed, yet they do not produce the desired effect during operation. It is only by watching the debug output during operation that you find these mistakes.
- Make sure that you have configured different DTMF digits for fax and for voice. If you configure both to be the same number, you are not notified immediately, as you would be with other Cisco IOS command errors. You find this error only if the **debug voip ivr script** command is enabled before a failing call comes in.

Monitoring the Voice Network

Monitoring the Voice Network

To monitor your voice network, use the following sections:

Periodic Monitoring Tasks

Use **show** commands to monitor your network.

The **show** commands are powerful monitoring and troubleshooting tools. You can use the **show** commands to perform a variety of functions:

- Monitor router behavior during initial installation
- Monitor normal network operation
- Isolate problem interfaces, nodes, media, or applications
- Determine when a network is congested
- Determine the status of servers, clients, or other neighbors

The following are some of the most commonly used **show** commands:

- **show version**—Displays the configuration of the system hardware, the software version, the names and sources of configuration files, and the boot images.
- **show running-config**—Displays the router configuration currently running.
- **show startup-config**—Displays the router configuration stored in nonvolatile RAM (NVRAM).
- **show interfaces**—Displays statistics for all interfaces configured on the router or access server. The resulting output varies, depending on the network for which an interface has been configured.
- **show controllers**—Displays statistics for interface card controllers.
- **show flash**—Displays the layout and contents of Flash memory.
- **show buffers**—Displays statistics for the buffer pools on the router.
- **show memory summary**—Displays memory pool statistics and summary information about the activities of the system memory allocator, and gives a block-by-block listing of memory use.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- **show process cpu**—Displays information about the active processes on the router.
- **show stacks**—Displays information about the stack utilization of processes and interrupt routines, as well as the reason for the last system reboot.
- **show cdp neighbors**—Provides reachability information for directly connected Cisco devices. This is an extremely useful tool for determining the operational status of the physical and data link layers. Cisco Discovery Protocol (CDP) is a proprietary data link layer protocol.
- **show debugging**—Displays information about the type of debugging that is enabled for your router.

You can always use the **?** at the command line for a list of subcommands.

Like the **debug** commands, some of the **show** commands listed previously are accessible only at the router's privileged EXEC mode (enable mode), which is explained in the "Debug Command Output on Cisco IOS Voice Gateways" chapter.

Hundreds of other **show** commands are available. For details on using and interpreting the output of specific **show** commands, refer to the Cisco IOS command references.

Tools for Monitoring the VoIP Network

The following tools can be used to monitor VoIP networks:

Cisco Voice Manager

Cisco Voice Manager (CVM) is a client-server, web-based voice management solution used by network administrators to configure and manage voice ports and create and modify dial plans on voice-enabled Cisco routers. Using CVM, network administrators can:

- Manage the configuration of FXO, FXS, E&M, and ISDN voice interfaces on voice-enabled routers
- Create and manage local (POTS) dial plans on voice-enabled routers
- Create and manage VoIP, VoFR, and VoATM network dial plans on voice-enabled routers
- Generate detailed reports using Telemate.net Quickview

For more information, see the Cisco Voice Manager documentation.

Quality of Service Device Manager

Cisco Quality of Service Device Manager (QDM) provides an easy-to-use application for configuring and monitoring advanced IP-based Quality of Service (QoS) functionality within Cisco routers and switches. The QDM application is stored in Flash memory on the Cisco product and can be run from any

workstation with proper support. QDM runs in a web browser as a Java applet. The QDM application uploads when the client web browser makes a connection to the embedded web server of the router or switch.

Once the QDM application is uploaded, the context-sensitive online help embedded within the application is designed to provide technical help associated with a particular QoS-related task. For information on the various QoS functions that can be configured in QDM, consult the online help within the QDM application. The Online Help Table of Contents can always be accessed by clicking the Help button in the upper-right corner of the QDM screen and then clicking Table of Contents. A glossary is also available as part of the online help.

QDM can be downloaded from Cisco.com and is available free of charge.

For more information, see the Quality of Service Device Manager documentation.

Cisco Service Assurance Agent

Cisco Service Assurance Agent (CSAA) is an application-aware synthetic operation agent that monitors network performance by measuring response time, network resource availability, application performance, jitter (interpacket delay variance), connect time, throughput, and packet loss. Performance can be measured between any Cisco device that supports this feature and any remote IP host (server), Cisco routing device, or mainframe host. Performance measurement statistics provided by this feature can be used for troubleshooting, for problem analysis, and for designing network topologies.

CSAA can be especially useful for enterprise and service provider networks, because it provides expanded measurement and management capabilities. In particular, the CSAA is a reliable mechanism for accurately monitoring the metrics in service level agreements (SLAs).

Because CSAA is accessible through Simple Network Management Protocol (SNMP), it can also be used in performance monitoring applications for Network Management Systems (NMSs) such as CiscoWorks2000 (CiscoWorks Blue) and the Internetwork Performance Monitor (IPM). CSAA notifications also can be enabled via Systems Network Architecture (SNA) network management vector transport (NMVT) for applications such as NetView.

SNMP notifications based on the data gathered by the CSAA allow the router to receive alerts when performance drops below a specified level and again when problems are corrected. The CSAA utilizes the Cisco Round Trip Time Monitor (RTTMON) MIB for interaction between external NMS applications and the CSAA running on the Cisco devices. For a complete description of the object variables referenced by the CSAA feature, refer to the text of the CISCO-RTTMON-MIB.my file, available from the Cisco MIB website.

For more information, refer to the "Network Monitoring Using Cisco Service Assurance Agent" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

CiscoWorks Voice Health Monitor

CiscoWorks Voice Health Monitor (VHM) helps network administrators and network operators set and maintain the stability of the VoIP network within their enterprise. VHM achieves this goal by using:

- A series of availability and health checks on the VoIP equipment in the network.
- A fault detection and escalation system to notify the users of any faults or exceptions detected.

VHM integrates with network management systems (NMSs) such as HP OpenView Network Node Manager.

With VHM, you can:

- Discover VoIP network devices and applications on a user-entered schedule
- Monitor faults in voice and data networks
- Run synthetic transaction tests, to check Cisco CallManager functions
- Check the availability and health of VoIP equipment and applications
- Obtain the status of each voice device group, such as Voice Cluster, Voice Gateway, Phone Access Switches, and Work Flow Applications
- Discover and manage Ethernet ports that have IP Phones connected to them
- Monitor IP phones in the network

For more information about VHM, refer to the CiscoWorks Voice Health Monitor documentation.

Cisco Gateway Management Agent

The Cisco Gateway Management Agent (CGMA) is the only real-time management Cisco IOS software agent and protocol for VoIP. The CGMA is a new gateway Cisco IOS agent that provides real-time call-state information for all VoIP calls. CGMA supports a push protocol, in which certain call-state changes result in a message being sent out of CGMA by the gateways. The interface from the CGMA is the Real Time Management Protocol (RTMP). RTMP is a lightweight XML-based protocol that uses TCP as the transport protocol. This solution allows service providers to monitor their calls (session initiation protocol (SIP) and H.323 networks) and to view call detail records (CDRs) and trunk utilization in real time. The validated gateways for the CGMA include the Cisco 2600 series, the Cisco 3600 series, and the Cisco Catalyst 5000 series. The Cisco IOS release that has been validated on all gateways is the 12.2(2)Xb mainline release.

Cisco QoS Policy Manager

QoS Policy Manager (QPM) lets you analyze traffic throughput by application or service class, and then leverage that information to configure QoS policies to differentiate traffic and to define the QoS functions to be applied to each type of traffic flow.

By simplifying QoS policy definition and deployment, QPM makes it easier for you to create and manage end-to-end differentiated services in your network, thus making more efficient and economical use of your existing network resources. For example, you can use policies that ensure that your mission-critical applications always get the bandwidth they require.

QPM is suitable for large-scale enterprise deployments and IP telephony deployments, consisting of hundreds or thousands of devices. QPM facilitates management of large networks by providing advanced user authorization capabilities through integration with Cisco Access Control Server (ACS).

You can partition the network into administrative and deployment domains. QPM allows you to organize policies in separate deployment groups, and it supports best practices for phased deployments. Using separate deployment groups, you can also use QPM to test what-if scenarios, and run time-based deployment.

For more information about Cisco QPM, refer to the QoS Policy Manager documentation.

For more information about VoIP QoS troubleshooting, refer to *Monitoring Voice over IP Quality of Service*, document 17962.

Voice Performance Statistics on Cisco Gateways

The Voice Performance Statistics on Cisco Gateways feature enables the collection of voice call signaling statistics and VoIP AAA accounting statistics based on user-configured time ranges. The statistics can be displayed on your console or formatted and archived to an FTP or syslog server. This feature can assist you in diagnosing performance problems on the network and identifying impaired voice equipment. This chapter contains the following topics:

Prerequisites for Voice Performance Statistics on Cisco Gateways

- Your gateway must be configured to support VoIP and must be functioning properly.

Restrictions for Voice Performance Statistics on Cisco Gateways

- This feature does not support Media Gateway Control Protocol (MGCP).
- This feature does not support parsing, presentation, or analysis of syslog files.
- The integrity of statistical information in the syslog files is not guaranteed because of unreliable User Datagram Protocol (UDP) transport.
- If the gateway clock needs to be synchronized after the gateway is reset or rebooted or during Network Time Protocol (NTP) client synchronization, there may be a problem for collection during the collection intervals.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

The call-statistics data is dependent on the start and end time of the interval; that is, the collection is time driven, not event driven. The following two situations will result in erroneous call-statistics data:

- Clock reset during a collection interval.
- Clock set to a time during an already specified interval.

To avoid any call-statistics data from being collected within an incomplete interval, this feature reports only call-statistics data that is collected during a complete interval. This includes call-statistics data that is pushed to an FTP server or stored on a gateway.

This feature cannot assure accuracy or consistency of the reports generated when large clock updates occur during a batch reporting period.

- Call statistics cannot be accessed using RADIUS protocols.
- The signaling behavior of two-stage non-Direct-Inward-Dialing (non-DID) ISDN calls using a default session application is not supported.
- Digital Signal 0 (DS0) is not supported.

Information About Voice Performance Statistics on Cisco Gateways

To configure the Voice Performance Statistics on Cisco Gateways feature, you should understand the following:

The benefits of this feature are listed in the "Benefits of Voice Performance Statistics on Cisco Gateways" section on page 9.

Basic Terminology and Feature Design

The Voice Performance Statistics on Cisco Gateways feature is designed to include many options for collecting, archiving, and displaying call statistics records, which include PSTN interface/port statistics, IP interface statistics, and counts of RADIUS accounting messages. This feature also enables counts of Cisco IOS generated internal error codes (IECs). The basic terminology that describes the functionality of the feature is explained in the subsections that follow.

The following are feature options:

- Counting accounting records (messages to and from RADIUS servers).
- Collecting various signal-layer (IP and PSTN interfaces) statistics from individual gateway ports.
- Displaying the signaling statistics at different aggregation levels.
- Collecting IECs.
- Collecting the statistics at user-configured time intervals.

- Archiving the statistics to an FTP or syslog server and formatting the output.
- Displaying the statistics on a console.
- Displaying the available memory and the memory that has been used for the collection of records.
- Specifying thresholds for packet jitter, lost packets, and packet latency.
- Specifying the length of time to be used as the maximum call duration.
- Specifying a maximum time for which to store the statistics in system memory.

This section has the following subsections:

- What Are the Types of Accounting Statistics?, page 3
- What Are the Types of Signaling Statistics and Aggregation Levels?, page 3
- What Are IECs?, page 5

What Are the Types of Accounting Statistics?

Accounting record statistics are counts of RADIUS messages that are transmitted to and received from a RADIUS server. They can be collected by method list, type of RADIUS message (for example, starts, interim updates, and/or stops), and call-leg interface association (PSTN or IP). This feature can be configured to count all of these message types or only a subset of them.

A method list is a sequential list used by the RADIUS client on the gateway that defines the authentication methods used to authenticate a user. For the purposes of the Voice Performance Statistics on Cisco Gateways feature, you are required to specify only the name of the method list on the gateway.

Once enabled and configured, the feature counts RADIUS messages on both inbound and outbound call legs. Each time a RADIUS accounting message is received by the gateway, it is counted as successful if it is accepted and processed by the RADIUS agent on the gateway; each time a RADIUS accounting message is transmitted by the gateway, it is counted as passed if an ACK comes back from the RADIUS server.

You can also specify that accounting messages be collected from a broadcast method list, in which case you can set all the server groups that are in a method list to monitor the server group acknowledgements.

What Are the Types of Signaling Statistics and Aggregation Levels?

The signaling statistics are collected at the port level, but can be displayed at various aggregation levels.

The aggregation levels are hierarchical. The highest level is a summary of total statistics for all aggregation levels on the gateway, whereas the lowest level provides statistics for each voice port. Statistics can also be collected for the following aggregation levels:

- Gateway level
- VoIP level
- PSTN level
- Trunk group level
- Voice-port level

An example of collected statistics at the different aggregation levels for a PSTN statistic labeled “X” is as follows:

- When the aggregation level is *gateway*: X = 4
- When the aggregation level is *trunk group*:

  - Trunk group A (configured ports 1 and 3): X = 3
  - Trunk group B (configured ports 2 and 4): X = 1
- When the aggregation level is *port*:
  - Port 1: X = 1
  - Port 2: X = 0
  - Port 3: X = 2
  - Port 4: X = 1

The following are supported call-statistics fields that can be collected on Cisco gateways:

- Incoming calls—All incoming call attempts, whether successful or not.
- Incoming calls answered by the gateway—Incoming calls that were answered.
- Incoming calls rejected by the gateway—Incoming calls that, for whatever reason, failed.
- Outgoing calls attempted—Outgoing calls regardless of whether they were successful.
- Outgoing calls that receive answers—Calls that were answered.
- Outgoing calls fail—Calls that failed.
- Total duration of all incoming and outgoing calls—Total duration from outgoing seizure to disconnect.
- Total duration of incoming and outgoing answered calls—Total connected time: from answer to disconnect.
- Originating side disconnected before outgoing calls connected.
- Number of incoming and outgoing calls whose connected time is less than the configured minimum call duration (MCD).
- Number of answered incoming and outgoing calls terminated with any cause codes other than “normal.”
- Total duration (after the dial delay) on incoming calls—Defined as “alert sent time–setup in time.”
- Total duration (after the dial delay) on outgoing calls—Defined as “alert received time–setup out time.”
- Total setup delay duration—Defined as “setup out time—setup in time.”
- IP-specific statistic fields (exist only in the IP-level statistics):
  - Number of calls losing more than the configured number of packets—The default is 1000.
  - Number of calls encountering more than the configured amount of latency—The default is 250 milliseconds.
  - Number of calls encountering more than the configured amount of jitter—The default is 250 milliseconds.
  - Number of incoming and outgoing calls disconnected with each cause code—The cause codes are defined in the Call Control Application Programming Interface (CCAPI) and in the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) standard Q.850.

What Are IECs?

Cisco IOS generated internal error codes (IECs) are gateway-detected errors that cause the gateway to release or refuse a call. IECs enhance troubleshooting for VoIP networks by helping to determine the source and reason for call termination.

Management of the Statistical Collection

The management of statistical collection involves time ranges, thresholds, storage capacities, and memory usage. This section has the following subsections:

What Are the Allowable Time Ranges?

The Voice Performance Statistics on Cisco Gateways feature enables you to configure time ranges to capture statistics. The time ranges are as follows:

- From the last reset time to the present. You can examine the statistics using the **show voice statistics** command and reset the statistics using the **clear voice statistics csr** command.
- By specific start and end time. That is, a set amount of minutes after the configuration time for preparation of the resource allocation of the gateway.
- By periodic intervals, with an optional total duration. Allowable intervals are 5 minutes, 15 minutes, 30 minutes, 1 hour, and 1 day. The optional total duration is unlimited but must be a multiple of the specified interval. For example, the interval could be 15 minutes, and the total duration could be 3 hours.

What Are Thresholds?

The following three thresholds can be configured to customize how you measure the voice signaling statistics for your network configuration:

- Packet jitter
- Packet latency
- Lost packets

These thresholds are all pre-configured with default settings. The jitter, latency, and lost packets thresholds only apply to IP statistics. In addition, you can configure the minimum call duration (MCD) value for determining which calls are measured during the statistics collection.

Packet Jitter

Jitter is a variation in the delay of received packets. At the sending side, packets are sent in a continuous stream, spaced evenly apart. Because of network congestion, improper queueing, or configuration errors, this steady stream can be interrupted by delays between packets.

You can specify the threshold at which a record will not be collected. For example, if you have set a threshold of 250 milliseconds and a delay exceeds that threshold, the message is not collected.

Packet Latency

Packet latency is the amount of time that it takes a packet to go from its source to its destination. You can specify the threshold at which a record will not be collected. For example, the gateway can be configured to drop messages that take more than 250 milliseconds to reach the destination.

Lost Packets

Lost packets are a result of jitter that is so great that it causes packets to be out of the range of the jitter buffer. These packets are discarded. You can specify the threshold for lost packets in milliseconds.

Minimum Call Duration

Using the minimum call duration (MCD) value, you can configure the gateway to collect statistics for calls that last a minimum amount of time. For example, if you configure the MCD value to 2 milliseconds, the gateway counts the number of incoming or outgoing calls with a connect time less than 2 milliseconds.

What Are the Allowable Storage Capacities?

The Voice Performance Statistics on Cisco Gateways feature allows you to specify how long the gateway will store the statistics. You can specify a number of days, hours, or minutes.

How Is Memory Used?

You can display how much system memory is being used when statistics are being collected, how much system memory is available, and how much system memory is estimated for future use by using the **show voice statistics memory-usage** command. The command displays all memory usage for accounting and signaling by fixed interval and since a system reset or reboot. The output includes the number of call statistics records per interval, each record size, the approximate memory used, and the allocated memory for future use.

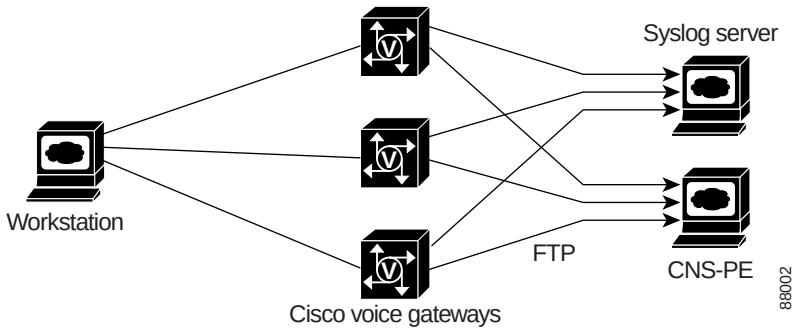
Management of the Archive Process

The FTP or syslog server archive process is used to download statistics to a file on the server for further processing, presentation, and analysis. The download to the server is configured using the command-line interface (CLI). When the specified end time of collection is reached, the gateway downloads the statistics data to the provisioned server. Using FTP, the gateway formats the statistics in an ASCII file and transfers the file to a Cisco Networking Services Performance Engine (CNS-PE). Using the syslog server, gateways send the information in syslog messages, and you can specify the message size.

Note Because of unreliable UDP transport, the integrity and completeness of the call statistics in the syslog files are not guaranteed.

Figure 56 shows the components as used by this feature.

Figure 56 Syslog and FTP Servers and the CNS-PE



You can format the output to display with specified record separators. The separator can be a space, tab, new line, or ASCII character.

Display of Records and Time Ranges

You can display statistics by accounting method lists, by aggregation levels, by numbered intervals (time ranges), and/or since the last system reset or reboot. This section has the following subsections:

- What Records Are Displayed Since System Reset or Reboot?, page 7
- What Time Ranges Are Displayed?, page 7

What Records Are Displayed Since System Reset or Reboot?

You can display the collected records since a system reset or reboot including accounting and/or signaling (aggregation level), and intervals (time ranges). There are show commands to display all signaling statistics that were collected since the last reset or reboot of the system. You can specify a concise or verbose display, and you can configure the gateway to push the statistics to the FTP or syslog server.

Displaying Accounting Statistics

You can use the **show voice statistics csr since-reset accounting** command to display the method list by RADIUS server, PSTN incoming and outgoing records that have passed and failed, and the IP incoming and outgoing records that have passed and failed.

Displaying Aggregation-Level Statistics

You can use the **show voice statistics csr since-reset aggregation-level** command to display all the collected statistics for every aggregation level or just the statistics for a specific level (gateway, IP, PSTN, trunk group, or voice port).

What Time Ranges Are Displayed?

You can use the **show voice statistics interval-tag** command to display the start and end times by a numbered interval. To display the time ranges by voice port, you can use the **show voice statistics csr interval aggregation** command.

Voice Interface Changes During Call-Statistics Collection Periods

It is recommended that you do not change the voice interface configuration during the period when call statistics are being collected. It is also recommended that during the period when call statistics are being collected, the following configuration changes not take place:

- Adding or removing a PRI or DS1 group
- Adding or removing a trunk group
- Adding or removing a trunk in a trunk group

Note It is recommended that any existing call-statistics collection be stopped and set to zero before any configuration modification is made to any of the voice interfaces.

This section has the following subsections:

- Addition or Removal of a Voice Port, page 8
- Configuration Change of Any Trunk Group, page 8

Addition or Removal of a Voice Port

The following scenarios apply to a voice port whether or not it is in any trunk group:

- The addition of a PRI or DS1 group, Foreign Exchange Station (FXS), Foreign Exchange Office (FXO), or ear and mouth (E&M) device.
- The removal of a PRI or DS1 group, FXS, FXO, or E&M device.

If a new voice interface is added during any collection period, new entries that correspond to the new voice interface are added in the statistics collected for that collection period.

If an existing voice interface is removed during any collection period, the statistics that correspond to that voice interface are still kept in the set. The statistics are frozen (that is, nothing more is added) after the time that the voice interface is removed. Gateways still send the statistics for the removed interface to the CNS-PE or syslog server.

In either of the above scenarios, the statistics for PSTN ports include all the collected data at the DS1 or channel associated signaling (CAS) level, except the statistics for interfaces added or removed during the collection period.

Configuration Change of Any Trunk Group

The following scenarios apply when there is a configuration change of any trunk group:

- The addition or removal of a DS1 or CAS group into or from a trunk group.
  - If you are adding a DS1 or CAS group to a trunk group during any collection period, the collection process moves the associated statistics from the upper aggregation level (PSTN) to the trunk-group level. The call statistics before the configuration change time are also totalled to the statistics of the trunk-group level at the end of the collection period. In this case, the statistics of the trunk-group level can exceed the limit. However, the PSTN-level statistic is still accurate.
  - If you are removing a DS1 or CAS group from a trunk group during any collection period, the collection process removes the trunk-group level of the PSTN aggregation. In contrast to the scenario of adding a DS1 or CAS group, the statistic of the trunk-group level is under the limit.

- The addition or removal of a trunk group.
  - If you are adding a new trunk group (one that contains a trunk or trunks) to a gateway during any collection period, the existing statistics of all member trunks aggregate to the trunk-group level statistic. That is, the statistic of the trunk-group level is over its limit. The PSTN-level call statistics are still accurate.
  - If you are removing an existing trunk group from a gateway during any collection period, the existing statistics of all member trunks are totaled at the configuration change time. The statistics of the trunk-group level and of the PSTN level are accurate.

Note The inaccurate call statistics that can result from the four scenarios that are listed above are acceptable because the transient information during the configuration change is often unusable.

Benefits of Voice Performance Statistics on Cisco Gateways

- Gateway call statistics can be audited or compared against the statistics of other network devices for improved monitoring.
- Malfunctioning DSPs (DS1 only) can be discovered.
- Discrepancies between RADIUS records sent by the gateway and received and reported on the server can be uncovered.
- Potential lost revenue can be highlighted.
- Call-success rates and accuracy of reports can be determined.

Configuring Voice Performance Statistics on Cisco Gateways

This section contains the following procedures:

Note If you need to obtain statistical information since reboot, the configuration should be stored in NVRAM before you restart the gateway.

Summary of Configuration Tasks

There are two general areas of configuration—one for the collection of signaling statistics and one for the collection of accounting statistics. The tasks required will depend on whether you configure the gateway to collect signaling statistics only, accounting statistics only, or both signaling and accounting statistics.

Required Task for Collection of All Call Statistics

Step 1 Configure the duration or time period for when call statistics are collected on the gateway. See the "Configuring the Duration and Time Periods of Call Statistics on the Gateway" section on page 11.

Required Task for Signaling Statistics

Step 1 Configure the gateway to support the collection of signaling statistics. See the "Enabling the Gateway to Collect Signaling Statistics" section on page 14.

Optional Task for Signaling Statistics

- Configure the call statistics record signaling parameters, changing the default values as needed. See the "Configuring the Minimum Call Duration and Signaling Thresholds" section on page 16

Required Tasks for Accounting Statistics (Configured in This Order)

Step 1 Configure the gateway to support the collection of accounting statistics. See the "Enabling the Collection of VoIP AAA Accounting Statistics on the Gateway" section on page 25.

Step 2 Configure accounting on the gateway. Refer to the **gw-accounting aaa** command configuration in the *Cisco IOS Voice Configuration Library*, Release 12.3.

Step 3 Specify that the accounting update is new information. Refer to the **aaa accounting update new-info** command configuration in the *Cisco IOS Security Configuration Guide*, Release 12.3, and the *Cisco IOS Security Command Reference*, Release 12.3 T.

Step 4 Define the AAA RADIUS server group. Refer to the **aaa group server radius** command configuration in the *Cisco IOS Security Configuration Guide*, Release 12.3, and the *Cisco IOS Security Command Reference*, Release 12.3 T.

Step 5 Define a designated broadcast accounting server group (**accounting acknowledge broadcast** command). See the "Configuring a Designated Server Group for a Broadcast Method List" section on page 27.

Step 6 Define the RADIUS server host, port, key, and vendor specific attributes (VSAs). Refer to the *Cisco IOS Security Configuration Guide*, Release 12.3 and the *Cisco IOS Security Command Reference*, Release 12.3 T.

Optional Tasks for Both Signaling and Accounting Statistics (Configured in Any Order)

- Configure the FTP server or syslog server download. See the "Configuring the Gateway to Archive Statistics to an FTP Server" section on page 37 and the "Configuring the Gateway to Archive Statistics to a Syslog Server" section on page 38.

Note Before configuring the gateway to archive statistics to an FTP server, you must first configure the FTP server to support the archiving process. See the "Configuring the FTP Server to Enable Archiving of Statistics from the Gateway" section on page 34.

Configuring the Duration and Time Periods of Call Statistics on the Gateway

Before you configure the gateway to collect call signaling statistics, VoIP AAA accounting statistics, or Cisco VoIP internal error codes (IECs), you must first configure the duration and time periods for when the call statistics are collected. There are three methods for collecting call statistics: periodic, since the last reset, and for specific times.

Note These interval methods are mutually exclusive, meaning that the gateway can be configured for only one collection interval at a time. The collection interval configured applies to all call statistics collected. For example, if you configure the collection interval for a periodic interval, and you configure the gateway to collect both signaling and VoIP AAA accounting statistics, then both types of statistics will be collected on the periodic basis.

To configure the duration and time period of when call statistics will be collected on the gateway, see one of the following sections:

- Configuring the Gateway to Collect Call Statistics on a Periodic Basis, page 11
- Configuring the Gateway to Collect Call Statistics Since the Last Reset, page 12
- Configuring the Gateway to Collect Call Statistics for a Specific Time Interval, page 13

Configuring the Gateway to Collect Call Statistics on a Periodic Basis

This task configures the gateway to collect call statistics on a periodic basis.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics time-range periodic** *interval* **start** *hh*:*mm* {**days-of-week** *{***Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** | **daily** | **weekdays** | **weekend***}} [***end** *hh*:*mm* *{***days-of-week** *{***Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday***}}]*
4. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics time-range periodic** *interval* **start** *hh:mm* {**days-of-week** *{***Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** | **daily** | **weekdays** | **weekend***}} [***end** *hh:mm {***days-of-week** *{***Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday***}}]* **Example:** **Router(config)# voice statistics time-range periodic 60minutes start 12:00 days-of-week Monday end 12:00 days-of-week Friday**	Configures the gateway to collect call statistics on a periodic basis.

Step 4	`exit`
	
	Example:
	`Router(config)# exit`
		Exits global configuration mode.

Configuring the Gateway to Collect Call Statistics Since the Last Reset

This task configures the gateway to collect call statistics since the last time the **clear voice statistics** command was entered, or since the last time the gateway was rebooted.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics time-range since-reset**
4. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics time-range since-reset** **Example:** **`Router(config)# voice statistics time-range since-reset`**	Configures the gateway to collect call statistics since the last reset or since the last time the gateway was rebooted.

Step 4	`exit`
	
	Example:
	`Router(config)# exit`
		Exits global configuration mode.

Configuring the Gateway to Collect Call Statistics for a Specific Time Interval

This task configures the gateway to collect call statistics for a specific time interval.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics time-range specific start** *hh*:*mm day month year* **end** *hh*:*mm day month year*
4. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**voice statistics time-range specific start** *hh:mm day month year* **end** *hh:mm day month year* **Example:** **Router(config)# voice statistics time-range specific start 10:00 1 January 2004 end 12:00 2 January 2004**	Configures the gateway to collect call statistics for a specific configured time period.
Step 4	exit **Example:** Router(config)# exit	Exits global configuration mode.

Configuring the Gateway to Collect Signaling Statistics

This section describes how to configure the gateway to collect signaling statistics. This section documents the following tasks:

- Enabling the Gateway to Collect Signaling Statistics, page 14
- Configuring the Minimum Call Duration and Signaling Thresholds, page 16
- Disabling the Collection of Signaling Statistics, page 17
- Displaying the Signaling Statistics for Each Aggregation Level, page 18
- Clearing Signaling Statistics, page 24

Enabling the Gateway to Collect Signaling Statistics

This task describes to how to enable the gateway to collect signaling statistics.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics type csr signaling**
4. **voice statistics max-storage-duration {day** *value* **| hour** *value* **| minute** *value***}**
5. **voice statistics display-format separator {space | tab | new-line | char** ***char*****}**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics type csr signaling** **Example:** `Router(config)# voice statistics type csr signaling`	Enables the collection of signaling statistics. **Note** To enable the collection of both signaling and VoIP AAA accounting statistics on the gateway, enter the command without the **signaling** keyword as follows: **voice statistics type csr**.
Step 4	**voice statistics max-storage-duration {day** *value* **| hour** *value* **| minute** *value***}** **Example:** `Router(config)# voice statistics max-storage-duration minute 60`	(Optional) Configures the maximum storage time in system memory of the gateway. The keywords and argument are as follows: • **day**—Number of days. The *value* argument has a valid range from 0 to 365. • **hour**—Number of hours. The *value* argument has a valid range from 0 to 720. • **minute**—Number of minutes. The *value* argument has a valid range from 0 to 1440. **Note** This command also applies to the collection of VoIP internal error codes (IECs).
Step 5	**voice statistics display-format separator {space | tab | new-line | char** ***char*****}** **Example:** `Router(config)# voice statistics display-format separator new-line`	(Optional) Specifies the way that displayed information is separated. The default is a comma (,).

Step 6	`exit`
	
	Example:
	`Router(config)# exit`
		Exits global configuration mode.

Configuring the Minimum Call Duration and Signaling Thresholds

The signaling parameters include minimum call duration, as well as thresholds for lost packet information, packet latency information, and packet jitter information. These parameters have default values so no configuration is required. However, you can customize any or all of these parameters as needed for your network configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics field-params mcd** *value*
4. **voice statistics field-params lost-packet** *value*
5. **voice statistics field-params packet-latency** *value*
6. **voice statistics field-params packet-jitter** *value*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics field-params mcd** ***value*** **Example:** `Router(config)# voice statistics field-params mcd 10`	Configures the minimum call duration (MCD) for collecting voice call statistics in milliseconds. Valid values are from 0 to 30 milliseconds. The default is 2 milliseconds. This value applies to both IP and PSTN statistics.
Step 4	**voice statistics field-params lost-packet** ***value*** **Example:** `Router(config)# voice statistics field-params lost-packet 5000`	Configures the lost voice packet threshold for collecting voice call statistics in milliseconds. Valid values are from *0 to 65535* milliseconds. The default is 1000 milliseconds. This value applies to IP statistics only.
Step 5	**voice statistics field-params packet-latency** ***value*** **Example:** `Router(config)# voice statistics field-params packet-latency 200`	Configures the voice packet-latency threshold parameter for voice call statistics in milliseconds. Valid values are from *0 to 500 milliseconds.* The default is 250 milliseconds. This value applies to IP statistics only.

	Command or Action	Purpose
Step 6	**voice statistics field-params packet-jitter** ***value*** **Example:** `Router(config)# voice statistics field-params packet-jitter 500`	Configures the voice packet-jitter threshold parameter for voice call statistics in milliseconds. Valid values are from *0 to 1000 milliseconds.* The default is 250 milliseconds. This value applies to IP statistics only.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Disabling the Collection of Signaling Statistics

This task disables the collection of signaling statistics.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no voice statistics type csr signaling**
4. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**no voice statistics type csr signaling** **Example:** `Router(config)# no voice statistics type csr signaling`	Disables the collection of signaling statistics. **Note** If the gateway is configured to collect both signaling and VoIP AAA accounting statistics, the accounting statistics will continue to be collected after the signaling statistics collection is disabled.
Step 4	exit **Example:** `Router(config)# exit`	Exits global configuration mode.

Displaying the Signaling Statistics for Each Aggregation Level

Signaling statistics can be collected and displayed for all aggregation levels. To display signaling statistics for the different aggregation levels, see the following sections:

- Displaying Signaling Statistics for All Aggregation Levels, page 18
- Displaying Gateway-Level Signaling Statistics, page 19
- Displaying VoIP-Level Signaling Statistics, page 20
- Displaying PSTN-Level Signaling Statistics, page 21
- Displaying Trunk-Group Level Signaling Statistics, page 22
- Displaying Voice-Port Level Signaling Statistics, page 23

All commands in this section are entered in privileged EXEC mode. The statistics displayed are based on the time range configured using the **voice statistics time-range** command. For example, if you set the time range to specify that the gateway collects statistics only since the last reset, then these displays show only the statistics since the gateway was last reset or rebooted.

With these commands, you can specify that the display shows either verbose or concise information. The verbose display shows all fields contained in the call statistics records, while the concise display shows only output that contains total calls, answered calls, and answered call duration. The verbose display mode is enabled by default.

In addition, you can specify that the gateway push the statistics display from the console to an FTP or syslog server. To configure the gateway to support pushing statistics to an FTP or syslog server, see the "Managing the Collection of Voice Statistics" section on page 34.

Displaying Signaling Statistics for All Aggregation Levels

This task displays signaling statistics for all aggregation levels.

SUMMARY STEPS

1. **enable**
2. **show voice statistics interval-tag**
3. **show voice statistics csr interval** *tag-number* **aggregation all [mode {concise | verbose}] [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset aggregation-level all [mode {concise | verbose**}] [**push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics interval-tag** **Example:** `Router# show voice statistics interval-tag`	Displays the configured interval numbers. This command is necessary to obtain the tag number required in the next step.
Step 3	**show voice statistics csr interval** *tag-number* **aggregation all [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr interval 102 aggregation all`	Displays signaling statistics for all aggregation levels for a given interval. **Note** This command is valid only if the **voice statistics time-range** command is configured to support either periodic statistics collection or statistics collection for a specific time period.
Step 4	**show voice statistics csr since-reset aggregation-level all [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr since-reset aggregation-level all`	Displays signaling statistics for all aggregation levels since the last reset or reboot of the gateway. **Note** This command is valid only if the **voice statistics time-range** command is configured to the **since-reset** value.

Displaying Gateway-Level Signaling Statistics

This task displays gateway-level signaling statistics.

SUMMARY STEPS

1. **enable**
2. **show voice statistics interval-tag**
3. **show voice statistics csr interval** *tag-number* **aggregation gateway [mode {concise | verbose}] [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset aggregation-level gateway [mode {concise | verbose}] [push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics interval-tag** **Example:** `Router# show voice statistics interval-tag`	Displays the configured interval numbers. This command is necessary to obtain the tag number required in the next step.
Step 3	**show voice statistics csr interval** *tag-number* **aggregation gateway [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr interval 102 aggregation gateway`	Displays the gateway-wide level statistics for a given interval. **Note** This command is valid only if the **voice statistics time-range** command is configured to support either periodic statistics collection or statistics collection for a specific time period.
Step 4	**show voice statistics csr since-reset aggregation-level gateway [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr since-reset aggregation-level gateway`	Displays the gateway-wide level statistics since the last reset or reboot of the gateway. **Note** This command is valid only if the **voice statistics time-range** command is configured to the **since-reset** value.

Displaying VoIP-Level Signaling Statistics

This task displays VoIP-level signaling statistics.

SUMMARY STEPS

1. **enable**
2. **show voice statistics interval-tag**
3. **show voice statistics csr interval** *tag-number* **aggregation ip [mode {concise | verbose}] [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset aggregation-level ip [mode {concise | verbose}] [push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics interval-tag** **Example:** `Router# show voice statistics interval-tag`	Displays the configured interval numbers. This command is necessary to obtain the tag number required in the next step.
Step 3	**show voice statistics csr interval** *tag-number* **aggregation ip [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr interval 102 aggregation ip`	Displays the VoIP interface-level statistics for a given interval. **Note** This command is valid only if the **voice statistics time-range** command is configured to support either periodic statistics collection or statistics collection for a specific time period.
Step 4	**show voice statistics csr since-reset aggregation-level ip [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr since-reset aggregation-level ip`	Displays the VoIP-level statistics since the last reset or reboot of the gateway. **Note** This command is valid only if the **voice statistics time-range** command is configured to the **since-reset** value.

Displaying PSTN-Level Signaling Statistics

This task displays PSTN-level signaling statistics.

SUMMARY STEPS

1. **enable**
2. **show voice statistics interval-tag**
3. **show voice statistics csr interval** *tag-number* **aggregation pstn [mode {concise | verbose}] [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset aggregation-level pstn [mode {concise | verbose}] [push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics interval-tag** **Example:** `Router# show voice statistics interval-tag`	Displays the configured interval numbers. • This command is necessary to obtain the tag number required in the next step.
Step 3	**show voice statistics csr interval** *tag-number* **aggregation pstn [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr interval 102 aggregation pstn`	Displays the telephone interface level statistics for a given interval. **Note** This command is valid only if the **voice statistics time-range** command is configured to support either periodic statistics collection or statistics collection for a specific time period.
Step 4	**show voice statistics csr since-reset aggregation-level all pstn [mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr since-reset aggregation-level pstn`	Displays the PSTN-level statistics since the last reset or reboot of the gateway. **Note** This command is valid only if the **voice statistics time-range** command is configured to the **since-reset** value.

Displaying Trunk-Group Level Signaling Statistics

This task displays trunk-group level signaling statistics.

SUMMARY STEPS

1. **enable**
2. **show voice statistics interval-tag**
3. **show voice statistics csr interval** *tag-number* **aggregation trunk-group** {**all** | *trunk-group-label*} **[mode {concise | verbose}] [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset aggregation-level trunk-group** {**all** | *trunk-group-label*} **[mode {concise | verbose}] [push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics interval-tag** **Example:** `Router# show voice statistics interval-tag`	Displays the configured interval numbers. This command is necessary to obtain the tag number required in the next step.
Step 3	**show voice statistics csr interval** *tag-number* **aggregation trunk-group** {**all** | *trunk-group-label*} **[mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr interval 102 aggregation trunk-group 20`	Displays the trunk-group level statistics for a given interval. Display statistics can be specified for a single trunk group or for all trunk groups. **Note** This command is valid only if the **voice statistics time-range** command is configured to support either periodic statistics collection or statistics collection for a specific time period.
Step 4	**show voice statistics csr since-reset aggregation-level trunk-group** {**all** | *trunk-group-label*} **[mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr since-reset aggregation-level trunk-group all`	Displays the trunk-group level statistics since the last reset or reboot of the gateway. You can display statistics for a specific trunk group or for all trunk groups. **Note** This command is valid only if the **voice statistics time-range** command is configured to the **since-reset** value.

Displaying Voice-Port Level Signaling Statistics

This task displays voice-port level signaling statistics.

SUMMARY STEPS

1. **enable**
2. **show voice statistics interval-tag**
3. **show voice statistics csr interval** *tag-number* **aggregation voice-port** {*voice-port-label* | **all**} **[mode {concise | verbose}] [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset aggregation-level voice-port** {**all** | *voice-port-label*} **[mode {concise | verbose}] [push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics interval-tag** **Example:** `Router# show voice statistics interval-tag`	Displays the configured interval numbers. This command is necessary to obtain the tag number required in the next step.
Step 3	**show voice statistics csr interval** *tag-number* **aggregation voice-port** {*voice-port-label* **| all**} **[mode {concise | verbose}] [push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr interval 102 aggregation voice-port all`	Displays voice-port level statistics for a given interval. You can display statistics for a specific voice port or for all voice ports. **Note** This command is valid only if the **voice statistics time-range** command is configured to support either periodic statistics collection or statistics collection for a specific time period.
Step 4	**show voice statistics csr since-reset aggregation-level voice-port** {**all** | *voice-port-label*} **[mode {concise | verbose}] [push {all | ftp | syslog}** **Example:** `Router# show voice statistics csr since-reset aggregation-level voice-port all`	Displays the voice-port level statistics since the last reset or reboot of the gateway. You can display statistics for a specific voice port or for all voice ports. **Note** This command is valid only if the **voice statistics time-range** command is configured to the **since-reset** value.

Clearing Signaling Statistics

This task clears signaling statistics from the gateway.

SUMMARY STEPS

1. **enable**
2. **clear voice statistics csr signaling**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`clear voice statistics csr signaling`
	
	Example:
	`Router# clear voice statistics csr signaling`
		Clears all signaling statistics.

Configuring the Gateway to Collect VoIP AAA Accounting Statistics

Using this feature, statistics can be collected to tally accounting records on billing servers (RADIUS) and on voice gateways based on call legs (both inbound and outbound). The statistics collected track whether calls were successfully accounted for based on the acknowledgement messages from designated billing or accounting servers.

This section shows you how to configure the collection of accounting statistics on the gateway. This section documents the following tasks:

Prerequisites

The definition of the AAA method list for accounting, the server groups, and the RADIUS servers should be configured. For more information, refer to the *Configuring AAA for Cisco Voice Gateways* document in the Cisco IOS Voice Configuration Library.

Restrictions

You can define “pass” criteria for calls on the basis of method lists but not on the basis of server groups. For broadcast method lists, if the gateway attempts to access multiple server groups simultaneously, additional configuration is needed. See the “Configuring a Designated Server Group for a Broadcast Method List” section on page 27.

Enabling the Collection of VoIP AAA Accounting Statistics on the Gateway

This task enables the collection of accounting statistics on the gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics type csr accounting**
4. **voice statistics accounting method** *method-list-name* **pass** {**start-interim-stop** | **start-stop** | **stop-only**}
5. **voice statistics max-storage-duration {day** *value* **| hour** *value* **| minute** *value***}**
6. **voice statistics display-format separator {space | tab | new-line | char** ***char*****}**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics type csr accounting** **Example:** `Router(config)# voice statistics type csr accounting`	Enables the collection of VoIP AAA accounting statistics. **Note** To enable the collection of both accounting and signaling statistics on the gateway, enter the command without the **accounting** keyword, as follows: **voice statistics type csr**.
Step 4	**voice statistics accounting method** *method-list-name* **pass {start-interim-stop | start-stop | stop-only}** **Example:** `Router(config)# voice statistics accounting method h323 pass stop-only`	Configures the method-list pass criteria. **Note** The *method-list-name* argument is the same as that configured using the **method** command in gateway-accounting AAA configuration mode. You can have multiple method lists configured on a gateway at one time.

	Command or Action	Purpose
Step 5	**voice statistics max-storage-duration {day** *value* **| hour** *value* **| minute** *value***}** **Example:** `Router(config)# voice statistics max-storage-duration minute 60`	(Optional) Configures the maximum storage time in system memory of the gateway. The keywords and argument are as follows: • **day**—Number of days. The *value* argument has a valid range from 0 to 365. • **hour**—Number of hours. The *value* argument has a valid range from 0 to 720. • **minute**—Number of minutes. The *value* argument has a valid range from 0 to 1440. **Note** This command also applies to the collection of VoIP internal error codes (IECs).
Step 6	**voice statistics display-format separator {space | tab | new-line | char** ***char*****}** **Example:** `Router(config)# voice statistics display-format separator new-line`	(Optional) Specifies the way in which displayed information is separated. The default is a comma (,).

Step 7	`exit`
	
	Example:
	`Router(config)# exit`
		Exits global configuration mode.

Configuring a Designated Server Group for a Broadcast Method List

This task is required if accounting CSRs need to be collected for a gateway that is configured for a broadcast method list. See the "Configuring the Gateway to Collect VoIP AAA Accounting Statistics" section on page 25. It is possible to set all the server groups in a method list for monitoring acknowledgements.

Prerequisites

The collection of accounting statistics should be enabled on the gateway. The pass criteria for the method list must already be defined.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **aaa group server radius** *name*
4. **accounting acknowledge broadcast**
5. end
6. show voice statistics | begin aaa group server

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**aaa group server radius** ***name*** **Example:** `Router(config)# aaa group server radius billing-grp`	Groups different RADIUS server hosts into distinct lists and distinct methods and enters server group RADIUS configuration mode.
Step 4	**accounting acknowledge broadcast** **Example:** `Router(config-sg-radius)# accounting acknowledge broadcast`	Enables the accounting broadcast functionality for the server groups configured in Step 3.

Step 5	`end`
	
	Example:
	`Router(config-sg-radius)# end`
		Ends the current configuration session and returns to privileged EXEC mode.

Step 6	`show voice statistics | begin aaa group server`
	
	Example:
	`Router# show voice statistics | begin aaa group server`
		Displays the AAA group server and its configuration.

Troubleshooting Tips

Acknowledgements of only designated server groups are considered when deciding whether the accounting for a given call leg is successful. If more than one server group is configured as designated, the gateway considers the response from all server groups in deciding whether the call leg accounting is successful.

Disabling the Collection of VoIP AAA Accounting Statistics

This task disables the collection of VoIP AAA accounting statistics.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no voice statistics type csr accounting**

4. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**no voice statistics type csr accounting** **Example:** `Router(config)# no voice statistics type csr accounting`	Disables the collection of accounting statistics. **Note** If the gateway is configured to collect both VoIP AAA accounting statistics and signaling statistics, the signaling statistics will continue to be collected after the accounting statistics collection is disabled.

Step 4	`exit`
	
	Example:
	`Router(config)# exit`
		Exits global configuration mode.

Displaying the VoIP AAA Accounting Statistics

After the gateway has been configured to collect VoIP AAA accounting statistics, you can display all accounting statistics or accounting statistics for a specific method list.

All commands in this section are entered in privileged EXEC mode. The statistics displayed are based on the time range configured using the **voice statistics time-range** command. For example, if you set the time range to specify that the gateway to collect statistics only since the last reset, then these displays will show only the statistics since the gateway was last reset or rebooted.

With these commands, you can specify that the gateway push the statistics display from the console to an FTP or syslog server. To configure the gateway to support pushing statistics to an FTP or syslog server, see the "Managing the Collection of Voice Statistics" section on page 34.

SUMMARY STEPS

1. **enable**
2. **show voice accounting method** [*method-list-name*]
3. **show voice statistics csr interval** *tag-number* **accounting {all | method-list** *method-list-name***} [push {all | ftp | syslog}]**
4. **show voice statistics csr since-reset accounting {all | method-list *method-list-name*} [push {all | ftp | syslog}]**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice accounting method** {*method-list-name*} **Example:** `Router# show voice accounting method mll`	Displays connectivity status information for accounting method lists.
Step 3	**show voice statistics csr interval** *tag-number* **accounting {all | method-list** *method-list-name*} **[push {all | ftp | syslog**}**]** **Example:** `Router# show voice statistics csr interval 10 accounting all`	Displays the VoIP AAA accounting statistics for the specified interval. You can display all accounting statistics or accounting statistics for a specific method list.
Step 4	**show voice statistics csr since-reset accounting {all | method-list** ***method-list-name***} **[push {all | ftp | syslog}]** **Example:** `Router# show voice statistics csr since-reset accounting method-list h323`	Displays all VoIP AAA accounting statistics since the last reset or reboot of the gateway. You can display all accounting statistics or accounting statistics for a specific method list.

Step 5s	`show voice accounting response pending`
	
	Example:
	`Router# show voice accounting response pending`
		Displays information regarding pending VoIP AAA accounting responses.

Clearing the VoIP AAA Accounting Statistics

This task describes how to clear the VoIP AAA accounting statistics.

SUMMARY STEPS

1. **enable**
2. **clear voice statistics csr accounting**
3. **clear voice accounting method** *method-list-name*

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`clear voice statistics csr accounting`
	
	Example:
	`Router# clear voice statistics csr accounting`
		Clears all accounting statistics.

Step 3	**`clear voice accounting method`** *`method-list-name`* **Example:** `Router# clear voice accounting method h323`	Clears accounting statistics for a specific accounting method.

Troubleshooting Tips for VoIP AAA Accounting Statistics

If any of the following messages is displayed, proceed with the solution shown.

- Message: “Either specific or periodic time-range has existed.”

  Solution: Delete any previously configured specific or periodic time range.

- Message: “Ending time must be greater than starting time.”

  Solution: Make sure that the configured end time is later than the start time for the time range configured.

- Message: “Starting time must be future time.”

  Solution: Make sure that the configured start time is in the future.

If accounting statistics are not being collected properly, the best place to start troubleshooting is by verifying your configurations as follows:

```
Router# show running-config | include voice statistics

voice statistics type csr
voice statistics accounting method h323 pass stop-only
! Specifies the pass criteria.
voice statistics time-range since-reset

Router# show running-config | include aaa

aaa new-model
aaa accounting connection h323 start-stop group radius
! Specifies the method list.
aaa session-id common
gw-accounting aaa
! Enables accounting on the gateway.
radius-server host 1.6.10.203 auth-port 1645 acct-port 1646
! Specifies RADIUS server IP address.
```

If your configurations are correct, turn on debugging with the following **debug** commands and check the output. You must turn on all three commands:

- **debug isdn q931**
- **debug radius accounting**
- **debug voip-aaa**

As long as accounting and signaling statistics are being collected, the output from these three debug commands will display on the console.

The following sample output shows that the accounting statistics are not being properly collected by the RADIUS server:

```
*Nov 22 14:54:49.350: ISDN Se6/0:15 Q931: RX <- SETUP pd = 8  callref = 0x125E
        Bearer Capability i = 0x8090A3
                Standard = CCITT
                Transfer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Called Party Number i = 0x80, '11'
                Plan:Unknown, Type:Unknown
!
Router#55:02.366: RADIUS: acct-delay-time for 4067CADC (at 4067CDF6) now 10
*Nov 22 14:55:04.366: RADIUS: Retransmit to (10.6.10.203:1645,1646) for id 21669/117
*Nov 22 14:55:04.366: RADIUS(0000053B): Retransmit id 21669/117
*Nov 22 14:55:04.366: RADIUS: acct-delay-time for 40278B3C (at 40278D04) now 15
*Nov 22 14:55:07.366: RADIUS: Retransmit to (10.6.10.203:1645,1646) for id 21669/118
*Nov 22 14:55:07.366: RADIUS(0000053B): Retransmit id 21669/118
*Nov 22 14:55:07.366: RADIUS: acct-delay-time for 4067CADC (at 4067CDF6) now 15
*Nov 22 14:55:09.366: RADIUS: Tried all servers.
*Nov 22 14:55:09.366: RADIUS: No valid server found. Trying any viable server.
```

The line above indicates that the RADIUS server is unreachable.

```
*Nov 22 14:55:09.366: RADIUS: Tried all servers.
*Nov 22 14:55:09.366: RADIUS: No response from (10.6.10.203:1645,1646) for id 21669/119
*Nov 22 14:55:09.366: RADIUS/DECODE: parse response no app start; FAIL
*Nov 22 14:55:09.366: RADIUS/DECODE: parse response; FAIL
*Nov 22 14:55:12.366: RADIUS: Tried all servers.
*Nov 22 14:55:12.366: RADIUS: No valid server found. Trying any viable server.
```

The line above indicates that the RADIUS server is not found.

```
*Nov 22 14:55:12.366: RADIUS: Tried all servers.
*Nov 22 14:55:12.366: RADIUS: No response from (10.6.10.203:1645,1646) for id 21669/120
*Nov 22 14:55:12.366: RADIUS/DECODE: parse response no app start; FAIL
*Nov 22 14:55:12.366: RADIUS/DECODE: parse response; FAIL
*Nov 22 14:55:12.366: voip_process_acct_reply(1339):
*Nov 22 14:55:12.366: voip_process_acct_reply(1339): event_message->type=0x210,
                      msg_t=2, rsp=2
*Nov 22 14:55:12.366: voip_process_acct_reply: acct notification call back for
                      method=h323
*Nov 22 14:55:12.366: acct_notif_cleanup:
*Nov 22 14:55:12.366: acct_ntf_deregistration:
*Nov 22 14:55:12.366: acct_ntf_deregistration: deregister with AAA EM, rsp_t=0x4,
                      ev_t=0x210
*Nov 22 14:55:12.366: acct_notif_cleanup: unlock adb
*Nov 22 14:55:12.366: voip_aaa_unlock_adb: uid(1339) count=0
*Nov 22 14:55:12.366: voip_aaa_cleanup_adb: dealloc uid (1339)
*Nov 22 14:55:12.366: voip_aaa_acct_get_dynamic_attrs: No cdb found from cdb tree
```

The following sample output shows that the RADIUS server was authenticated:

```
*Nov 22 14:56:46.006: ISDN Se6/0:15 Q931: RX <- SETUP pd = 8  callref = 0x125F
        Bearer Capability i = 0x8090A3
                Standard = CCITT
                Transfer Capability = Speech
                Transfer Mode = Circuit
                Transfer Rate = 64 kbit/s
        Channel ID i = 0xA98381
                Exclusive, Channel 1
        Called Party Number i = 0x80, '11'
                Plan:Unknown, Type:Unknown
*Nov 22 14:56:46.010: voip_start_ccapi_accounting(784):
*Nov 22 14:56:46.010: voip_start_accounting_internal:
*Nov 22 14:56:46.010: voip_start_accounting_internal(784): peer_tag=100
*Nov 22 14:56:46.010: get_acct_params: suppressed=0
*Nov 22 14:56:46.010: get_acct_params(784): Use method “h323” set by global config.
*Nov 22 14:56:46.014:     method:
*Nov 22 14:56:46.014:     cdrtag:
*Nov 22 14:56:46.014: voip_start_accounting_internal: Getting new uid
*Nov 22 14:56:46.014: voip_alloc_aaa_uid
*Nov 22 14:56:46.014: voip_aaa_acct_get_retrieved_attrs
*Nov 22 14:56:46.014: voip_aaa_acct_get_nas_port_details:
*Nov 22 14:56:46.014: get_nas_port: avail=1 type=4 nas-port=ISDN 6/0:D:1
*Nov 22 14:56:46.014: voip_aaa_lock_adb: uid(1341) count=1
*Nov 22 14:56:46.014: voip_start_accounting_internal: UID=1341
*Nov 22 14:56:46.014: voip_start_accounting_internal: Telephony Leg
*Nov 22 14:56:46.014: voip_start_accounting_internal(784):
*Nov 22 14:56:46.014:       calling num     :
*Nov 22 14:56:46.014:       called num      : 11
*Nov 22 14:56:46.014:       account num     :
*Nov 22 14:56:46.014:       setup time      : *14:56:46.014 PST Fri
*Nov 22 14:56:46.014:       gateway id      : Router.cisco2.com
*Nov 22 14:56:46.014:       connection id   : 82831163 FDA411D6 8660B639 91DC0E5F
*Nov 22 14:56:46.014:       call origin     : answer
*Nov 22 14:56:46.014:       call type       : Telephony
*Nov 22 14:56:46.014:       incoming conn id: 82831163 FDA411D6 8660B639 91DC0E5F

*Nov 22 14:56:46.018: RADIUS(0000053D): sending
*Nov 22 14:56:46.018: RADIUS(0000053D): Send Accounting-Request to 10.6.10.203:1646 id
                      121, len 462
*Nov 22 14:56:46.018: RADIUS:  authenticator C7 53 14 06 45 F2 49 DF - 53 D8 90 B3 1C
                      D1 74 03
*Nov 22 14:56:46.018: RADIUS:  Acct-Session-Id     [44]  10  “0000084C”
*Nov 22 14:56:46.018: RADIUS:  Vendor, Cisco       [26]  57
*Nov 22 14:56:46.018: RADIUS:   h323-setup-time    [25]  51
                      “h323-setup-time=*14:56:46.014 PST
*Nov 22 14:56:46.018: RADIUS:  Vendor, Cisco       [26]  36
*Nov 22 14:56:46.018: RADIUS:   h323-gw-id         [33]  30
                      “h323-gw-id=Router.cisco2.com”
*Nov 22 14:56:46.018: RADIUS:  Vendor, Cisco       [26]  56
*Nov 22 14:56:46.018: RADIUS:   Conf-Id            [24]  50  “h323-conf-id=82831163
                      FDA411D6 8660B639 91DC0E5F”
*Nov 22 14:56:46.018: RADIUS:  Vendor, Cisco       [26]  31
*Nov 22 14:56:46.018: RADIUS:   h323-call-origin   [26]  25  “h323-call-origin=answer”
*Nov 22 14:56:46.018: RADIUS:  Vendor, Cisco       [26]  32
*Nov 22 14:56:46.018: RADIUS:   h323-call-type     [27]  26
                      “h323-call-type=Telephony”
Accounting-response, len 20
!
```

Router received response from RADIUS server.

```
!
```

```
**Nov 22 14:56:49.022: RADIUS:  authenticated 08 C6 FC 0F 31 1D FA EA - 68 A7 5D 48 6F
                       47 96 FA
*Nov 22 14:56:49.022: voip_process_acct_reply(1341):
*Nov 22 14:56:49.022: voip_process_acct_reply(1341): event_message->type=0x210,
                       msg_t=2, rsp=1
*Nov 22 14:56:49.022: voip_process_acct_reply: acct notification call back for
                       method=h323
*Nov 22 14:56:49.022: acct_notif_cleanup:
*Nov 22 14:56:49.022: acct_ntf_deregistration:
*Nov 22 14:56:49.022: acct_ntf_deregistration: deregister with AAA EM, rsp_t=0x4,
                       ev_t=0x210
*Nov 22 14:56:49.022: acct_notif_cleanup: unlock adb
*Nov 22 14:56:49.022: voip_aaa_unlock_adb: uid(1341) count=0
*Nov 22 14:56:49.022: voip_aaa_cleanup_adb: dealloc uid (1341)
*Nov 22 14:56:49.022: voip_aaa_acct_get_dynamic_attrs: No cdb found from cdb tree
```

Managing the Collection of Voice Statistics

This section describes how to manage the collection of voice statistics on the gateway and documents the following tasks:

Configuring the FTP Server to Enable Archiving of Statistics from the Gateway

This task shows how to configure the FTP server to accept archived statistics from a Cisco IOS gateway.

Prerequisites

FTP Server

An FTP server must be configured before you can archive the collected statistics.

FTP Service Port

Normally, the FTP port is a well-known number, such as 21. However, another port number (not well-known) can receive data for specific purposes (for example, security), as long as the FTP client on voice gateways is configured to use the same port number.

User Account and File Directory

In order for the FTP client on the voice gateway to write files on the FTP server, FTP user accounts must be available (or well-known) to the FTP client. The FTP user accounts can be normal UNIX user accounts.

The FTP file upload directory in the FTP servers can also be specified for directory management purposes. System administrators can also restrict the privilege level of the user accounts in the upload directory for security and directory management purposes.

Note For this task, the external devices are assumed to be UNIX-like platforms (for example, Linux).

Step 1 Install the software.

Ensure that both the anonftp package and the wu-ftpd package are installed on the system. The versions installed should, at a minimum, match those below:

```
anonftp-3.0-9
wu-ftpd-2.6.1-6
```

Check to see whether the installation can be done with the following command:

rpm -qa | egrep '(wu-ftpd|anonftp)'

Step 2 Configure the IP aliasing for virtual domains.

Configure the IP aliases for the virtual domains so that there is an IP address routed through one of the available network interfaces.

The programs “netcfg” or “linuxconf” can also be used to set up the IP aliases (replacing 10.10.10.10 with your actual IP address for the FTP site).

If the IP address is to resolve to a domain name, you must set up a DNS server.

Step 3 Configure xinetd.conf.

Configure the /etc/xinetd.d/wu-ftpd file to handle FTP access, for example:

```
[root@wyvern xinetd.d]# cat wu-ftpd

# default: on

# description: The wu-ftpd FTP server knows the FTP # connections. It uses \
normal,unencrypted # usernames and passwords for authentication.

service ftp
{
        disable                 = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l -a
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                    = 10
}
```

Note It is very important that the “-1 -a” is specified in server_args and that the disable line is set to “no.” This tells inetd.conf to reference the commands in the /etc/ftpaccess file.

Step 4 **Edit /etc/ftpaccess.**

Edit the /etc/ftpaccess file. Make the basic changes using Linuxconf as the root. Additional changes must be made in this file manually. The virtual entry in the file should be placed at the bottom of the file and resembles the following:

```
# Virtual FTP entries for 10.10.10.10
virtual  10.10.10.10   root        /home/domain1
virtual  10.10.10.10   banner      /home/ftp/domain1/banner.msg
virtual  10.10.10.10   logfile     /var/log/virtual/domain1/xferlog
```

Where /home/domain1 is the root path for the virtual FTP server, /home/ftp/domain1/banner.msg is the path to the banner message to be displayed upon login, and /var/log/virtual/domain1/xferlog is the path to the transfer log.

Step 5 **Specify other options.**

Specify that all users can have access with the following line:

```
virtual 10.10.10.10  allow *
```

If only specific users are to be allowed access, list their usernames, as shown here:

```
virtual 10.10.10.10 allow user1
```

If anonymous FTP logins are to be disabled, set the IP address to private as shown here:

```
virtual 10.10.10.10 private
```

Step 6 **Secure or hide the FTP server.**

When FTP users log on to the system, they should be allowed into the directory specified in the root path only. There are several steps as follows:

a. Determine whether or not there is one user or a group of users that will be logging into the virtual FTP site.

b. Specify the home directory in the /etc/passwd file for the user specified in the /etc/ftpaccess/ file, followed by a /./. The entry will look like this:

```
user1:X:2453:group1::/var/ftp/home/domain1/./:/bin/bash
```

Include the following line in your FTP access file, if user1 is the only user accessing this virtual FTP site, following the virtual configuration lines:

```
guestuser user1
```

When logging into 10.10.10.10, user1 is automatically dropped into /home/domain1 but will see this as the / directory. User1 will not be able to move outside of that directory.

c. Specify a group of users in addition to user1 in /etc/group. You should then add the following line to /etc/ftpaccess following the virtual FTP entry:

```
guestgroup group1
```

d. Specify the administrator of the FTP site (user2 in the example and exempt from the group rule) as follows:

```
guestgroup group1
realuser user2
```

Step 7 **Provide for basic shell access.**

Open the /var/ftp (created on the system when the anonftp rpm was installed), and copy /var/ftp/bin, /var/ftp/etc/, and /var/ftp/lib into the root directory of the virtual FTP site (in this case, /var/ftp/home/domain1).

Step 8 **Restart services so that the configuration takes effect.**

Close the configuration file after making all of the changes, and restart the inet services (where FTP services are specified) by typing the following:

```
/sbin/service xinetd stop
```

then:

```
/sbin/service xinetd start
```

Configuring the Gateway to Archive Statistics to an FTP Server

This task shows you how to configure the gateway to archive the collected statistics to the specified FTP server and how to configure the maximum allowable file size.

Note This procedure can also be used to archive Cisco VoIP internal error codes (IECs) to an FTP server.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics push ftp url** ftp-url [max-file-size value]
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics push ftp url** *ftp-url* **[max-file-size** *value***]** **Example:** `Router(config)# voice statistics push ftp url ftp://me:secret@abccompanyhost:23/products max-file-size 3000`	Configures the FTP server location where statistics will be archived. The *ftp-url* argument is expressed in the following form: ftp://user:password@host:port/path1/path2/ • The **max-file-size** keyword is optional and specifies the maximum FTP file size, in bytes. The *value* argument has a valid range from 1024 to 4294967295. The default is 400000000 (4 GB).
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Example

The following sample output is the message that you see on your console when the statistics are being archived to the FTP server:

```
Writing /ftp_files/vstats.3660-1.2002-04-25T020500Z !
Writing /ftp_files/vstats.3660-1.2002-04-25T020500Z.done !
!
```

```
01:43:22: %VSTATS-6-VCSR: SEQ=0:
stats_type,version,format,gw_id,start_time,end_time,rec_count
VCSR_SIG,1,asc,3660-1,2002-04-25T02:00:00Z,2002-04-25T02:05:00Z,9
record_type,trunk_group_id,voice_port_id,in_call,in_ans,in_fail,out_call,out_ans,out_fail,
in_szre_d,out_szre_d,in_conn_d,out_conn_d,orig_disconn,in_ans_abnorm,out_ans_abnorm,in_mcd
out_mcd,in_pdd,out_pdd,in_setup_delay,out_setup_delay,lost_pkt,latency,jitter,in_cc_no,
out_cc_no,in_cc_id,in_cc_cntr,out_cc_id,out_cc_cntr
gw,,,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,16,0,16,0
ip,,,0,0,0,0,0,0,0,0,0,0,
01:43:22: %VSTATS-6-VCSR: SEQ=1:
0,0,0,0,0,0,0,0,0,0,0,1,1,16,0,16,0
pstn,,,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
vp,,4/0/0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
vp,,4/0/1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
vp,,4/1/0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
vp,,4/1/1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
vp,,2/0:23,0,0,0,0,0,0,|0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
vp,,2/1:23,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,,,1,1,16,0,16,0
stats_type,version,format,gw_id,start_time,e
01:43:22: %VSTATS-6-VCSR: SEQ=2:
nd_time,rec_count
VCSR_ACC,1,asc,3660-1,2002-04-25T02:00:00Z,2002-04-25T02:05:00Z,2
acct_type,acct_type_name,acct_pass_criteria,pstn_in_pass,pstn_in_fail,pstn_out_pass,
pstn_out_fail,ip_in_pass,ip_in_fail,ip_out_pass,ip_out_fail
ml,h323,1,0,0,0,0,0,0,0
ml,h323-1,1,0,0,0,0,0,0,0
```

Configuring the Gateway to Archive Statistics to a Syslog Server

This task shows you how to configure the gateway to archive the call statistics records to a syslog server.

Note This procedure can also be used to archive Cisco VoIP internal error codes (IECs) to a sylog server.

Prerequisites

The external syslog server must be configured to receive voice call statistics from the gateway. For information about enabling a syslog server to receive voice call statistics information from a gateway, refer to Task 2 in *Enabling Management Protocols: NTP, SNMP, and Syslog* on Cisco.com.

For information about configuring a Solaris syslog server, refer to Task 4 (“Using Syslog, NTP, and Modem Call Records to Isolate and Troubleshoot Faults”) in the *Basic Dial NMS Implementation Guide* on Cisco.com.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice statistics push syslog [max-msg-size *value*]**
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice statistics push syslog [max-msg-size *value*]** **Example:** `Router(config)# voice statistics push syslog`	Archives voice statistics to an external syslog server and specifies the maximum syslog message size. • The **max-msg-size keyword is optional and specifies the maximum size in bytes of a voice statistics file to be pushed to the syslog server**. Valid values are from 1024 to 4294967295. The default value is 4000000000 (4 GB).
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Displaying Memory Usage

This task shows you how to display memory usage either as an absolute value or as a percentage.

SUMMARY STEPS

1. **enable**
2. **show voice statistics memory-usage** {**all** | **csr** | **iec}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics memory-usage {all | csr | iec}** **Example:** `Router(config)# show voice statistics memory-usage csr`	Displays current memory usage (absolute or percentage).

Displaying All Statistics and Pushing Them to an FTP or Syslog Server

This task shows how to display all statistics and push them to an FTP or syslog server.

Note This procedure can also be used if you are collecting statistics for VoIP internal error codes (IECs).

SUMMARY STEPS

1. **enable**
2. **show voice statistics csr since-reset all** [**mode** {**concise** | **verbose**}]
3. **show voice statistics csr since-reset all push {all | ftp | syslog}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics csr since-reset all** [**mode** {**concise** | **verbose**}] **Example:** `Router# show voice statistics csr since-reset all mode concise`	Displays all statistics (including both signaling and accounting statistics) since the last reset or reboot of the gateway.
Step 3	**show voice statistics csr since-reset all push {all | ftp | syslog}** **Example:** `Router# show voice statistics csr since-reset all push syslog`	Pushes all displayed statistics to either the FTP or syslog server, or to both servers. • The statistics are first displayed on the console before being pushed to the FTP or syslog server.

Clearing the Collected Call Statistics

This task shows you how to clear the collected statistics. The results of the **clear voice statistics csr** command can be viewed using the **show voice statistics** command.

Once the **clear voice statistics csr** command has been issued, all statistics collected using the **voice statistics time-range since-reset** command are removed and the counters are reset.

Restrictions

Only since-reset counters can be reset. Specific or periodic counts cannot be reset using the **clear voice statistics csr** command. This command cannot be used in nonprivileged mode.

SUMMARY STEPS

1. **enable**
2. **clear voice statistics csr**
3. **show voice statistics csr since-reset all**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear voice statistics csr** **Example:** `Router# clear voice statistics csr`	Clears the collected statistics.
Step 3	**show voice statistics csr since-reset all** **Example:** `Router# show voice statistics csr since-reset all`	Displays the collected statistics since a reset occurred. Enter this command after entering the **clear voice statistics csr** command to verify that the statistics have been cleared.

Troubleshooting Tips

The gateway does not recognize the **clear voice statistics csr** command in nonprivileged mode and displays the following message:

```
Router> clear voice statistics csr
                ^
% Invalid input detected at '^' marker.
```

If you see this message, enter this command from privileged EXEC mode.

Monitoring the Statistical Reporting

This section contains the following subsections:

Using Debug Commands for Monitoring Gateway Reporting

To monitor the statistical reporting and the collection of data by the gateway, you must turn on the following debug commands:

- debug radius accounting
- debug event-manager
- debug voice statistics csr
- debug voice statistics accounting
- debug voice statistics core

Once the debugging is turned on, you can review the data, evaluate the performance of the network, and identify impaired voice equipment.

The following output examples show a collection of records occurring between intervals and a voice call going through the gateway.

Example of Record Collection Between Intervals

In the following example, the gateway was unable to "push" the data to the FTP server.

```
vstats_timer_handle_interval_event():Between Intervals!

04:52:37: vstats_acct_interval_end: interval_tag = 4
04:52:37: vstats_acct_interval_end: pushing out, tag=3
04:52:37: vstats_acct_clean_history_stats:
04:52:37: vstats_acct_clean_history_stats: stats (tag=3) not to be deleted
04:52:37: vstats_acct_clean_history_stats: stats (tag=2) not to be deleted
04:52:37: vstats_acct_create_empty_stats:
04:52:37: vstats_acct_create_new_rec_list:
04:52:37: vstats_acct_create_new_rec_list: add acct rec: methodlist=h323, acct-criteria=2
04:52:37: vstats_acct_create_new_rec:
04:52:37: vstats_acct_add_rec_entry:
04:52:37: vstats_acct_add_stats_entry:
04:52:37: vstat_push_driver_file_open():Cannot open
ftp://sgcp:sgcp@abc-pc:21//ftp_files/vstats.5400-GW.2003-02-13T162000Z.
errno=65540=Unknown error 65540
vstat_push_drv_activate_ftp_file_tx():open file
(ftp://sgcp:sgcp@jeremy-pc:21//ftp_files/vstats.5400-GW.2003-02-13T162000Z)=(ftp://sgcp:sg
cp@abc-pc:21//ftp_files/vstats.5400-GW.2003-02-13T162000Z)failed!
vstats_push_api_push_formatted_text():Start CMD error!
```

Example of a Voice Call Going Through the Gateway

```
04:55:07: EM: Notify the producer not to produce
04:55:07: RADIUS(00000019): Storing nasport 0 in rad_db
04:55:07: RADIUS(00000019): Config NAS IP: 0.0.0.0
04:55:07: RADIUS(00000019): sending
04:55:07: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.6.10.203
04:55:07: RADIUS(00000019): Send Accounting-Request to 10.6.10.203:1646 id 21645/49,len496
04:55:07: RADIUS:  authenticator C5 B8 AA 2E C3 AF 02 93 - 45 0B AE E5 B6 B2 99 1F
04:55:07: RADIUS:  Acct-Session-Id     [44]  10 "00000020"
04:55:07: RADIUS:  Vendor, Cisco       [26]  57
04:55:07: RADIUS:   h323-setup-time    [25]  51  "h323-setup-time=*16:22:30.994 UTC
          Thu Feb 13 2003"
04:55:07: RADIUS:  Vendor, Cisco       [26]  27
04:55:07: RADIUS:   h323-gw-id         [33]  21  "h323-gw-id=5400-GW."
04:55:07: RADIUS:  Vendor, Cisco       [26]  56
04:55:07: RADIUS:  Conf-Id             [24]  50  "h323-conf-id=2F4ED2E3 3EA611D7 800E0002
          B935C142"
04:55:07: RADIUS:  Vendor, Cisco       [26]  31
04:55:07: RADIUS:  h323-call-origin    [26]  25  "h323-call-origin=answer"
```

```
04:55:07: RADIUS:  Vendor, Cisco       [26]  32
04:55:07: RADIUS:   h323-call-type     [27]  26  "h323-call-type=Telephony"
04:55:07: RADIUS:  Vendor, Cisco       [26]  65
04:55:07: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=2F4ED2E3 3EA611D7
          800E0002 B935C142"
04:55:07: RADIUS:  Vendor, Cisco       [26]  30
04:55:07: RADIUS:   Cisco AVpair       [1]   24  "subscriber=RegularLine"
04:55:07: RADIUS:  Vendor, Cisco       [26]  35
04:55:07: RADIUS:   Cisco AVpair       [1]   29  "gw-rxd-cdn=ton:0,npi:0,#:11"
04:55:07: RADIUS:  Vendor, Cisco       [26]  32
04:55:07: RADIUS:   Cisco AVpair       [1]   26  "calling-party-category=9"
04:55:07: RADIUS:  Vendor, Cisco       [26]  33
04:55:07: RADIUS:   Cisco AVpair       [1]   27  "transmission-medium-req=0"
04:55:07: RADIUS:  User-Name           [1]   4   "22"
04:55:07: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
04:55:07: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
04:55:07: RADIUS:  Vendor, Cisco       [26]  20
04:55:07: RADIUS:   cisco-nas-port     [2]   14  "ISDN 6/0:D:1"
04:55:07: RADIUS:  NAS-Port            [5]   6   0
04:55:07: RADIUS:  Calling-Station-Id  [31]  4   "22"
04:55:07: RADIUS:  Called-Station-Id   [30]  4   "11"
04:55:07: RADIUS:  Service-Type        [6]   6   Login                     [1]
04:55:07: RADIUS:  NAS-IP-Address      [4]   6   10.6.43.101
04:55:07: RADIUS:  Acct-Delay-Time     [41]  6   0
04:55:07: RADIUS(0000001A): Config NAS IP: 0.0.0.0
04:55:07: RADIUS(0000001A): sending
04:55:07: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.6.10.203
04:55:07: RADIUS(0000001A): Send Accounting-Request to 10.6.10.203:1646 id 21645/50,len427
04:55:07: RADIUS:  authenticator E4 98 06 8C 48 63 4F AA - 56 4F 40 12 33 F0 F5 99
04:55:07: RADIUS:  Acct-Session-Id     [44]  10  "00000021"
04:55:07: RADIUS:  Vendor, Cisco       [26]  57
04:55:07: RADIUS:   h323-setup-time    [25]  51  "h323-setup-time=*16:22:31.006 UTC
          Thu Feb 13 2003"
04:55:07: RADIUS:  Vendor, Cisco       [26]  27
04:55:07: RADIUS:   h323-gw-id         [33]  21  "h323-gw-id=5400-GW."
04:55:07: RADIUS:  Vendor, Cisco       [26]  56
04:55:07: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=2F4ED2E3 3EA611D7 800E0002
          B935C142"
04:55:07: RADIUS:  Vendor, Cisco       [26]  34
04:55:07: RADIUS:   h323-call-origin   [26]  28 "h323-call-origin=originate"
04:55:07: RADIUS:  Vendor, Cisco       [26]  27
04:55:07: RADIUS:   h323-call-type     [27]  21  "h323-call-type=VoIP"
04:55:07: RADIUS:  Vendor, Cisco       [26]  65
04:55:07: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=2F4ED2E3 3EA611D7
          800E0002 B935C142"
04:55:07: RADIUS:  Vendor, Cisco       [26]  30
04:55:07: RADIUS:   Cisco AVpair       [1]   24  "subscriber=RegularLine"
04:55:07: RADIUS:  Vendor, Cisco       [26]  30
04:55:07: RADIUS: Cisco AVpair       [1]   24  "session-protocol=cisco"
04:55:07: RADIUS:  Vendor, Cisco       [26]  35
04:55:07: RADIUS:   Cisco AVpair       [1]   29  "gw-rxd-cdn=ton:0,npi:0,#:11"
04:55:07: RADIUS:  User-Name           [1]   4   "22"
04:55:07: RADIUS:  Acct-Status-Type    [40]  6   Start                     [1]
04:55:07: RADIUS:  Calling-Station-Id  [31]  4   "22"
04:55:07: RADIUS:  Called-Station-Id   [30]  4   "11"
04:55:07: RADIUS:  Service-Type        [6]   6   Login                     [1]
04:55:07: RADIUS:  NAS-IP-Address      [4]   6   10.6.43.101
04:55:07: RADIUS:  Acct-Delay-Time     [41]  6   0
04:55:07: EM: No consumer registered for event type NEWINFO
04:55:07: EM: Notify the producer not to produce
04:55:07: EM: No consumer registered for event type NEWINFO
04:55:07: EM: Notify the producer not to produce
04:55:08: RADIUS: no sg in radius-timers: ctx 0x65BAB1BC sg 0x0000
04:55:08: RADIUS: Retransmit to (10.6.10.203:1645,1646) for id 21645/50
```

```
04:55:08: RADIUS: acct-delay-time for 403963FC (at 403965A1) now 1
04:55:09: RADIUS: no sg in radius-timers: ctx 0x65ADB8EC sg 0x0000
04:55:09: RADIUS: Retransmit to (10.6.10.203:1645,1646) for id 21645/49
04:55:09: RADIUS: acct-delay-time for 40389BFC (at 40389DE6) now 1
04:55:10: RADIUS: no sg in radius-timers: ctx 0x65BAB1BC sg 0x0000
04:55:10: RADIUS: Fail-over to (10.8.159.105:1645,1645) for id 21645/51
04:55:10: RADIUS: acct-delay-time for 403963FC (at 403965A1) now 2
04:55:10: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.8.159.105
04:55:10: RADIUS: Received from id 21645/53 10.8.159.105:1645, Accounting-response, len 20
04:55:10: RADIUS:  authenticator 57 EF DD 90 0F 88 76 EA - A5 3D A7 44 0D 90 66 16
04:55:10: vstats_acct_rsp_handler:   methodlist=h323, rsp_type=0x1
04:55:10: acct_rsp_status=1 callid= 26, incoming=0, leg=2
04:55:10: vstats_acct_rsp_handler: last acct msg not sent yet. methodlist: h323
04:55:10: RADIUS: no sg in radius-timers: ctx 0x65ADB8EC sg 0x0000
04:55:10: RADIUS: Fail-over to (10.8.159.105:1645,1645) for id 21645/52
04:55:10: RADIUS: acct-delay-time for 40389BFC (at 40389DE6) now 2
04:55:10: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.8.159.105
04:55:10: RADIUS: Received from id 21645/54 10.8.159.105:1645, Accounting-response, len 20
04:55:10: RADIUS:  authenticator 97 88 6C BA DA 22 E7 5E - 73 EC 21 C6 36 1B 93 18
04:55:10: vstats_acct_rsp_handler:   methodlist=h323, rsp_type=0x1
04:55:10: acct_rsp_status=callid= 25, incoming=1, leg=1
04:55:10: vstats_acct_rsp_handler: last acct msg not sent yet. methodlist: h323
04:55:13: RADIUS(0000001A): Config NAS IP: 0.0.0.0
04:55:13: RADIUS(0000001A): sending
04:55:13: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.6.10.203
04:55:13: RADIUS(0000001A): Send Accounting-Request to 10.6.10.203:1646 id 21645/55,len885
04:55:13: RADIUS:  authenticator F8 4F F1 30 7E 8B 5B 46 - EF AE 17 2D 5C BA 36 E5
04:55:13: RADIUS:  Acct-Session-Id     [44]  10  "00000021"
04:55:13: RADIUS:  Vendor, Cisco       [26]  57
04:55:13: RADIUS:   h323-setup-time    [25]  51  "h323-setup-time=*16:22:31.006 UTC
          Thu Feb 13 2003"
04:55:13: RADIUS:  Vendor, Cisco       [26]  27
04:55:13: RADIUS:   h323-gw-id         [33]  21  "h323-gw-id=5400-GW."
04:55:13: RADIUS:  Vendor, Cisco       [26]  56
04:55:13: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=2F4ED2E3 3EA611D7 800E0002
          B935C142"
04:55:13: RADIUS:  Vendor, Cisco       [26]  34
04:55:13: RADIUS:   h323-call-origin   [26]  28  "h323-call-origin=originate"
04:55:13: RADIUS:  Vendor, Cisco       [26]  27
04:55:13: RADIUS:   h323-call-type     [27]  21  "h323-call-type=VoIP"
04:55:13: RADIUS:  Vendor, Cisco       [26]  65
04:55:13: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=2F4ED2E3 3EA611D7
          800E0002 B935C142"
04:55:13: RADIUS:  Vendor, Cisco       [26]  30
04:55:13: RADIUS:   Cisco AVpair       [1]   24  "subscriber=RegularLine"
04:55:13: RADIUS:  Vendor, Cisco       [26]  30
04:55:13: RADIUS:   Cisco AVpair       [1]   24  "session-protocol=cisco"
04:55:13: RADIUS:  Vendor, Cisco       [26]  35
04:55:13: RADIUS:   Cisco AVpair       [1]   29  "gw-rxd-cdn=ton:0,npi:0,#:11"
04:55:13: RADIUS:  Vendor, Cisco       [26]  59
04:55:13: RADIUS:   h323-connect-time  [28]  53  "h323-connect-time=*16:22:31.046 UTC
          Thu Feb 13 2003"
04:55:13: RADIUS:  Acct-Input-Octets   [42]  6   2241
04:55:13: RADIUS:  Acct-Output-Octets  [43]  6   81

04:55:13: RADIUS:  Acct-Input-Packets  [47]  6   113
04:55:13: RADIUS:  Acct-Output-Packets [48]  6   5
04:55:13: RADIUS:  Acct-Session-Time   [46]  6   5
04:55:13: RADIUS:  Vendor, Cisco       [26]  62
04:55:13: RADIUS:   h323-disconnect-tim[29]  56  "h323-disconnect-time=*16:22:36.070 UTC
          Thu Feb 13 2003"
04:55:13: RADIUS:  Vendor, Cisco       [26]  32
04:55:13: RADIUS:   h323-disconnect-cau[30]  26  "h323-disconnect-cause=10"
04:55:13: RADIUS:  Vendor, Cisco       [26]  38
```

```
04:55:13: RADIUS:   h323-remote-address[23]  32  "h323-remote-address=10.0.0.110"
04:55:13: RADIUS:  Vendor, Cisco       [26]  24
04:55:13: RADIUS:   Cisco AVpair       [1]   18  "release-source=1"
04:55:13: RADIUS:  Vendor, Cisco       [26]  29
04:55:13: RADIUS:   h323-voice-quality [31]  23  "h323-voice-quality=-1"
04:55:13: RADIUS:  Vendor, Cisco       [26]  57
04:55:13: RADIUS:   Cisco AVpair       [1]   51  "alert-timepoint=*16:22:31.030 UTC
          Thu Feb 13 2003"
04:55:13: RADIUS:  Vendor, Cisco       [26]  39
04:55:13: RADIUS:   Cisco AVpair       [1]   33  "remote-media-address=10.0.0.110"
04:55:13: RADIUS:  Vendor, Cisco       [26]  44
04:55:13: RADIUS:   Cisco AVpair       [1]   38  "gw-final-xlated-cdn=ton:0,npi:0,#:11"
04:55:13: RADIUS:  Vendor, Cisco       [26]  44
04:55:13: RADIUS:   Cisco AVpair       [1]   38  "gw-final-xlated-cgn=ton:0,npi:1,#:22"
04:55:13: RADIUS:  User-Name           [1]   4   "22"
04:55:13: RADIUS:  Acct-Status-Type    [40]  6   Stop                       [2]
04:55:13: RADIUS:  Calling-Station-Id  [31]  4   "22"
04:55:13: RADIUS:  Called-Station-Id   [30]  4   "11"
04:55:13: RADIUS:  Service-Type        [6]   6   Login                      [1]
04:55:13: RADIUS:  NAS-IP-Address      [4]   6   10.6.43.101
04:55:13: RADIUS:  Acct-Delay-Time     [41]  6   0
04:55:13: RADIUS(00000019): Using existing nas_port 0
04:55:13: RADIUS(00000019):Config NAS IP: 0.0.0.0
04:55:13: RADIUS(00000019):sending
04:55:13: RADIUS/ENCODE: Best Local IP-Address 1.6.43.101 for Radius-Server 10.6.10.203
04:55:13: RADIUS(00000019): Send Accounting-Request to 10.6.10.203:1646 id 21645/56,len766
04:55:13: RADIUS:  authenticator 61 60 EB 92 29 5C DE B4 - CE 40 1C AB E3 A1 C8 F7
04:55:13: RADIUS:  Acct-Session-Id     [44]  10  "00000020"
04:55:13: RADIUS:  Vendor, Cisco       [26]  57
04:55:13: RADIUS:   h323-setup-time    [25]  51  "h323-setup-time=*16:22:30.994 UTC Thu
          Feb 13 2003"
04:55:13: RADIUS:  Vendor, Cisco       [26]  27
04:55:13: RADIUS:   h323-gw-id         [33]  21  "h323-gw-id=5400-GW."
04:55:13: RADIUS:  Vendor, Cisco       [26]  56
04:55:13: RADIUS:   Conf-Id            [24]  50  "h323-conf-id=2F4ED2E3 3EA611D7 800E0002
          B935C142"
04:55:13: RADIUS:  Vendor, Cisco       [26]  31
04:55:13: RADIUS:   h323-call-origin   [26]  25  "h323-call-origin=answer"
04:55:13: RADIUS:  Vendor, Cisco       [26]  32
04:55:13: RADIUS:   h323-call-type     [27]  26  "h323-call-type=Telephony"
04:55:13: RADIUS:  Vendor, Cisco       [26]  65
04:55:13: RADIUS:   Cisco AVpair       [1]   59  "h323-incoming-conf-id=2F4ED2E3 3EA611D7
          800E0002 B935C142"
04:55:13: RADIUS:  Vendor, Cisco       [26]  30
04:55:13: RADIUS:   Cisco AVpair       [1]   24  "subscriber=RegularLine"
04:55:13: RADIUS:  Vendor, Cisco       [26]  35
04:55:13: RADIUS:   Cisco AVpair       [1]   29  "gw-rxd-cdn=ton:0,npi:0,#:11"
04:55:13: RADIUS:  Vendor, Cisco       [26]  32
04:55:13: RADIUS:   Cisco AVpair       [1]   26  "calling-party-category=9"
04:55:13: RADIUS:  Vendor, Cisco       [26]  33
04:55:13: RADIUS:   Cisco AVpair       [1]   27  "transmission-medium-req=0"
04:55:13: RADIUS:  Vendor, Cisco       [26]  59
04:55:13: RADIUS:   h323-connect-time  [28]  53  "h323-connect-time=*16:22:31.046 UTC Thu
          Feb 13 2003"
04:55:13: RADIUS:  Acct-Input-Octets   [42]  6   81
04:55:13: RADIUS:  Acct-Output-Octets  [43]  6   2241

04:55:13: RADIUS:  Acct-Input-Packets  [47]  6   5
04:55:13: RADIUS:  Acct-Output-Packets [48]  6   113
04:55:13: RADIUS:  Acct-Session-Time   [46]  6   5
04:55:13: RADIUS:  Vendor, Cisco       [26]  62
04:55:13: RADIUS:   h323-disconnect-tim[29]  56  "h323-disconnect-time=*16:22:36.064 UTC
          Thu Feb 13 2003"
04:55:13: RADIUS:  Vendor, Cisco       [26]  32
```

```
04:55:13: RADIUS:   h323-disconnect-cau[30]  26  "h323-disconnect-cause=10"
04:55:13: RADIUS:  Vendor, Cisco       [26]  35
04:55:13: RADIUS:   Cisco AVpair       [1]   29  "h323-ivr-out=Tariff:Unknown"
04:55:13: RADIUS:  Vendor, Cisco       [26]  24
04:55:13: RADIUS:   Cisco AVpair       [1]   18  "release-source=1"
04:55:13: RADIUS:  Vendor, Cisco       [26]  28
04:55:13: RADIUS:   h323-voice-quality [31]  22  "h323-voice-quality=0"
04:55:13: RADIUS:  User-Name           [1]   4   "22"
04:55:13: RADIUS:  Acct-Status-Type    [40]  6   Stop [2]
04:55:13: RADIUS:  NAS-Port-Type       [61]  6   Async                     [0]
04:55:13: RADIUS:  Vendor, Cisco       [26]  20
04:55:13: RADIUS:   cisco-nas-port     [2]   14  "ISDN 6/0:D:1"
04:55:13: RADIUS:  NAS-Port            [5]   6   0
04:55:13: RADIUS:  Calling-Station-Id  [31]  4   "22"
04:55:13: RADIUS:  Called-Station-Id   [30]  4   "11"
04:55:13: RADIUS:  Service-Type        [6]   6   Login                     [1]
04:55:13: RADIUS:  NAS-IP-Addres       [4]   6   10.6.43.101
04:55:13: RADIUS:  Acct-Delay-Time     [41]  6   0
04:55:14: RADIUS: no sg in radius-timers: ctx 0x65BAB070 sg 0x0000
04:55:14: RADIUS: Retransmit to (10.6.10.203:1645,1646) for id 21645/55
04:55:14: RADIUS: acct-delay-time for 40553934 (at 40553CA3) now 1
04:55:14: RADIUS: no sg in radius-timers: ctx 0x65BA8284 sg 0x0000
04:55:14: RADIUS: Retransmit to (10.6.10.203:1645,1646) for id 21645/56
04:55:14: RADIUS: acct-delay-time for 405546C4 (at 405549BC) now 1
04:55:15: RADIUS: no sg in radius-timers: ctx 0x65BAB070 sg 0x0000
04:55:15: RADIUS: Fail-over to (10.8.159.105:1645,1645) for id 21645/57
04:55:15: RADIUS: acct-delay-time for 40553934 (at 40553CA3) now 2
04:55:15: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.8.159.105
04:55:15: RADIUS: no sg in radius-timers: ctx 0x65BA8284 sg 0x0000
04:55:15: RADIUS: Fail-over to (10.8.159.105:1645,1645) for id 21645/58
04:55:15: RADIUS: acct-delay-time for 405546C4 (at 405549BC) now 2
04:55:15: RADIUS/ENCODE: Best Local IP-Address 10.6.43.101 for Radius-Server 10.8.159.105
04:55:15: RADIUS: Received from id 21645/59 10.8.159.105:1645, Accounting-response, len 20
04:55:15: RADIUS:  authenticator B1 C4 5E FC DB FA 74 A4 - 05 E2 34 52 1A 11 26 06
04:55:15: vstats_acct_rsp_handler:   methodlist=h323, rsp_type=0x4
04:55:15: acct_rsp_status=1 callid= 26, incoming=0, leg=2
04:55:15: vstats_acct_rsp_handler: increment since-reset counter
04:55:15: vstats_acct_rsp_handler: increment interval counter
04:55:15: RADIUS: Received from id 21645/60 10.8.159.105:1645, Accounting-response, len 20
04:55:15: RADIUS:  authenticator 0E 70 74 2F E5 D8 EE 98 - B9 C0 DA 66 74 ED 84 77
04:55:15: vstats_acct_rsp_handler:   methodlist=h323, rsp_type=0x4
04:55:15: acct_rsp_status=1 callid= 25, incoming=1, leg=1
04:55:15: vstats_acct_rsp_handler: increment since-reset counter
04:55:15: vstats_acct_rsp_handler: increment interval counter
```

Using Cause Code Statistics

By examining disconnect cause codes, you can understand the distribution of the various cause codes on the voice ports, trunk groups, and gateway, and which can help you determine why the voice calls were disconnected.

The call disconnection cause values are taken from International Telecommunication Union Telecommunication Standardization Sector (ITU-T) standard Q.931 and are as follows:

- Private Network-Network Interface (PNNI) references **ITU-T** standard Q.2931 for the information element (IE) causes.
- PNNI R15 6.3.6.3 contains the “crankback” causes.
- **ITU-T** standard Q.2931 references **ITU-T** standard Q.2610.
- **ITU-T** standard Q.2610 lists a few cause codes and references **ITU-T** standard Q.850.
- ITU-T standard Q.850 lists the bulk of the cause codes.

For specific information on the call disconnection cause values, see the *Cisco IOS Voice Troubleshooting and Monitoring Guide*.

Prerequisites

CSR configurations must be enabled before you can examine the voice calls that are made through the gateway.

Restrictions

Statistics of non-DID calls are not consistent with those of the underlying ISDN module.

SUMMARY STEPS

1. **enable**
2. **show voice statistics csr since-reset aggregation-level all**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics csr since-reset aggregation-level all** **Example:** `Router# show voice statistics csr since-reset aggregation-level all`	Displays all aggregation-level statistics since the last system reset or reboot.

Example

The following sample output shows cause-code statistics since the last reset for all aggregation levels.

```
Router# show voice statistics csr since-reset aggregation-level all

Client Type: VCSR
        Start Time: 1993-03-01T00:03:15Z        End Time: 1993-03-03T22:02:18Z

record_type=gw,trunk_group_id=,voice_port_id=,in_call=26,in_ans=16,in_fail=10,out_call=26,
out_ans=16,out_fail=10,in_szre_d=387,out_szre_d=380,in_conn_d=330,out_conn_d=323,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=2340,
out_pdd=12340,in_setup_delay=2340,out_setup_delay=3340,lost_pkt=0,latency=0,jitter=0,
in_disc_cc_16=16,in_disc_cc_18=2,in_disc_cc_19=3,in_disc_cc_34=5,out_disc_cc_16=16,
out_disc_cc_18=2,out_disc_cc_19=3,out_disc_cc_34=5
!
record_type=ip,trunk_group_id=,voice_port_id=,in_call=26,in_ans=16,in_fail=10,out_call=0,
out_ans=0,out_fail=0,in_szre_d=387,out_szre_d=0,in_conn_d=330,out_conn_d=0,orig_disconn=0,
in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=2340,out_pdd=0,
in_setup_delay=2340,out_setup_delay=0,lost_pkt=20,latency=15,jitter=10,in_disc_cc_16=16,
in_disc_cc_18=2,in_disc_cc_19=3,in_disc_cc_34=5,out_disc_cc_16=3
!
```

Table 55 shows two types of cause codes listed in the output above.

Table 55 Significant Fields of the show voice statistics csr since-reset aggregation-level all Command

Field	Description
in_disc_cc_16=16	Count of incoming calls that are disconnected with a specific cause code number. In this example, the value 16 indicates normal call clearing
out_disc_cc_16=3	Count of outgoing calls that are disconnected with a specific cause code number.

Displaying Quality of Service Indicators

The quality of service (QoS) indicators per voice call are the results of transmitting and receiving voice packets in the IP interface. These results are included in the CSRs and are as follows:

- Lost packet value: the number of calls losing more than the configured number of packets. The default lost packet threshold is 1000 milliseconds.
- Packet latency value: the number of calls with voice packets encountering more than the configured amount of latency. The default packet latency threshold is 250 milliseconds.
- Packet jitter value: the number of calls with voice packets encountering more than the configured amount of jitter. The default packet jitter threshold is 250 milliseconds.

Before you can determine that any voice call with IP voice packets is deviating from the desired level of quality, you must configure the threshold values of lost packets, latency, and jitter. See the following sections:

- Configuring the Minimum Call Duration and Signaling Thresholds, page 16
- Displaying Memory Usage, page 39

Restrictions

Call statistics for MGCP calls are not guaranteed to be correct and accurate. In addition, statistics of non-DID ISDN calls are not consistent with those of the underlying ISDN module.

SUMMARY STEPS

1. **enable**
2. **show voice statistics csr since-reset aggregation-level all**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show voice statistics csr since-reset aggregation-level all** **Example:** `Router# show voice statistics csr since-reset aggregation-level all`	Displays the voice statistics of the voice calls made through the gateway.

Example

The following sample output displays lost packet, latency, and jitter QoS information:

```
Router# show voice statistics csr since-reset aggregation-level all

Client Type: VCSR
        Start Time: 1993-03-01T00:03:15Z        End Time: 1993-03-03T22:02:18Z

record_type=gw,trunk_group_id=,voice_port_id=,in_call=26,in_ans=16,in_fail=10,out_call=26,
out_ans=16,out_fail=10,in_szre_d=387,out_szre_d=380,in_conn_d=330,out_conn_d=323,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=2340,
out_pdd=12340,in_setup_delay=2340,out_setup_delay=3340,lost_pkt=10,latency=3,jitter=5,
in_disc_cc_16=16,in_disc_cc_18=2,in_disc_cc_19=3,in_disc_cc_34=5,out_disc_cc_16=16,
out_disc_cc_18=2,out_disc_cc_19=3,out_disc_cc_34=5
!
record_type=ip,trunk_group_id=,voice_port_id=,in_call=26,in_ans=16,in_fail=10,out_call=0,
out_ans=0,out_fail=0,in_szre_d=387,out_szre_d=0,in_conn_d=330,out_conn_d=0,orig_disconn=0,
in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=2340,out_pdd=0,
in_setup_delay=2340,out_setup_delay=0,lost_pkt=10,latency=3,jitter=5,in_disc_cc_16=16,
in_disc_cc_18=2,in_disc_cc_19=3,in_disc_cc_34=5,out_disc_cc_16=0
!
```

Configuration Examples for Voice Performance Statistics on Cisco Gateways

This section includes the following examples:

- Maximum Duration for Storage: Example, page 55

User-Specific Configurations for Call Statistic Collection: Example

The following example shows the user-specific configuration that includes accounting method, time ranges, maximum storage duration, and the archiving of the statistics to an FTP and syslog server:

```
Router# show running-config | include voice statistics

voice statistics accounting method h323-1 pass start-interim-stop
voice statistics time-range since-reset
voice statistics time-range periodic 5minutes start 00:00 end 00:00 days-of-week daily
voice statistics max-storage-duration minute 40
voice statistics push ftp url ftp://sgcp:sgcp@anyone-pc:21//ftp_files
voice statistics push syslog max-file-size 4000000000
voice statistics display-format separator new-line
voice statistics field-params lost-packet 1000
voice statistics field-params mcd 2
voice statistics field-params packet-jitter 255
voice statistics field-params packet-latency 150
```

Note The type of statistics is not shown in a running configuration because the **voice statistics type csr** command activates only the collection of statistics. If you do not specify a type, both signaling and accounting statistics are collected.

Manually Clearing Statistics: Example

The following example shows how to clear all of the voice statistics that have been collected:

```
Router# clear voice statistics csr
```

The following example verifies the clearing by showing that all of the parameters have been set to zero:

```
Router# show voice statistics csr since-reset all

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-04-25T01:49:27Z

record_type=gw,trunk_group_id=,voice_port_id=,in_call=0,in_ans=0,in_fail=0,out_c
all=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,
out_pdd=0,in_setup_delay=0,out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0
out_disc_cc_16=0
!
record_type=ip,trunk_group_id=,voice_port_id=,in_call=0,in_ans=0,in_fail=0,out_c
all=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0,
out_disc_cc_16=0
!
record_type=pstn,trunk_group_id=,voice_port_id=,in_call=0,in_ans=0,in_fail=0,out_call=0,
out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,orig_disconn=0,
in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,in_setup_delay=0,
out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
record_type=vp,trunk_group_id=,voice_port_id=4/0/0,in_call=0,in_ans=0,in_fail=0,
out_call=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,
```

```
out_pdd=0,in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
Client Type: Voice ACCT Stats
        Start Time: 2002-04-25T01:48:04Z        End Time: 2002-04-25T01:49:30Z

methodlist=h323-1,acc_pass_criteria=0,pstn_in_pass=0,pstn_in_fail=0,pstn_out_pass=0,
pstn_out_fail=0,ip_in_pass=0,ip_in_fail=0,ip_out_pass=0,ip_out_fail=0
```

Collection of Aggregation-Level Statistics: Example

The following example shows the different levels of signaling statistics that can be collected:

```
Router# show voice statistics csr since-reset aggregation-level ?

  all          Statistics at all signaling levels
  gateway      Gateway wide
  ip           Voip interface level
  pstn         Telephony interface level
  trunk-group  Trunk group level
  voice-port   Voice-port level
```

The following example shows the collected statistics:

```
Router# show voice statistics csr since-reset aggregation-level all

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-04-25T01:50:01Z

record_type=gw,trunk_group_id=12,voice_port_id=1/0,in_call=10,in_ans=10,in_fail=0,
out_call=3,out_ans=0,out_fail=3,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
!
record_type=ip,trunk_group_id=1,voice_port_id=2/0,in_call=1,in_ans=1,in_fail=0
!
record_type=pstn,trunk_group_id=,voice_port_id=,in_call=0,in_ans=0,in_fail=0,out_call=0,
out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,o
!
record_type=vp,trunk_group_id=1,voice_port_id=4/0/0,in_call=0,in_ans=0,in_fail=0,
out_call=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
```

The following example shows the intervals at which statistics were collected:

```
Router# show voice statistics interval-tag

Current Time: 2002-04-25T01:52:11Z

INTERVAL-TAG   START TIME             END TIME
============== ====================== ======================
10             2002-04-25T01:10:00Z   2002-04-25T01:15:00Z
11             2002-04-25T01:15:00Z   2002-04-25T01:20:00Z
12             2002-04-25T01:20:00Z   2002-04-25T01:25:00Z
13             2002-04-25T01:25:00Z   2002-04-25T01:30:00Z
14             2002-04-25T01:30:00Z   2002-04-25T01:35:00Z
15             2002-04-25T01:35:00Z   2002-04-25T01:40:00Z
16             2002-04-25T01:40:00Z   2002-04-25T01:45:00Z
17             2002-04-25T01:45:00Z   2002-04-25T01:50:00Z
18             2002-04-25T01:50:00Z   2002-04-25T01:52:11Z
```

Memory Usage: Example

The following example shows the amount of memory being used for accounting and signaling CSRs by fixed interval and since a reset or reboot. It also shows the estimated memory allocated for future use.

```
Router# show voice statistics memory-usage csr

*** Voice Call Statistics Record Memory Usage ***
        Fixed Interval Option -
                CSR size: 136 bytes
                Number of CSR per interval: 9
                Used memory size (proximate): 0
                Estimated future claimed memory size (proximate): 10
        Since Reset Option -
                CSR size: 136 bytes
                Total count of CSR: 9
                Used memory size (proximate): 1224

*** Voice Call Statistics Accounting Record Memory Usage ***
        Fixed Interval Option -
                ACCT REC size: 80 bytes
                Number of ACCT REC per interval: 1
                Used memory size (proximate): 0
                Estimated future claimed memory size (proximate): 25
        Since Reset Option -
                ACCT REC size: 80 bytes
                Total count of ACCT REC: 1
                Used memory size (proximate): 80
```

Collection of Statistics Since System Reset: Examples

This section provides the following examples:

- Example of the show voice statistics csr since-reset accounting all Command, page 52
- Example of the show voice statistics csr since-reset aggregation-level all Command, page 53
- Example of the show voice statistics csr since-reset aggregation-level gateway Command, page 53
- Example of the show voice statistics csr since-reset aggregation-level ip Command, page 53
- Example of the show voice statistics csr since-reset aggregation-level pstn Command, page 54
- Example of the show voice statistics csr since-reset aggregation-level trunk-group all Command, page 54
- Example of the show voice statistics csr since-reset aggregation-level voice-port all Command, page 54

Example of the show voice statistics csr since-reset accounting all Command

The following example shows all accounting statistics:

```
Router# show voice statistics csr since-reset accounting all

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time:2002-04-25T01:50:01Z
methodlist=h323-1,acc_pass_criteria=1,pstn_in_pass=10,pstn_in_fail=0,pstn_out_pass=10
pstn_out_fail=0,ip_in_pass=15,ip_in_fail=0,ip_out_pass=15,ip_out_fail=0
```

Example of the show voice statistics csr since-reset aggregation-level all Command

The following example shows all aggregation level statistics:

```
Router# show voice statistics csr since-reset aggregation-level all

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-04-25T01:50:01Z

record_type=gw,trunk_group_id=1,voice_port_id=2,in_call=120,in_ans=120,in_fail=0
out_call=10,out_ans=10,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0,
out_disc_cc_16=0
!
record_type=ip,trunk_group_id=2,voice_port_id=3,in_call=120,in_ans=120,in_fail=0
out_call=10,out_ans=10,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0,
out_disc_cc_16=0
!
record_type=pstn,trunk_group_id=5,voice_port_id=2,in_call=150,in_ans=100,in_fail=50,
out_call=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
```

Example of the show voice statistics csr since-reset aggregation-level gateway Command

The following example shows the gateway aggregation level statistics:

```
Router# show voice statistics csr since-reset aggregation-level gateway

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-05-02T21:21:14Z

record_type=gw,trunk_group_id=1,voice_port_id=1,in_call=10,in_ans=10,in_fail=0,out_call=50
out_ans=10,out_fail=40,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,orig_disconn=0,
in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,in_setup_delay=0,
out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0,out_disc_cc_16=0
```

Example of the show voice statistics csr since-reset aggregation-level ip Command

The following example shows the IP aggregation level statistics:

```
Router# show voice statistics csr since-reset aggregation-level ip

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-05-02T21:21:27Z

record_type=ip,trunk_group_id=1,voice_port_id=1,in_call=10,in_ans=10,in_fail=0,out_call=50
out_ans=10,out_fail=40,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,orig_disconn=0,
in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,in_setup_delay=0,
out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0,out_disc_cc_16=0
```

Example of the show voice statistics csr since-reset aggregation-level pstn Command

The following example shows the PSTN aggregation level statistics:

```
Router# show voice statistics csr since-reset aggregation-level pstn

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-05-02T21:21:42Z

record_type=pstn,trunk_group_id=1,voice_port_id=1,in_call=10,in_ans=10,in_fail=0,
out_call=50,out_ans=10,out_fail=40,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,lost_pkt=0,latency=0,jitter=0,in_disc_cc_16=0,
out_disc_cc_16=0
```

Example of the show voice statistics csr since-reset aggregation-level trunk-group all Command

The following example shows the trunk-group aggregation level statistics:

```
Router# show voice statistics csr since-reset aggregation-level trunk-group all

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-05-02T21:22:08Z

record_type=pstn,trunk_group_id=15,voice_port_id=3,in_call=20,in_ans=20,in_fail=0,
out_call=10,out_ans=10,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
record_type=pstn,trunk_group_id=16,voice_port_id=4,in_call=20,in_ans=20,in_fail=0,
out_call=10,out_ans=10,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
record_type=pstn,trunk_group_id=17,voice_port_id=5,in_call=20,in_ans=20,in_fail=0,
out_call=10,out_ans=10,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
record_type=pstn,trunk_group_id=18,voice_port_id=6,in_call=20,in_ans=20,in_fail=0,
out_call=10,out_ans=10,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
```

Example of the show voice statistics csr since-reset aggregation-level voice-port all Command

The following example shows the voice-port aggregation level statistics:

```
Router# show voice statistics csr since-reset aggregation-level voice-port all

Client Type: VCSR
        Start Time: 2002-04-25T01:48:12Z        End Time: 2002-05-02T21:22:52Z

record_type=vp,trunk_group_id=a,voice_port_id=4/0/0,in_call=22,in_ans=14,in_fail=8,
out_call=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
!
record_type=vp,trunk_group_id=b,voice_port_id=4/0/1,in_call=22,in_ans=14,in_fail=8,
out_call=0,out_ans=0,out_fail=0,in_szre_d=0,out_szre_d=0,in_conn_d=0,out_conn_d=0,
orig_disconn=0,in_ans_abnorm=0,out_ans_abnorm=0,in_mcd=0,out_mcd=0,in_pdd=0,out_pdd=0,
in_setup_delay=0,out_setup_delay=0,in_disc_cc_16=0,out_disc_cc_16=0
```

Designated Server Group: Example

The following example shows that a designated server group has been configured for a broadcast method list.

```
Router# show voice statistics | begin aaa group server

aaa group server radius billing-grp
 server 1.6.37.20 auth-port 1645 acct-port 1646
 accounting acknowledge broadcast
```

Location of the FTP Server: Example

The following example shows the location of the FTP server:

```
Router# show running-config

Building configuration...

Current configuration : 5444 bytes
!
version 12.3
!
voice statistics push ftp url ftp://sgcp:sgcp@abc-pc:21//ftp_files
voice statistics push ftp max-file-size 4000000000
!
```

Maximum File Size for the Syslog Server: Example

The following example shows the maximum file size to be downloaded to the syslog server:

```
Router# show running-config

Building configuration...

Current configuration : 5444 bytes
!
version 12.3
!
voice statistics push syslog max-msg-size 4000000000
```

Maximum Duration for Storage: Example

The following configuration example shows a maximum storage of 60 minutes:

```
Router# show running-config | include voice statistics
!
voice statistics max-storage-duration minute 60
```

Cause Codes and Debug Values

This chapter contains the following information:

Additional cause code information can be found in the *Cisco IOS Debug Command Reference*.

Details of Cause Codes and Debug Values for VoIP

Use the following tables when reading debugs and the associated values within the debugs.

Q.931 Call Disconnection Causes

These are cause codes from the **debug voip ccapi inout** command.

Call Disconnection Cause Value (in Hex)	Meaning and Number (in Decimal)
CC_CAUSE_UANUM = 0x1	Unassigned number (1)
CC_CAUSE_NO_ROUTE = 0x3	No route to destination (3)
CC_CAUSE_NORM = 0x10	Normal call clearing (16)
CC_CAUSE_BUSY = 0x11	User busy (17)
CC_CAUSE_NORS = 0x12	No user response (18)
CC_CAUSE_NOAN = 0x13	No user answer (19)
CC_CAUSE_REJECT = 0x15	Call rejected (21)
CC_CAUSE_INVALID_NUMBER = 0x1C	Invalid number (28)
CC_CAUSE_UNSP = 0x1F	Normal, unspecified (31)
CC_CAUSE_NO_CIRCUIT = 0x22	No circuit (34)
CC_CAUSE_NO_REQ_CIRCUIT = 0x2C	No requested circuit (44)

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

CC_CAUSE_NO_RESOURCE = 0x2F	No resource (47)
CC_CAUSE_NOSV = 0x3F	Service or option not available, or unspecified (63)

Codec Negotiation Values

These codec negotiation values are from the **debug voip ccapi inout** command.

Negotiation Value	Meaning
codec=0x00000001	G711 ULAW 64K PCM
codec=0x00000002	G711 ALAW 64K PCM
codec=0x00000004	G729
codec=0x00000004	G729IETF
codec=0x00000008	G729a
codec=0x00000010	G726r16
codec=0x00000020	G726r24
codec=0x00000040	G726r32
codec=0x00000080	G728
codec=0x00000100	G723r63
codec=0x00000200	G723r53
codec=0x00000400	GSMFR
codec=0x00000800	G729b
codec=0x00001000	G729ab
codec=0x00002000	G723ar63
codec=0x00004000	G723ar53
codec=0x00008000	CLEAR_CHANNEL

Tone Types

Tone Types	Meaning
CC_TONE_RINGBACK 0x1	Ring tone
CC_TONE_FAX 0x2	Fax tone
CC_TONE_BUSY 0x4	Busy tone
CC_TONE_DIALTONE 0x8	Dial tone
CC_TONE_OOS 0x10	Out of service tone
CC_TONE_ADDR_ACK 0x20	Address acknowledgement tone
CC_TONE_DISCONNECT 0x40	Disconnect tone
CC_TONE_OFF_HOOK_NOTICE 0x80	Tone indicating that the phone is off-hook
CC_TONE_OFF_HOOK_ALERT 0x100	A more urgent version of CC_TONE_OFF_HOOK_NOTICE
CC_TONE_CUSTOM 0x200	Custom tone—used when you are specifying a custom tone
CC_TONE_NULL 0x0	Null tone

FAX-Rate and VAD Capabilities Values

Values	Meaning
CC_CAP_FAX_NONE 0x1	Fax disabled or not available
CC_CAP_FAX_VOICE 0x2	Voice call
CC_CAP_FAX_144 0x4	14,400 baud
CC_CAP_FAX_96 0x8	9,600 baud
CC_CAP_FAX_72 0x10	7,200 baud
CC_CAP_FAX_48 0x20	4,800 baud
CC_CAP_FAX_24 0x40	2,400 baud
CC_CAP_VAD_OFF 0x1	VAD disabled
CC_CAP_VAD_ON 0x2	VAD enabled

Internal Cause Codes for SIP and H.323

Each H.323 and SIP standard cause code accurately reflects the nature of the associated internal failure. This capability makes the H.323 and SIP call control protocols consistent with cause codes that are generated for common problems. For each internal failure, an ITU-T Q.850 release cause code is also assigned and maps the new standard categories with the Q.850 release cause code and description that is assigned to each category.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Unallocated (unassigned) number	Typical scenarios include: • The number is not in the routing table, or it has no path across the ISDN network.	1	Indicates that the destination requested by the calling user cannot be reached because the number is unassigned.
No route to specified transit network (national use)	Typical scenarios include: • The wrong transit network code was dialed. • The transit network does not serve this equipment. • The transit network does not exist.	2	Indicates that the gateway is asked to route the call through an unrecognized intermediate network.
Destination address resolution failure	Typical scenarios include: • Domain Name System (DNS) resolution failure • Invalid session target in configuration	3	CC_CAUSE_NO_ROUTE Indicates that the called party cannot be reached because the network that the call has been routed through does not serve the desired destination.
Send special information tone	Typical scenarios include: • The dialed number has a special condition applied to it.	4	Indicates that the called party cannot be reached for reasons that are of a long-term nature and that the special information tone should be returned to the calling party.
Misdialed trunk prefix (national use)	Typical scenarios include: • The wrong trunk prefix was dialed.	5	Indicates the erroneous inclusion of a trunk prefix in a called party number.
Channel unacceptable	Typical scenarios include: • Failed channel on the network.	6	Indicates that the channel most recently identified is not acceptable to the sending entity for use in this call.
Call awarded and being delivered in an established channel	Typical scenarios include: • Successful call.	7	Indicates that the user has been awarded the incoming call and that the incoming call is being connected to a channel already established to that user for similar calls.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Preemption	Typical scenarios include: • Emergency services	8	Indicates the call is being pre-empted.
Preemption. Circuit reserved for reuse	Typical scenarios include: • Emergency services	9	Indicates the call is being pre-empted and the circuit is reserved for reuse by pre-empting exchange.
Normal call clearing	Typical scenarios include: • A call participant hung up.	16	Indicates that the call is being cleared because one of the users involved with the call has requested that the call be cleared.
User busy	Typical scenarios include: • User is already using the telephone.	17	Indicates that the called party is unable to accept another call because the user busy condition has been encountered. This cause value can be generated by the called user or by the network. In the case of user determined user busy, it is noted that the user equipment is compatible with the call.
No user responding	Typical scenarios include: • The user is not answering the telephone.	18	Used when the called party does not respond to a call establishment message with either an alerting or connect indication within the time allotted. The number that is being dialed has an active D-channel, but the far end chooses not to answer.
No answer from the user (user alerted)	Typical scenarios include: • The user is not answering the telephone.	19	Used when the called party has been alerted but does not respond with a connect indication within the time allotted. This cause is not generated by Q.931 procedures but can be generated by internal network timers.
Subscriber absent	Typical scenarios include: • The user lost network connectivity or is out of range.	20	Used when a mobile station has logged off, radio contact is not obtained with a mobile station, or if a personal telecommunication user is temporarily not addressable at any user-network interface.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Call rejected	Typical scenarios include: • Subscriber has a service constraint that does not accept this call.	21	Indicates that the equipment sending this cause code does not wish to accept this call, although it could have accepted the call because the equipment sending the cause is neither busy nor incompatible. Might also be generated by the network indicating that the call was cleared because of a supplementary service constraint. The diagnostic field might contain additional information about the supplementary service and reason for rejection.
Number changed	Typical scenarios include: • A subscriber has changed their number.	22	Returned to a calling party when the called number indicated by the calling party is no longer assigned. The new called party number might be optionally included in this diagnostic field.
Redirection to a new destination	Typical scenarios include: • Call is forwarded	23	Used by a general ISUP protocol mechanism that decides that the call should be sent to a different called number.
Exchange routing error	Typical scenarios include: • Network is overloaded	25	Indicates that the destination indicated by the user cannot be reached because an intermediate exchange has released the call due to reaching a limit in executing the hop counter procedure.
Nonselected user clearing	Typical scenarios include: • Called number failure	26	Indicates that the user has not been awarded the incoming call.
Socket failure	Typical scenarios include: • Transmission Control Protocol (TCP) socket connection failure • Problem sending an H.323 SETUP • Problem sending a Session Initiation Protocol (SIP) INVITE • Send or receive error occurs on connected socket	27	CC_CAUSE_DESTINATION_OUT_OF_ORDER Indicates that the destination indicated by the user cannot be reached because the destination's interface is not functioning correctly. The signaling message cannot be delivered to the remote party.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Invalid number format	Typical scenarios include: • the caller is calling out using a network type number (enterprise) rather instead of Unknown or National.	28	Indicates that the called party cannot be reached because the called party number is not in a valid format or is not complete.
Facility rejected	Typical scenarios include: • A network service is not functioning.	29	Indicates that a supplementary service requested by the user cannot be provided by the network.
Response to STATUS ENQUIRY	Typical scenarios include: • A STATUS message is returned.	30	Included in the STATUS message when the reason for generating the STATUS message was the prior receipt of a STATUS ENQUIRY message.
Normal, unspecified	Typical scenarios include: • Normal operation	31	Reports a normal event only when no other cause in the normal class applies.
No circuit/channel available	Typical scenarios include: • No B-channels are available to make the selected call.	34	Indicates that there is no appropriate circuit or channel presently available to handle the call.
Network out of order	Typical scenarios include: • Network failure.	38	Indicates that the network is not functioning correctly and that the condition is likely to last for an extended period.
Permanent frame mode connection is out of service	Typical scenarios include: • Equipment or section failure.	39	Included in a STATUS message to indicate that a permanently established frame mode connection is out of service.
Permanent frame mode connection is operational	Typical scenarios include: • Normal operation.	40	Included in a STATUS message to indicate that a permanently established frame mode connection is operational and capable of carrying user information.
Temporary failure	Typical scenarios include: • Network failure.	41	Indicates that the network is not functioning correctly and that the condition is likely to be resolved quickly.
Switching equipment congestion	Typical scenarios include: • High traffic	42	Indicates that the switching equipment generating this cause is experiencing high traffic.
Access information discarded	Typical scenarios include: • Usually reported when the far-end ISDN switch removes some piece of information before tandem-switching a call.	43	Indicates that the network could not deliver access information to the remote user as requested.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Requested circuit/channel not available	Typical scenarios include: • Occurs during glare condition when both sides are selected top-down or bottom-up. Change the Allocation Direction so that one end is top-down and the other is bottom-up.	44	Returned when the circuit or channel indicated by the requested entity cannot be provided by the other side of the interface.
Precedence call blocked	Typical scenarios include: • Caller is busy and the priority level of active call is equal or higher than the incoming call.	46	Indicates that there are no pre-emptable circuits or that the called user is busy with a call of equal or higher pre-emptable level.
Internal resource allocation failure	Typical scenarios include: • Out of memory • Internal access to the TCP socket is unavailable	47	CC_CAUSE_NO_RESOURCE Indicates a “resource unavailable” event.
QoS error	Typical scenarios include: • Quality of service (QoS) error	49	CC_CAUSE_QOS_UNAVAILABLE Indicates that the requested QoS cannot be provided.
Requested facility not subscribed	Typical scenarios include: • The caller is trying to use a service that is not permitted.	50	Indicates that the user has requested a supplementary service that the user is not authorized to use.
Outgoing calls barred within Closed User Group (CUG)	Typical scenarios include: • Subscriber configuration contains this limitation.	53	Indicates that although the calling party is a member of a CUG for the outgoing CUG call, outgoing calls are not allowed for this member of the CUG.
Incoming calls barred within Closed User Group (CUG)	Typical scenarios include: • Subscriber configuration contains this limitation.	55	Indicates that although the called party is a member of a CUG for the incoming CUG call, incoming calls are not allowed for this member of the CUG.
Bearer capability not authorized	Typical scenarios include: • The caller is not authorized to use the bearer capability.	57	Indicates that the user has requested a bearer capability which is implemented on the equipment but the user is not authorized to use.
Bearer capability not presently available	Typical scenarios include: • A call is placed with a bearer capacity that the service provider does not have the capacity to supply.	58	Indicates that the user has requested a bearer capability which is implemented by the equipment and is currently unavailable.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Inconsistency in designated outgoing access information and subscriber class	Typical scenarios include: • Network error.	62	Indicates that there is an inconsistency in the designated outgoing access information and subscriber class.
Service or option not available, unspecified	Typical scenarios include: • Service not available.	63	Reports a service or option not available event only when no other cause in the service or option not available class applies.
Media negotiation failure	Typical scenarios include: • No codec match occurred. • H.323 or H.245 problem leading to failure in media negotiation	65	CC_CAUSE_BEARER_CAPABILITY_NOT_IMPLEMENTED Indicates that the equipment sending this cause does not support the bearer capability requested.
Channel type not implemented	Typical scenarios include: • Channel type match not found.	66	Indicates that the equipment sending this cause does not support the channel type requested.
Requested facility not implemented	Typical scenarios include: • Service type match not found.	69	Indicates that the equipment sending this cause does not support the requested supplementary service.
Only restricted digital information bearer capability is available (National use)	Typical scenarios include: • Routing error.	70	Indicates that the calling party has requested an unrestricted bearer service but that the equipment sending this cause only supports the restricted version of the requested bearer capacity.
Service or option not implemented, unspecified	Typical scenarios include: • Service not implemented.	79	Reports a service or option not implemented event only when no other cause in the service or option not implemented class applies.
Invalid call reference value	Typical scenarios include: • The far-end switch did not recognize the call reference for a message sent by the gateway.	81	Indicates that the equipment sending the cause has received a message with a call reference which is not currently in use on the user-network interface.
Identified channel does not exist	Typical scenarios include: • Fractional PRI error.	82	Indicates a call attempt on a channel that is not configured.
A suspended call exists, but this call identity does not	Typical scenarios include: • Call ID mismatch	83	Indicates a call resume has been attempted with a call identity which differs from that in use for any presently suspended calls.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Call identity in use	Typical scenarios include: • Equipment error.	84	Indicates that the network has received a call suspended request containing a call identity which is already in use for a suspended call.
No call suspended	Typical scenarios include: • Equipment error.	85	Indicates that the network has received a call resume request containing a call identity information element which does not indicate any suspended call.
Call having the requested call identity has been cleared	Typical scenarios include: • Network timeout • Call cleared by remote user.	86	Indicates that the network has received a call identity information element indicating a suspended call that has in the meantime been cleared wile suspended.
User is not a member of Closed User Group (CUG)	Typical scenarios include: • Caller is not authorized.	87	Indicates that the called user for the incoming CUG call is not a member of the specified CUG.
Incompatible destination	Typical scenarios include: • Number dialed is not capable of this type of call. • Caller is calling a restricted line in unrestricted mode. • Caller is calling a POTS phone using unrestricted mode.	88	Indicates that the equipment sending this cause has received a request to establish a call which has compatibility attributes which cannot be accommodated.
Nonexistent Closed User Group (CUG)	Typical scenarios include: • Configuration or dialing error.	90	Indicates that the specified CUG does not exist.
Invalid transit network selection (National use)	Typical scenarios include: • Network error. • Identification mismatch	91	Indicates that a transit network identification was received which is of an incorrect format.
Invalid message received error	Typical scenarios include: • An invalid message was received	95	CC_CAUSE_INVALID_MESSAGE Indicates an invalid message event.
Mandatory IE missing error	Typical scenarios include: • Mandatory Contact field missing in SIP message. • Session Description Protocol (SDP) body is missing.	96	CC_CAUSE_MANDATORY_IE_ MISSING Indicates that the equipment sending this cause code has received a message that is missing an information element (IE). This IE must be present in the message before the message can be processed.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Message type nonexistent or not implemented	Typical scenarios include: • Message type information is missing.	97	Indicates that the equipment sending this cause has received a message which is missing an information element that must be present in the message before the message can be processed.
Message not compatible with call state or message type nonexistent or not implemented	Typical scenarios include: • ISDN protocol mismatch • ISDN state machine violation	98	Indicates that the equipment sending this cause has received a message such that the procedures do not indicate that this is a permissible message to receive while in this call state.
An information element or parameter does not exist or is not implemented	Typical scenarios include: • Element mismatch	99	Indicates that the equipment sending this cause has received a message which includes information elements or parameters not recognized because the information element or parameter names are not defined or are defined but not implemented by the equipment.
Invalid IE contents error	Typical scenarios include: • SIP Contact field is present, but format is bad	100	CC_CAUSE_INVALID_IE_ CONTENTS Indicates that the equipment sending this cause code has received an IE that it has implemented. However, the equipment sending this cause code has not implemented one or more of the specific fields.
Message in invalid call state	Typical scenarios include: • An unexpected message was received that is incompatible with the call state	101	CC_CAUSE_MESSAGE_IN_ INCOMP_CALL_STATE Indicates that a message has been received that is incompatible with the call state.
Call setup timeout failure	Typical scenarios include: • No H.323 call proceeding • No H.323 alerting or connect message received from the terminating gateway • Invite expires timer reached maximum number of retries allowed	102	CC_CAUSE_RECOVERY_ON_ TIMER_EXPIRY Indicates that a procedure has been initiated by the expiration of a timer in association with error handling procedures.

Table 56 H.323 and SIP Standard Category With Corresponding Q.850 Cause Code Information

Standard Category	Standard Category Description	Q.850 Cause Code	Q.850 Release Cause Description
Parameter nonexistent or not implemented—passed on (National use)	Typical scenarios include: • Configuration mismatch.	103	Indicates that the equipment sending this cause has received a message which includes parameters not recognized because the parameters are not defined or are defined but not implemented on the equipment.
Message with unrecognized parameter discarded	Typical scenarios include: • Unrecognized parameter.	110	Indicates that the equipment sending this cause has discarded a received message which includes a parameter that is not recognized.
Protocol error, unspecified	Typical scenarios include: • Protocol error.	111	Reports a protocol error event only when no other cause in the protocol error class applies.
Internal error	Typical scenarios include: • Failed to send message to Public Switched Telephone Network (PSTN)	127	CC_CAUSE_INTERWORKING Indicates that there has been interworking with a network that does not provide causes for actions it takes. Precise cause cannot be ascertained.