

# Service Provider Features for Voice over IP

---

## Feature Summary

The Cisco voice Service Provider features include enhancements made to the functionality and configuration of both the gateway and the Voice over IP (VoIP) gatekeeper. The architecture of these features provides the Quality of Service (QoS), stability, and functionality necessary for carrier class, real-time IP communications services.

This document contains a basic description of the H.323 VoIP gateway in addition to features required to implement the applications to run VoIP in a service provider environment. The features address the service provider needs to offer security, billing, scaling, and reliability.

The VoIP gateway is a high performance H.323-compliant gateway optimized for VoIP applications. Supporting up to two T1/E1 digital channels, it connects with existing telephones and fax machines through the Public Switched Telephone Network (PSTN), key systems, and PBXs, making the process of placing calls over the IP network transparent to users.

The gateway capability allows the access device to function as an H.323 endpoint. Therefore, the gateway provides admission control, address lookup and translation, and accounting services.

The gatekeeper manages H.323 endpoints in a consistent manner, allowing them to register with the gatekeeper and to locate another gatekeeper. The gatekeeper provides logic variables for proxies or gateways in a call path to provide connectivity with the PSTN, to improve QoS, and to enforce security policies. Multiple gatekeepers may be configured to communicate with one another, either by integrating their addressing into Domain Naming System (DNS), or via Cisco IOS configuration options.

---

**Note** For complete information about the gatekeeper functionality, refer to the Cisco Product Documentation at the following Cisco Connection Online location:  
[http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113na/1137na/mcm\\_cfg.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113na/1137na/mcm_cfg.htm)

---

## Benefits

- Carrier-class voice quality
- End-to-end solutions
- Voice-enabled everywhere
- High-density voice gateways
- Quality, scalability, and services

## Restrictions

- The H.323 gateway feature and supporting software applications have the following release-specific requirements:
  - Release 11.3(6)NA2 requires VCWare version 2.4.
  - Release 11.3(7)NA requires VCWare version 2.5.
  - Release 12.0(3)T requires VCWare version 2.5
- Cisco Secure 2.1.8.4 or higher is required if H.323 accounting is being used.

## Platforms

This feature is supported on the following platforms:

- Cisco AS5300 universal access server
- Cisco 2500 series routers
- Cisco 3600 series routers

## Supported MIBs and RFCs

This feature supports the CISCO-VOICE-DIAL-CONTROL-MIB and the SNMP MIBs. The CISCO-VOICE-DIAL-CONTROL-MIB supports the QoS and QoV of VoIP calls. The SNMP MIBs are available on CCO. Refer to the online support reference listed at the following location:

- <http://www.cisco.com/public/mibs/supportlists/as5300/supportlist.html>
- Select the desired platform from support list to get them. VoIP MIBs are included in 11.3
- Significant MIBs of interest related to the Service Provider VoIP features are:
  - DIAL-CONTROL-MIB.my
  - CISCO-DIAL-CONTROL-MIB.my
  - CISCO-VOICE-DIAL-CONTROL-MIB.my
  - CISCO-VOICE-IF-MIB.my
  - CISCO-DSP-MGMT-MIB.my

## List of Terms

**AAA**—authentication, authorization, and accounting. AAA is a suite of network security services that provides the primary framework through which access control can be set up on your Cisco router or access server.

**ANI**—Automatic number identification.

**ARQ**—Admission request.

**CAS**—Channel associated signaling.

**CCAPI**—Call control applications programming interface.

**CEI**—European channelized time division multiplexing (TDM) with 32 channels of 64 kHz each.

**CLI**—Command line interface.

**CO**—Central office.

**CPE**—Customer premises equipment. Terminating equipment, such as terminals, telephones, and modems, supplied by the telephone company, installed at the customer sites, and connected to the telephone company network.

**CSM**—Call switching module.

**dial peer**—An addressable call endpoint. In Voice over IP (VoIP), there are two types of dial peers: POTS and VoIP.

**DNS**—Domain name system used to address translation to convert H.323 IDs, URLs, or e-mail IDs to IP addresses. DNS is also used to assist in the location of remote gatekeepers and to reverse-map raw IP addresses to host names of administrative domains.

**DNIS**—Dialed number identification service (the called number).

**DSP**—Digital signal processor.

**DTMF**—Dual tone multi-frequency.

**E.164**—The international public telecommunications numbering plan. A standard set by ITU-T that addresses telephone numbers.

**E&M**—Ear and mouth RBS signaling.

**endpoint**—An H.323 terminal or gateway. An endpoint can call and be called. It generates and/or terminates the information stream.

**gatekeeper**—A gatekeeper maintains a registry of devices in the multimedia network. The devices register with the gatekeeper at startup, and request admission to a call from the gatekeeper.

The gatekeeper is an H.323 entity on the LAN that provides address translation and control access to the LAN for H.323 terminals and gateways. The gatekeeper may provide other services to the H.323 terminals and gateways, such as bandwidth management and locating gateways.

**gateway**—A gateway allows H.323 terminals to communicate with non-H.323 terminals by converting protocols. A gateway is the point at which a circuit-switched call is encoded and repackaged into IP packets.

An H.323 gateway is an endpoint on the LAN that provides real-time, two-way communications between H.323 terminals on the LAN and other ITU-T terminals in the WAN, or to another H.323 gateway.

**H.323**—An International Telecommunication Union (ITU-T) standard that describes packet-based video, audio, and data conferencing. H.323 is an umbrella standard that describes the architecture of the conferencing system, and refers to a set of other standards (H.245, H.225.0, and Q.931) to describe its actual protocol.

**H.323 RAS**—Registration, admission, and status. The RAS signaling function performs registration, admissions, bandwidth changes, status and disengage procedures between the VoIP gateway and the gatekeeper.

**HSRP**—Hot Standby Routing Protocol. HSRP is a Cisco-proprietary protocol that provides a redundancy mechanism when more than one router is connected to the same segment/subnet of an Ethernet/FDDI/Token Ring network.

**ITU-T**—Telecommunication standardization sector of ITU.

**IVR**—Integrated voice response. When someone dials in, it responds with a prompt to get a personal identification number (PIN), and so on.

**LEC**—Local exchange carrier.

**LRQ**—Location request.

**MCU**—Multipoint control unit

**MF**—Multifrequency tones are made of six frequencies that provide 15 two frequency combinations for indication digits 0-9 and KP/ST signals.

**multicast**—A process of transmitting PDUs from one source to many destinations. The actual mechanism (that is, IP multicast, multi-unicast, and so forth) for this process might be different for LAN technologies.

**multipoint-unicast**—A process of transferring PDUs (Protocol Data Units) where an endpoint sends more than one copy of a media stream to different endpoints. This might be necessary in networks which do not support multicast.

**node**—An H.323 entity that uses RAS to communicate with the gatekeeper. For example, an endpoint such as a terminal, proxy, or gateway.

**PDU**—Protocol data units. Used by bridges to transfer connectivity information.

**POTS**—Plain old telephone service. Basic telephone service supplying standard single line telephones, telephone lines, and access to the PSTN.

**PSTN**—Public Switched Telephone Network. PSTN refers to the local telephone company.

**QoS**—Quality of service, which refers to the measure of service quality provided to the user.

**RAS**—Registration, Admission, and Status protocol. Protocol used between endpoints and the gatekeeper to perform management functions.

**RBS**—Robbed bit signaling.

**RRQ**—Registration request.

**SPI**—Service provider interface.

**TCL**—Tool command language.

**TDM**—Time division multiplexing. Technique in which information from multiple channels can be allocated bandwidth on a single wire based on preassigned time slots. Bandwidth is allocated to each channel regardless of whether the station has data to transmit.

**VoIP**—Voice over IP. The ability to carry normal telephone-style voice over an IP-based internet with POTS-like functionality, reliability, and voice quality. VoIP is a blanket term which generally refers to Cisco's standards based (for example, H.323) approach to IP voice traffic.

**VTSP**—Voice telephony service provider.

**zone**—A collection of all terminals (tx), gateways (GW), and Multipoint Control Units (MCU) managed by a single gatekeeper (GK). A zone includes at least one terminal, and can include gateways or multipoint control units (MCUs). A zone has only one gatekeeper. A zone may be independent of LAN topology and can be comprised of multiple LAN segments which are connected using routes or other devices.

---

**Note** For a list of other internetworking terms, see the *Internetworking Terms and Acronyms* document that accompanied your access server and is available on the Documentation CD-ROM and Cisco Connection Online (CCO).

---

## Functional Description

This section provides an overview of how gatekeepers and gateways work together.

### Gatekeeper and Gateway Functional Description

Understanding the interrelationship between gatekeepers and gateways is needed when you perform the tasks involved with the software configuration for internetworking between the two. The functionality of the major voice network components is described below.

### Gatekeepers

Gatekeepers are optional nodes that manage other nodes in an H.323 network. Other nodes communicate with the gatekeeper using the RAS protocol.

These nodes attempt to register with a gatekeeper on startup. When they wish to communicate with another endpoint, they request admission to the call, using a symbolic alias for the endpoint name such as an E.164 address or an e-mail ID. If the gatekeeper decides the call can proceed, it returns a destination IP address to the originating endpoint. This IP address cannot be the actual address of the target endpoint, and it can be an intermediate address. Finally, a gatekeeper and its registered endpoints exchange *status* information.

### Gatekeeper Zones

H.323 endpoints are grouped together in zones. Each zone has one gatekeeper that manages all of the endpoints in the zone. A zone is an administrative convenience similar to a DNS domain. (Because a zone is, by definition, the area of control of a gatekeeper, you will find the terms “zone name” and “gatekeeper name” used synonymously in this document.)

### H.323 Terminals

An H.323 terminal is an endpoint in the LAN that provides for real-time, two-way communications with another H.323 terminal or gateway. This communication consists of control, indications, audio, moving color video pictures, or data between the two terminals. A terminal may provide audio only; audio and data; audio and video; or audio, data, and video.

### Gateways

Gateways allow H.323 terminals and routers to communicate with terminals running other protocols. They provide protocol conversion between terminals and routers running different types of protocols. A gateway is the point at which a circuit-switched call is encoded and repackaged into IP packets.

An H.323 gateway is an endpoint on the LAN that provides real-time two-way communications between H.323 terminals on the LAN and other ITU-T terminals in the WAN, or to another H.323 gateway.

### Gatekeeper Functionality

The following sections describe the main features and functionality of a gatekeeper in an H.323 network:

- Zone and Subnet Configuration
- Terminal Name Registration
- Interzone Communication
- Endpoint Identification via RADIUS
- Accounting via RADIUS

## Zone and Subnet Configuration

A zone is the set of H.323 nodes controlled by a single gatekeeper. Gatekeepers co-existing on a network may be configured so that they register endpoints from different subnets.

Endpoints attempt to discover a gatekeeper, and consequently what zone they are members of, by using the RAS message protocol. The protocol supports a discovery message that may be sent multicast or unicast.

If the message is sent multicast, the endpoint registers nondeterministically with the first gatekeeper to respond. To enforce predictable behavior, where endpoints on certain subnets are assigned to specific gatekeepers, you can use the **zone subnet** command to define the subnets that constitute a given gatekeeper's zone. Any endpoint on a subnet that is not enabled for the gatekeeper will not be accepted as a member of that gatekeeper's zone. If the gatekeeper receives a unicast discovery message from such an endpoint, it will send an explicit reject, but if the message was received multicast, the gatekeeper simply ignores it.

## Terminal Name Registration

Gatekeepers recognize one of two types of terminal aliases, or terminal names:

- H.323 identifiers (IDs), which are arbitrary, case-sensitive text strings
- E.164 addresses, which are telephone numbers

If an H.323 network deploys interzone communication, each terminal should at least have a fully-qualified e-mail name as its H.323 ID. For example, bob@cisco.com. The domain name of the e-mail ID should be the same as the configured domain name for the gatekeeper of which it is to be a member. As in the previous example, the domain name would be cisco.com.

## Interzone Communication

To allow endpoints to communicate between zones, gatekeepers must be able to determine which zone an endpoint is in and locate the gatekeeper responsible for that zone. If DNS is available, you can associate a DNS domain name to each gatekeeper.

## Endpoint Identification via RADIUS

Version 1 of the H.323 specification does not provide a mechanism for authenticating registered endpoints. No credential information is passed. However, by enabling authorization, authentication and accounting (AAA) on the gatekeeper and configuring for RADIUS, you can achieve a rudimentary form of identification.

If you enable this feature, the gatekeeper attempts to use the registered aliases along with a password, and do an authentication transaction to a RADIUS server. The registration will only be accepted if RADIUS successfully authenticates the name.

The gatekeeper can be configured to use a default password for all users. It can also be configured to recognize a password separator character that allows users to piggyback their passwords onto H.323-ID registrations by using it to separate the ID and password fields.

## Accounting via RADIUS

If you enable AAA on the gatekeeper, the gatekeeper will emit an accounting record each time an endpoint registers or unregisters, or each time a call is admitted or disconnected.

The balance of this section provides detailed information on the following topics:

- Service Provider Features for Voice over IP
  - “Gatekeeper Features”
  - “Gateway Features”
  - “Gateway RAS Implementation”
  - “AAA Functionality”
  - “Interactive Voice Response”
  - “ISDN Redirect Number Support”
  - “Rotary Call Pattern”
- Gatekeeper Features
  - “HSRP Support”
  - “E.164 Addresses”
  - “The gateway technology prefix is set up using the following new commands:”
- Configuring the VoIP Gatekeeper and Gateway
  - “Sample Gatekeeper Configuration”
  - “Sample Gatekeeper Configuration”

## Service Provider VoIP Feature Overview

The Service Provider features for VoIP include enhancements made to the functionality and configuration of both the gateway and the VoIP gatekeeper. The architecture of these features provides the Quality of Service (QoS), stability, and functionality necessary for carrier class real-time IP communications services.

This document contains a basic description of the gateway and gatekeeper features required to implement the applications to run VoIP in a service provider environment. The features address the needs of the service provider to offer security, billing, scaling, and reliability.

The gateway functionality and gatekeeper functionality work in concert to provide the ITU-T H.323 infrastructure. Therefore, the two main components in the voice architecture that interoperate to enable the service provider feature set are:

- The gatekeeper, which provides H.323 scaling. See “Gatekeeper Features.”
- The VoIP gateway, which delivers carrier class voice quality. See “Gateway Features.”

Refer to “VoIP PSTN Signaling Architecture” for a graphical description of the gateway and gatekeeper internetworking functionality.

To help understand the overall implication of which features affect which portion of the internetworking environment, these features are described in two different categories:

- Gatekeeper Features

Gatekeepers can manage a zone and provide bandwidth management, and address resolution services to the gateways when present in the network.

- Gateway Features

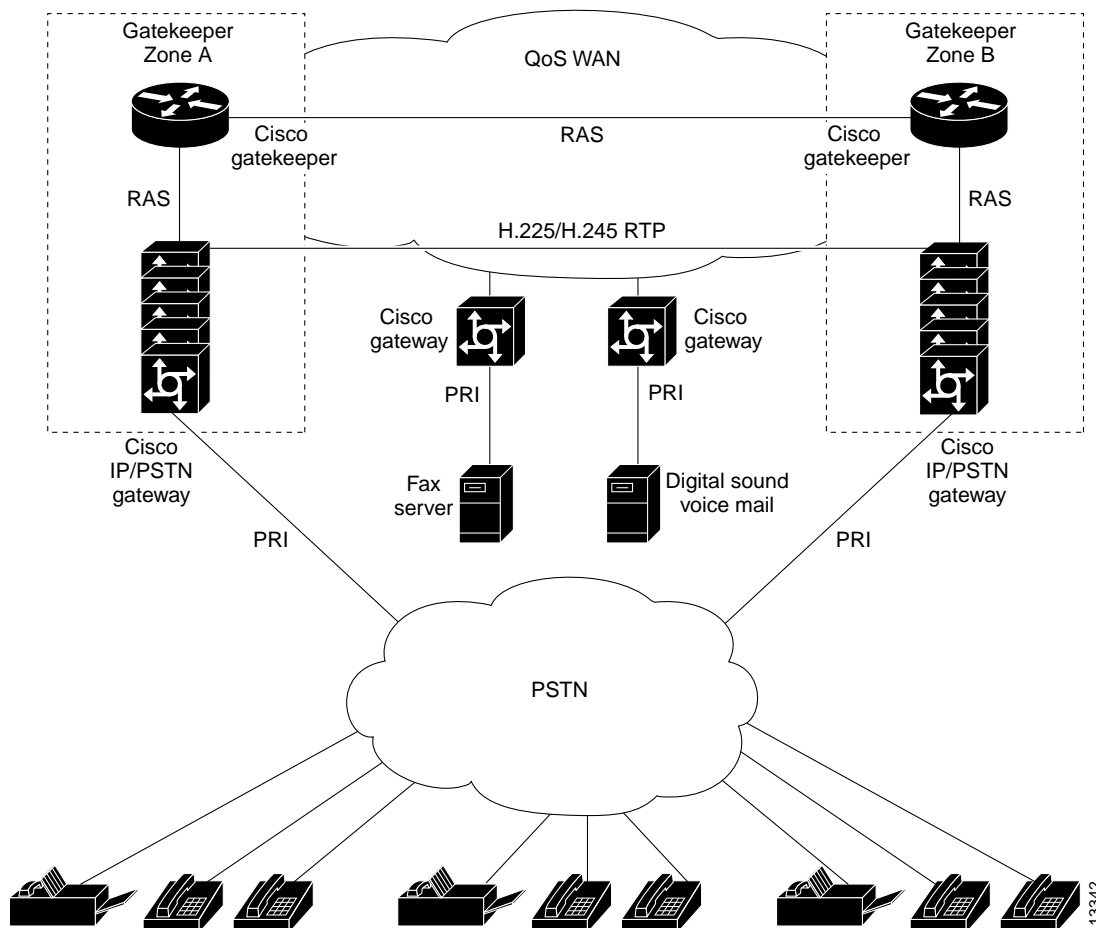
Gateways can terminate a call from the PSTN and provides user admission control using Interactive Voice Response (IVR) and provide accounting records for the call. The gateway also can direct the call to the destination, or can terminate the call from another gateway and send the call to the PSTN. The gateway (in this document) refers to the voice-capable platform with voice cards and the VoIP image. Gateways may also be referred to as VoIP gateways.

---

**Note** See the “Gatekeeper and Gateway Functional Description” for a brief description of how gateways and gatekeepers interoperate.

---

Figure 1 VoIP PSTN Signaling Architecture



## Gatekeeper Features

The gatekeeper manages H.323 endpoints in a consistent manner, allowing them to register with the gatekeeper and to locate another gatekeeper. The gatekeeper provides logic variables for proxies or gateways in a call path to provide connectivity with the PSTN, to improve QoS, and to enforce security policies. Multiple gatekeepers may be configured to communicate with one another, either by integrating their addressing into DNS, or via software configuration options.

---

**Note** For complete information about the gatekeeper functionality, refer to the Cisco Product Documentation on CCO.

---

In addition to the Cisco IOS Release 11.3NA functionality described in documentation on CCO, the gatekeeper has been enhanced with two important new features:

- HSRP support
  - Allows a standby gatekeeper to assume the role of a failed gatekeeper.
- E.164 addresses
  - Enables inter-zone call routing using E.164 addresses.

- Technology prefixes

---

**Note** Refer to the section “Gatekeeper Feature Descriptions” for the description of each feature. Refer to the section “Command Reference” for descriptions of new commands for these features.

---

The routers performing the access server functionality for the gatekeeper are:

- Cisco 2500 series modular routers
- Cisco 3600 series modular access routers

## Gateway Features

The gateway capability is the ability of the voice-capable platform to function as an H.323 endpoint. Therefore, the gateway provides the following admission control, address lookup and translation, and accounting services:

- Gateway RAS implementation
- AAA accounting
- Interactive voice response
- ISDN redirect number support
- Rotary call pattern

User configuration and implementation of these features all takes place while setting up and configuring the gatekeeper and the gateway. See the section “Configuring the VoIP Gatekeeper and Gateway.”

Descriptions of each gateway feature and the commands follow.

## Gateway RAS Implementation

RAS is a signaling function that performs registration, admissions, status, and disengage procedures between the VoIP gateway and gatekeeper.

Two new fields have been added to the dial-peer entry. The gateway relies on Cisco IOS command line interface commands, outside of gateway configuration mode, to configure handling of the AAA servers. See the section “This is an example of the configuration steps required to allow the internetworking functionality between the VoIP gateway and the gatekeeper..”

### RAS Command Fields

The following changes have been made to the gateway to enable the RAS implementation on the H.323 gateway.

Two new fields are added to the dial-peer entry:

- *technology prefix*

The *technology prefix* field is applicable to the dial-peer of the “voip” encapsulation type. This field is used to indicate to the gatekeeper the type of service that the outbound call is requesting. For a complete description of the technology prefix see the “The gateway technology prefix is set up using the following new commands:.”

- *session target RAS*

The *session target* field of the VoIP dial-peer indicates the address of the remote gateway where the call is terminated. If RAS (Request, Admission, and Status) Protocol is used, the *session target* field is used to indicate that a gatekeeper needs to be consulted in order to translate an E.164 address to an IP address. (For example, if an IP address is used: **session target ipv4:A.B.C.D**. If DNS is used: **session target dns: gateway@domain**. If Gatekeeper is used: **session target ras**.)

- Gatekeeper Discovery

A Gateway must register with a gatekeeper. If a gatekeeper is not available, the gateway will periodically attempt discovery until one is available.

When a gatekeeper is discovered, the gateway registers its aliases and call signaling address with it. At this point, the gateway is able to accept calls.

If HSRP is not configured on the gatekeeper, the gateway detects when a gatekeeper is offline. For example, if the gateway detects that a gatekeeper with which it registered is offline, the gateway will attempt to rediscover, and will accept no new RAS calls until discovery is complete. Active calls will not be affected.

- RAS State Machine

The RAS state machine operates in the Request/Reject/Confirm mode. The gateway issues a Request message to the gatekeeper. The expected response from the gatekeeper is either a Confirm message or a Reject Message.

In simple terms, the gateway employs an intelligent backoff and retry mechanism to handle transient gatekeepers or network failures.

## AAA Functionality

The AAA features are required in the VoIP gateway. The standard AAA functionality is enhanced to collect digits during the call processing process. Processes such as:

- Create a call detail record
- Authenticate based on information collected from the Interactive Voice Response (IVR) feature, or from caller identification data.

This AAA feature permits RADIUS to be used to authenticate users (typically incoming calls) on the gateway. It is normally used with IVR to check the legitimacy of a prospective gateway user based on an account number (collected by IVR) or based on answer number identification (ANI).

---

**Note** For a complete description of the standard AAA feature, refer to the CCO web site located at URL:

[http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113t/113t\\_3/aaalists.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113t/113t_3/aaalists.htm)

---

## Authentication

Authentication is based on RADIUS and is performed on the gateway (as opposed to the gatekeeper).

User account and PIN information is collected by the IVR application and is passed to the AAA interface. The AAA interface then makes a RADIUS authentication request with the given information and returns to the IVR application with status of success or failure.

RADIUS is an IETF protocol based on UDP. It functions by exchanging a set of attribute/value pairs between the client, here a VoIP gateway and a RADIUS server. Standard RADIUS server implementations include CiscoSecure, Cisco UCP, Livingston, and Merit.

### Authorization

An authenticated user is authorized. There is no authorization of specific user capabilities for the service provider voice applications.

### Accounting

Accounting uses a basic start-stop method and standard RADIUS attributes where possible. Attributes that cannot be mapped to standard RADIUS are packed into the *Acct-Session-Id* attribute field as '/' separated ASCII string.

Data items are collected for each call leg that gets created on the gateway. A call leg is the internal representation of a connection to the gateway. Each call that is made through the gateway consists of two call legs: an incoming and an outgoing call leg. The call leg information that is emitted by the gateway(s) can be correlated by their connection ID, which is the same for all call legs of a connection.

---

**Note** If you are using H.323 accounting and are also using CiscoSecure NT, you must use CiscoSecure version 2.1.8.4 or higher.

---

### Standard RADIUS Attributes

The standard RADIUS attributes supported are:

- Calling station ID
- Called station ID
- Call duration
- Received bytes
- Transmitted bytes
- Received packets
- Transmitted packets

### Nonstandard RADIUS Attributes

The nonstandard RADIUS attributes that are packed into the *Acct-Session-Id* attribute field are:

- Call leg setup time
- Gateway identifier
- Connection ID
- Call leg direction (incoming to or outgoing from the gateway)
- Call leg type (Telephony or IP)
- Call leg connect time
- Call leg disconnect time

- Call leg disconnect cause (Q.931 code)
- Remote gateway IP address

## RADIUS Accounting with Overloaded Session ID

To take advantage of standard RADIUS implementations that do not support vendor specific attributes a new method is defined that embeds the unsupported information elements in the RADIUS Acct-Session-Id. The *Acct-Session-Id* field has a maximum length of 256 characters. It is defined to contain the RADIUS account session ID which is a unique identifier that links accounting records associated with the same login session for a user. The internal representation of this field is long. Therefore, the value of this session ID can become very large as the number of sessions on a router increases.

To support additional fields, following is the format for this field:

```
<session id>/<call leg setup time>/<gateway id>/<connection id>/<call origin>/
<call type>/<connect time>/disconnect time>/<disconnect cause>/<remote ip address>
```

Field	Description
session id	The standard RADIUS account session ID
call leg setup time	The Q.931 setup time for this connection in NTP format.
gateway id	The name of the underlying gateway. Name string is of form "gateway.domain_name"
connection id	A unique global identifier used to correlate call legs that belong to the same end to end call. The field consists of 4 long words (128 bits). Each long word is displayed in hexadecimal value and separated by a space character.
call origin	Indicates origin of the call relative to the gateway. Possible values are <b>originate</b> and <b>answer</b> .
call type	Indicates call leg type. Possible values are: <b>Telephony</b> and <b>VoIP</b> .
connect time	The Q.931 connect time for this call leg in NTP format.
disconnect time	The Q.931 disconnect time for this call leg in NTP format.
disconnect cause	Documented in Q.931 specification. Can be in the range of 1 to 160.
remote ip address <sup>1</sup>	Address of the remote gateway port where the call is connected.

1. Support for **remote ip address** field was introduced with Cisco IOS Release 11.3(7)NA.

NTP time formats are displayed as: %H:%M:%S.%k %Z %tw %tn %td %Y where:

- %H is hour (00 to 23)
- %M is minutes (00 to 59)
- %S is seconds (00 to 59)
- %k is milliseconds (000 to 999)
- %Z is timezone string
- %tw is day of week (Saturday through Sunday)
- %tn is month name (January through December)

- %td is day of month (01 to 31)
- %Y is year including century (for example.,1998)

Note that because of the limited size of the *session id* string, it is not possible to embed very many information elements in it. Therefore, this feature supports only a limited set of accounting information elements. For implementations that can advantage of more information elements, Cisco's VSA implementation is recommended.

The following examples illustrate the use of the Acct-Session-Id field:

### Example 1 Start Record

```
Client-Id = 172.29.248.123
NAS-Port-Type = 0
User-Name = "4004"
Called-Station-Id = "+111"
Calling-Station-Id = "+222"
Acct-Status-Type = Start
User-Service-Type = Login-User
Acct-Session-Id = "4/23:21:14.078 UTC Sat Jul 18 1998/ak3620-1.cisco.com/859BF275
D7C80001 0 3AFF4/originate/VoIP///"
Acct-Delay-Time = 0
```

### Example 2 Stop Record

```
Client-Id = 172.29.248.123
NAS-Port-Type = 0
User-Name = "4004"
Called-Station-Id = "+111"
Calling-Station-Id = "+222"
Acct-Status-Type = Stop
User-Service-Type = Login-User
Acct-Session-Id = "4/23:21:14.078 UTC Sat Jul 18 1998/ak3620-1.cisco.com/859BF275
D7C80001 0 3AFF4/originate/VoIP/23:21:14.093 UTC Sat Jul 18 1998/23:21:23.084 UTC Sat
Jul 18 1998/4/123.45.67.890" Acct-Input-Octets = 8340
Acct-Output-Octets = 8900
Acct-Input-Packets = 417
Acct-Output-Packets = 445
Acct-Session-Time = 9
Acct-Delay-Time = 0
```

### Example 3 Update Record

```
Client-Id = 172.29.248.123
NAS-Port-Type = 0
User-Name = "4004"
Called-Station-Id = "+111"
Calling-Station-Id = "+222"
Acct-Status-Type = 3
User-Service-Type = Login-User
Acct-Session-Id = "4/21:54:17.052 UTC Mon Jul 20
1998/ak3620-1.cisco.com/BF1AC9CA 8DE60006 0 5ED24/originate/VoIP///"
Acct-Delay-Time = 0
```

## Syslog Accounting

The syslog accounting option exports the information elements associated with each call leg through a system log message, which can be captured by a syslog daemon on the network. The syslog output consists of the following:

<server timestamp> <gateway id> <message number> : <message label> : <list of AV pairs>

Field	Description
server timestamp	The timestamp created by the server when it receives the message to log.
gateway id	The name of the gateway emitting the message.
message number	The number assigned to the message by the gateway.
message label	Is a string used to identify the message category.
list of AV pairs	Is a string consisting of <attribute name> <attribute value> pairs separated by commas.

### Example

```
%VOIPAAA-5-VOIP_CALL_HISTORY:CallLegType 2,ConnectionId 300094C0 60E0F3A0 60C894C0
60C90000,
SetupTime 22:35:22.023UTC Tue Aug 11 1998, PeerAddress 999, PeerSubAddress ,
DisconnectCause 10 ,DisconnectText normal call clearing., ConnectTime 22:35:24.027 UTC
Tue Aug 11 1998,
DisconnectTime 22:35:29.028 UTC Tue Aug 11 1998, CallOrigin 1, ChargedUnits 0, InfoType
2,TransmitPackets 0, TransmitBytes 0, ReceivePackets 0, ReceiveBytes 0
```

## AAA Commands

The following Cisco IOS commands are designed for configuring the Service Provider Voice over IP AAA functionality.

- Debug Commands
- **application**
- **gw-accounting**
- **debug voip aaa**

Refer to the “Command Reference” section for more information on VoIP AAA commands.

---

**Note** For additional information about the standard AAA functionality, see Cisco IOS Release 11.3(T) Software Configuration New Features. On CCO, go to:  
[http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed\\_cr/secur\\_c/scprt1/index.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt1/index.htm).

AAA is also referred to as Security Services.

---

## Interactive Voice Response

Service provider for VoIP includes basic Interactive Voice Response (IVR) capabilities necessary to collect caller Personal Identification Number (PIN), passwords, and destination phone numbers. IVR consists of simple voice prompting and digit collection to collect information from the caller for the purpose of authenticating the user and to identify the destination.

“Simple” IVR allows the use of one of several interactive voice response scripts embedded in Cisco IOS software. The ability to modify the embedded scripts is not yet provided. However, the audio files (for the prompts) can be modified for the user.

The user receives a voice prompt instruction to enter a specific type of information, for example, a PIN number. After playing the voice prompt, the IVR feature starts the process of collecting some number of touch tones (digit collection).

The IVR application specifies a sequence of voice prompts and touch-tone collection instructions. IVR applications can be assigned to specific ports, or can be invoked based on DNIS. An IP/PSTN gateway can have several different IVR applications to accommodate many different gateway services. The IVR applications can be customized to present different interfaces to the caller. The functionality includes the ability to:

- Play out customized prompts
- Collect account numbers and PIN numbers
- Collect destination phone numbers
- Perform AAA authentication using a variety of servers
- Place calls

### IVR Application Field

The IVR application field is used to associate an application with an incoming call. This field is applicable to the dial-peer of the “pots” encapsulation type only. The application field, in the inbound dial-peer, is used to indicate the application that this incoming call should be delivered to. Examples of applications entered in this field are the IVR scripts. See the “IVR Script Description” section for more information.

### ANI Authorization

IVR scripts also utilize ANI authorization, which initially takes place with the ANI as the account number. Based on authentication of the ANI and DNIS for the call, the user is either denied service (with an appropriate voice message) or prompted for an account number and PIN if authentication fails. If authentication succeeds (or subsequent authentication with the supplied account/PIN succeeds), the user is prompted for the destination phone number and the call is placed.

See the “IVR Script Description” section for more information.

### IVR Script Description

The IVR scripts available are displayed below. Audio files are provided by Cisco, however it is recommended that you record your own audio file to be used with these scripts.

---

**Note** Use the **copy** command to copy your audio file (.au file) to your Flash and the **audio-prompt load** command will read it into RAM. See “audio-prompt load” on page 51.

---

To obtain a complete description of each script, use the **show call application voice [app-name | summary]** command and insert the desired script name in the *app-name* field. The output displays a description of each script.

A brief description of the IVR scripts follow:

- **clid\_authen**  
Authenticates the call with ANI and DNIS, collects the destination data, and makes the call.
- **clid\_authen\_collect**

Authenticates the call with ANI and DNIS, collects the destination data but if authentication fails, it collects the account and password.

- **clid\_authen\_npw**

Same as **clid\_authen**, but uses a NULL password when authenticating rather than DNIS.

- **clid\_authen\_col\_npw**

Same as **clid\_authen\_collect**, but uses a NULL password (does not use DNIS, or collect it).

- **fax\_hop\_on\_1**

This application interacts with the redialer and collects digits from it. For example, collect account number, and destination number. When placing the call to H.323, the set of fields in the call info structure are *entered*, *destination*, *account*.

- **clid\_col\_npw\_3**

Support for this script was introduced with Cisco IOS Release 11.3(7)NA.

This script is similar to **clid\_authen\_col\_npw**, but it allows two retries (3 tries total) for entering the account and password. For each of the two retries, it plays a special retry message.

- **clid\_col\_npw\_npw**

Support for this script was introduced with Cisco IOS Release 11.3(7)NA.

This is similar to **clid\_col\_npw\_3**, but it does not collect a PIN number. Instead, it uses the collected account number with a NULL password for authentication.

## Message Flow for clid\_col\_npw\_3

The message progression for the **clid\_col\_npw\_3** IVR script is exactly the same as **clid\_authen\_col\_npw** except if authentication with the collected (account and PIN number) failed, the old script just played a failure message (**auth\_failed.au**) and then hung up.

After the first two failures, the new script will play **auth\_fail\_retry.au**, and collect the account and PIN numbers again. The caller can interrupt the message by entering digits for the account number. Entering the PIN number will be prompted with the same message as the first try.

If authentication fails the third time, it will play **auth\_fail\_final.au**, and then hang up.

This script plays the following prompts:

- **flash:enter\_account.au**

Asks caller to enter account number the first time.

- **flash:auth\_fail\_retry.au**

Played after first two failures, asks user to re-enter account number.

- **flash:enter\_pin.au**

Asks caller to enter PIN number.

- **flash:enter\_destination.au**

Asks caller to enter destination phone number.

- **flash:auth\_fail\_final.au**

Informs caller that authorization failed 3 times.

Two of these audio files were first introduced with Cisco IOS Release 11.3(7)NA:

## Functional Description

---

- `auth_fail_retry.au`— “Authorization failed. Please reenter your account number followed by the pound sign (#).”
- `auth_fail_final.au`— “I’m sorry, your account number cannot be verified. Please hang up and try again.”

### Message Flow for `clid_col_npw_npw`

This call application tries to authenticate using (`ani`, `NULL`) as the (`userid`, `user password`) pair. If that fails, it collects the account number and authenticates with (`account`, `NULL`). It allows three tries for entering the account number before failing the call. If authentication succeeds, it plays a prompt to collect the destination number.

This IVR script plays the following `.au` files:

- `flash:enter_account.au`  
Asks the caller to enter account number the first time.
- `flash_auth_fail_retry.au`  
Plays after first two failures, asks user to re-enter account number.
- `flash:aumenter_destination.au`  
Asks the caller to enter destination phone number.
- `flash:auth_fail_final.au`  
Informs the caller that authorization has failed three times.

### IVR Script Samples

The supported IVR scripts are described below using the **show call application voice** *app name* command:

## show call application voice clid\_authen

```

voicelab>show call application voice clid_authen
Application clid_authen has 8 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=dnis
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State collect_dest has 3 actions and 5 events
  Do Action IVR_ACT_TONE. tone=8
  Do Action IVR_ACT_COLLECT_DIALPLAN.
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
  If Event IVR_EV_DIAL_COL_FAIL goto state collect_fail
  If Event IVR_EV_TIMEOUT do action IVR_ACT_TONE
    and goto state collect_fail count=0
State place_call has 1 actions and 4 events
  Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling= account=
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_UP goto state active
  If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State collect_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY.
    URL: flash:collect_failed.au
    allowInt=0, pContent=0x0
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY_FAILURE_TONE.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing

```

### show call application voice clid\_authen\_collect

```

voicelab>show call application voice clid_authen_collect
Application clid_authen_collect has 10 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=dnis
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state get_account
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State get_account has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_account.au
    allowInt=1, pContent=0x60E4C564
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_pin has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_pin.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern pin is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=pin
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_TIMEOUT goto state get_pin count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_pin
State authenticate has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=pin
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_TIMEOUT do nothing count=0
  If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State collect_dest has 4 actions and 8 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_destination.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_DIALPLAN.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PLAY_COMPLETE do nothing

```

```

    If Event IVR_EV_ABORT goto state collect_dest
    If Event IVR_EV_TIMEOUT goto state collect_dest count=0
    If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
    If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
    If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling=    account=
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_CALL_UP goto state active
    If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY_FAILURE_TONE.
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing

sblabl15>show call application voice clid_authen_collect
Application clid_authen_collect has 10 states with 0 calls active
State start has 1 actions and 5 events
    Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=dnis
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
        and goto state start
    If Event IVR_EV_AAA_SUCCESS goto state collect_dest
    If Event IVR_EV_AAA_FAIL goto state get_account
State end has 1 actions and 3 events
Do Action IVR_ACT_END.
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
        and do nothing
State get_account has 4 actions and 7 events
Do Action IVR_ACT_PLAY.
    URL: flash:enter_account.au
    allowInt=1, pContent=0x60E4C564
Do Action IVR_ACT_ABORT_KEY. abortKey=*
Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
        patName=account
    If Event IVR_EV_ABORT goto state get_account
    If Event IVR_EV_PLAY_COMPLETE do nothing
    If Event IVR_EV_TIMEOUT goto state get_account count=0
    If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_pin has 4 actions and 7 events
Do Action IVR_ACT_PLAY.
    URL: flash:enter_pin.au
    allowInt=1, pContent=0x0
Do Action IVR_ACT_ABORT_KEY. abortKey=*
Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
Do Action IVR_ACT_COLLECT_PATTERN. Pattern pin is .+

```

## Functional Description

---

```
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=pin
If Event IVR_EV_PLAY_COMPLETE do nothing
If Event IVR_EV_ABORT goto state get_account
If Event IVR_EV_TIMEOUT goto state get_pin count=0
If Event IVR_EV_PAT_COL_FAIL goto state get_pin
State authenticate has 1 actions and 5 events
Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=pin
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_AAA_SUCCESS goto state collect_dest
If Event IVR_EV_TIMEOUT do nothing count=0
If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State collect_dest has 4 actions and 8 events
Do Action IVR_ACT_PLAY.
    URL: flash:enter_destination.au
    allowInt=1, pContent=0x0
Do Action IVR_ACT_ABORT_KEY. abortKey=*
Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
Do Action IVR_ACT_COLLECT_DIALPLAN.
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_PLAY_COMPLETE do nothing
If Event IVR_EV_ABORT goto state collect_dest
If Event IVR_EV_TIMEOUT goto state collect_dest count=0
If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling= account=
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_CALL_UP goto state active
If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY_FAILURE_TONE.
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
```

## show call application voice clid\_authen\_npw

```

voicelab>show call application voice clid_authen_npw
Application clid_authen_npw has 8 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=NULL
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State collect_dest has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_destination.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_DIALPLAN.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state collect_dest
  If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
  If Event IVR_EV_DIAL_COL_FAIL goto state collect_fail
  If Event IVR_EV_TIMEOUT goto state collect_fail count=0
State place_call has 1 actions and 4 events
  Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling= account=
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_UP goto state active
  If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State collect_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY.
    URL: flash:collect_failed.au
    allowInt=0, pContent=0x0
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY_FAILURE_TONE.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing

```

show call application voice clid\_authen\_col\_npw

```

voicelab>show call application voice clid_authen_col_npw
Application clid_authen_col_npw has 10 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=NULL
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state get_account
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State get_account has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_account.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_pin has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_pin.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern pin is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=pin
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_TIMEOUT goto state get_pin count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_pin
State authenticate has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=pin
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_TIMEOUT do nothing count=0
  If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State collect_dest has 4 actions and 8 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_destination.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_DIALPLAN.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PLAY_COMPLETE do nothing

```

```

    If Event IVR_EV_ABORT goto state collect_dest
    If Event IVR_EV_TIMEOUT goto state collect_dest count=0
    If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
    If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
    If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
  Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling=      account=
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_UP goto state active
  If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY_FAILURE_TONE.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
```

### show call application voice fax\_hop\_on\_1

```

voicelab>show call application voice fax_hop_on_1
Application fax_hop_on_1 has 8 states with 0 calls active
State start has 2 actions and 5 events
  Do Action IVR_ACT_PLAY.
    URL: flash:redialer_tone.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern init_seq is \*\*
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_PAT_COL_SUCCESS goto state collect_account
    patName=init_seq
  If Event IVR_EV_PLAY_COMPLETE do nothing
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State collect_account has 2 actions and 3 events
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state init_seq
    patName=account
State init_seq has 1 actions and 3 events
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern init_seq is \*\*
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state collect_dest
    patName=init_seq
State collect_dest has 2 actions and 3 events
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern destination is .+
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=destination
State authenticate has 1 actions and 4 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=NULL
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_AAA_SUCCESS goto state place_call
  If Event IVR_EV_TIMEOUT do nothing count=0
State place_call has 1 actions and 3 events
  Do Action IVR_ACT_PLACE_CALL.
    destination=dnis called=account
    calling=destination    account=account
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_UP goto state active
State active has 0 actions and 2 events
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing

```

## show call application voice clid\_col\_npw\_3

```

voicelab# show call app voice clid_col_npw_3
Application clid_col_npw_3 has 12 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=NULL
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state get_account
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State get_account has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL:flash:enter_account.au
    allowInt=1, pContent=0x60F66AD0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_account_retry has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL:flash:auth_fail_retry.au
    allowInt=1, pContent=0x60F87454
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_pin has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL:flash:enter_pin.au
    allowInt=1, pContent=0x60F6E178
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern pin is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=pin
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_TIMEOUT goto state get_pin count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_pin
State authenticate has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=pin

```

## Functional Description

---

```

    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_AAA_SUCCESS goto state collect_dest
    If Event IVR_EV_TIMEOUT do nothing count=0
    If Event IVR_EV_AAA_FAIL goto state fail_count
State fail_count has 1 actions and 5 events
Do Action IVR_ACT_COUNT. maxCount = 3
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_COUNT_LIMIT goto state authenticate_fail
    If Event IVR_EV_COUNT_OK goto state get_account_retry
    If Event IVR_EV_TIMEOUT do nothing count=0
State collect_dest has 4 actions and 8 events
Do Action IVR_ACT_PLAY.
    URL:flash:enter_destination.au
    allowInt=1, pContent=0x60F75C10
Do Action IVR_ACT_ABORT_KEY. abortKey=*
Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
Do Action IVR_ACT_COLLECT_DIALPLAN.
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_PLAY_COMPLETE do nothing
    If Event IVR_EV_ABORT goto state collect_dest
    If Event IVR_EV_TIMEOUT goto state collect_dest count=0
    If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
    If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
    If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling=         account=
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_CALL_UP goto state active
    If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY.
    URL:flash:auth_fail_final.au
    allowInt=0, pContent=0x60F92304
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY_FAILURE_TONE.
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing

```

## show call application voice clid\_col\_npw\_npw

```

voicelab# show call app voice clid_col_npw_npw
Application clid_col_npw_npw has 11 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=NULL
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state get_account
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State get_account has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL:flash:enter_account.au
    allowInt=1, pContent=0x60F66AD0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_account_retry has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL:flash:auth_fail_retry.au
    allowInt=1, pContent=0x60F87454
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State authenticate has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=NULL
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_TIMEOUT do nothing count=0
  If Event IVR_EV_AAA_FAIL goto state fail_count
State fail_count has 1 actions and 5 events
  Do Action IVR_ACT_COUNT. maxCount = 3
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_COUNT_LIMIT goto state authenticate_fail
  If Event IVR_EV_COUNT_OK goto state get_account_retry
  If Event IVR_EV_TIMEOUT do nothing count=0
State collect_dest has 4 actions and 8 events
  Do Action IVR_ACT_PLAY.
    URL:flash:enter_destination.au

```

```
        allowInt=1, pContent=0x60F75C10
Do Action IVR_ACT_ABORT_KEY. abortKey=*
Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
Do Action IVR_ACT_COLLECT_DIALPLAN.
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_PLAY_COMPLETE do nothing
If Event IVR_EV_ABORT goto state collect_dest
If Event IVR_EV_TIMEOUT goto state collect_dest count=0
If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling=      account=
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_CALL_UP goto state active
If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY.
    URL:flash:auth_fail_final.au
    allowInt=0, pContent=0x60F92304
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY_FAILURE_TONE.
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
```

## Fax Hop On/Off

Fax hop on/off is a specialized IVR application to support the use of redialer boxes in fax applications. Redialers are small units that connect between a fax machine and a telephone line, intercept the phone number dialed by the fax machine, and place an outgoing call to another phone number (in this case, that of the voice gateway), and then forward the destination number intercepted from the fax machine to the gateway when prompted. Optionally, an account number can be included to identify the caller's organization for authentication and billing purposes.

## IVR Commands

New Cisco IOS commands are available to deal with IVR functionality. These commands are entered when the dial peer is being configured. The commands are as follows:

- **application**
- **audio-prompt load**
- **show call application voice**
- **debug voip ivr**
- **debug voip ccapi**
- **debug cch323 h225**

See the "Command Reference" section for detailed information on these commands.

## ISDN Redirect Number Support

This feature supports the redirecting call feature of the VoIP gateway. The *redirecting number* is an optional field of the Q.931 *Setup* message.

When a local exchange carrier (LEC) switch detects an incoming call that is destined for a busy or nonanswering party, the switch formulates a Q.931 *Setup* message with *redirecting number* field set to the original destination number, and sends it to the gateway. The *called party number* of the setup message will be set to one of the service access numbers dialed number identification service (DNIS) of the gateway.

If a redirect number is present on an incoming call, then it is used in place of the destination number (DNIS).

### Dial Peer Configuration Restrictions for ISDN Redirect

The dial peer configuration for ISDN redirect involves setting up two audio scripts.

- Incoming Dial Peer
- Outgoing Dial Peer

#### Incoming Dial Peer

To process incoming ISDN voice calls, incoming dial peers need to be configured. The dialed number identification service (DNIS) number of the incoming call is used to match the DNIS number field of the incoming dial peer. The direct-inward-dial flag of the dial peer determines whether a second dial tone is given to the caller to collect the target destination number. For this Service Provider feature, the DNIS is set to the access phone number of the gateway, and the direct-inward-dial flag is set to TRUE.

#### Outgoing Dial Peer

The outgoing dial peer is selected based on the DNIS number of the incoming call. The outgoing dial peer indicates the session target of the outgoing call.

### ISDN Redirect Call Flow Scenario

The following scenario describes how an ISDN call is redirected.

- call placed from phone A to phone B
- phone B is busy so call rerouted by switch to a voice-capable gateway
- incoming call to a voice-capable platform in the call setup message includes calling/called and RDN info
- calling = phone A
- called = gateway
- RDN = phone B
- A voice-capable platform matches outgoing dial-peer destination pattern against the called (in this case GW) number
- A voice-capable platform places call to remote IP endpoint/gateway with the following
  - calling = phone A

- called =phone B
- remote endpoint/gateway matches outgoing (telephony) dial-peer with destination pattern B.

## Rotary Call Pattern

The Rotary Calling Pattern feature provides the ability to route an incoming call arriving via a telephony interface back out via another telephony interface under certain circumstances. This is primarily used to provide reliable service during network failures. Call establishment via *Rotary Call Pattern* will be supported via rotary group support of dial peers, where multiple dial peers may match a given destination phone number and will be selected in sequence.

Before Cisco IOS Release 11.3(6)NA2, if you wanted the system to search through a number of destinations, when a call came into a given number, you needed to configure those dial peers with the same destination pattern. Now with the Rotary Call Pattern feature, if you want the destinations to be tried in a certain order, you can assign preference. Use the **preference** command when configuring the dial peers to reflect the preferred order.

## Rotary Feature Functionality

If there are several dial peers that match a particular destination pattern, the system attempts to place a call to the one with the highest preference. If the call cannot be completed because of a system outage, for example, the gatekeeper or gateway cannot be contacted, the Rotary feature performs the following tasks:

- Lists all the conditions where this instance occurs
- Retries the call to the next highest *preference* dial peer
- Continues until no more matching dial peers are found

If there are equal priority dial peers, the order is determined randomly.

---

**Note** The hunting algorithm precedence is configurable. See the **preference** command in the “Command Reference” section.

---

## Gatekeeper Feature Descriptions

The two main features of the gatekeeper that have been enhanced to support internetworking with the gateway are:

- HSRP Support
- E.164 Addresses

Brief descriptions of these features follow.

---

**Note** See the “Command Reference” section for a complete description of the new gatekeeper Cisco IOS commands used to configure the gatekeeper features.

---

### HSRP Support

Gatekeeper HSRP (Hot Standby Router Protocol) support consists of elements in both the gateway and gatekeeper functions in the router. The gateway periodically retries its registration when it detects a possible gatekeeper failure, in order to register itself with the backup gatekeeper. Although it is a backup, the gatekeeper operates in a passive mode in which it does not accept registrations, and becomes active when it is notified by HSRP that it will become the primary gatekeeper.

---

**Note** See the “Command Reference” section for a complete description of the new gatekeeper Cisco IOS commands used to configure the gatekeeper features.

---

### Configuring Gatekeepers for Hot Standby

Gatekeepers can be configured to use HSRP so that when one gatekeeper fails, the standby gatekeeper assumes its role.

Select one interface on each gatekeeper which will serve as the HSRP interface and configure these two interfaces so that they belong to the same HSRP group and have different priorities. The one with the higher priority will be the active gatekeeper; the other assumes the standby role. Make a note of the virtual HSRP IP address shared by both.

Configure the gatekeepers so that this HSRP virtual IP address is the RAS address for all local zones. Make sure that the gatekeeper mode configurations on both routers are identical.



#### Tips

- If you configure your endpoints and gateways so that they use a specific gatekeeper address (rather than multicasting) then you should use the HSRP virtual IP address as the gatekeeper's address.
- You can also leave the endpoints and gateways to find the gatekeeper by multicasting. In standby status, the secondary gatekeeper will neither receive nor respond to multicast or unicast requests.
- Remember that gatekeeper failover will not be completely transparent to endpoints and gatekeepers. When the standby gatekeeper takes over, it does not have the state of the failed gatekeeper.
- If an endpoint which had registered with the failed gatekeeper now makes a request to the new gatekeeper, the gatekeeper responds with a reject, which indicates that it does not recognize the endpoint. If this occurs, the must endpoint must register with the new gatekeeper before it can continue H.323 operations.

## E.164 Address Support

There are two types of addresses used in H.323 destination calls:

- H.323-ID (a character string)
- E.164 (a string containing phone-keypad characters)

The Cisco IOS Release 11.3(2)NA software feature Multimedia Conference Manager dealt primarily with H.323-ID addressing in interzone calls. With the new prefix commands, the administrator can now also configure interzone routing when calls are made using E.164 addresses.

---

**Note** To refer to the Multimedia Conference Manager, see the Cisco CCO product documentation web site at the following URL:  
[http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113na/1137na/mcm\\_cfg.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113na/1137na/mcm_cfg.htm)

---

### H.323 ID Addresses

When using H.323-ID addresses, interzone routing is handled through the use of domain names. For example, to resolve “bob@cisco.com,” the source endpoint’s gatekeeper finds the gatekeeper for “cisco.com,” and sends it the Location Request (LRQ) for target address “bob@cisco.com.” The destination gatekeeper looks in the registration database, sees “bob” registered, so returns the appropriate IP address to get to bob.

### E.164 Addresses

When using E.164 addresses, call routing is handled through means of zone prefixes and “gateway-type” or technology prefixes.

#### Zone Prefixes

Zone prefixes (typically area codes) serve the same purpose as the domain names in the H.323-ID address space.

For instance, if our local gatekeeper has been configured with the knowledge that zone prefix “212.....” (that is, any address beginning “212” and followed by 7 arbitrary digits) is handled by gatekeeper “gk-ny”:

```
router(config-gk)# zone prefix gk-ny 212.....
```

then when the local gatekeeper is asked to admit a call to destination address “2125551111”, it knows to send the Location Request to gk-ny.

However, when the query gets to gk-ny, gk-ny still needs to resolve the address so that the call can be sent to its final destination. There may actually be an H.323 endpoint that has registered with gk-ny with that E.164 address, in which case gk-ny will return the IP address for that endpoint. However, the probability is that the E.164 address belongs to a non-H.323 device (for example, a telephone or an H.320 terminal). Because non-H.323 devices do not register with gatekeepers, gk-ny has no knowledge of whom the address belongs to. It needs to be able to select a gateway which can be used to reach the non-H.323 device. This is where the technology prefixes (or “gateway-type”) becomes useful.

## Technology Prefixes

The network administrator selects technology prefixes (tech-prefixes) to denote different types or classes of gateways. The gateways are then configured to register with their gatekeepers with these prefixes. For example, voice gateways may register with tech-prefix “1#,” H.320-gateways with tech-prefix “2#,” voicemail-gateways with “3#,” and so on. More than one gateway may register with the same type prefix. When that happens, the gatekeeper makes a random selection among gateways of the same type.

A caller, who knows the type of device he or she is trying to reach, can now prepend a tech-prefix to the destination address to indicate the type of gateway to use to get to the destination.

### Example:

The caller might ask for 1#2125551111 if they know that the address 2125551111 is for a telephone and the tech-prefix for voice gateways is “1#.” When the voice gateway receives the call for 1#2125551111, it strips off the tech-prefix and bridges the next leg of the call to the telephone at 2125551111.

In cases where the call scenario is:

```
telephone -----> voice-gw1 -----> voice-gw2 ----->telephone
      (PSTN)           (H.323)           (PSTN)
```

voice-gw1 can be configured (using the dial-peer command) to prepend the voice tech-prefix “1,” so that the use of *technology prefixes* is completely transparent to the caller.

---

**Note** Technology prefixes are transmitted (as part of the called\_number) to the destination gateway. Therefore, the customer must configure the dial peers at the destination gateway to match on the technology prefix.

---

The gateway technology prefix is set up using the following new commands:

- **gw-type-prefix**
- **show gatekeeper gw-type-prefix**

There are a couple of interesting technology prefix features in the implementation of the **gw-type-prefix** command.

- The ability to define a particular gw-type prefix as being the default gateway type to be used for unresolvable addresses
- The ability to force a tech-prefix hop-off to a particular zone

---

**Note** See the “Command Reference” section for a description of the technology prefix related commands.

---

## Default Technology

If the majority of calls hop off on a particular type of gateway, the gatekeeper can be configured to use that type of gateway as the default type, so that callers no longer have to prepend a tech-prefix on the address. For example, if what you use in your network are mostly voice gateways, and you have configured all your voice gateways to register with **tech-prefix 1**, you can configure your gatekeeper to use “1#” gateways as the default:

```
router(config-gk)# gw-type-prefix 1# default-technology
```

Now a caller no longer needs to prepend “1#” to use a voice gateway. Any address that does not contain an explicit tech-prefix will be routed to one of the voice gateways which registered with “1#.”

With this default-technology definition, suppose a caller asks the gatekeeper for admission to 2125551111. If the local gatekeeper does not recognize the zone prefix as belonging to any remote zone, it will route the call to one of its local “1#” voice gateways, so the call hops off locally. However, if it knows that gk-ny handles the 212 area code, it sends a Location Request for 2125551111 to gk-ny. This requires that gk-ny also be configured with some default gateway type prefix, and that its voice gateways be registered with that prefix.

### Force Technology Prefix Hop-off

The other gateway-type feature is the ability to force a hop-off to a particular zone. Normally, when an endpoint or gateway makes a call-admission request to its gatekeeper, the gatekeeper resolves the destination address by first looking for the tech-prefix. When that is matched, the remaining string is compared against known zone prefixes. If the address resolves to a remote zone, the entire address, including both technology and zone prefixes, is sent to the remote gatekeeper in a Location Request. That remote gatekeeper then uses the tech-prefix to decide which of its gateways to hop off.

The zone-prefix determines the routing to a zone, when there, the tech-prefix determines the gateway in that zone. See the “zone prefix” section for a description of the command.

This behavior can be overridden by associating a forced hop-off zone with a particular tech-prefix. What this does is force the call to the specified zone, regardless of what the zone-prefix in the address is.

A hypothetical example to demonstrate a forced hop-off follows: you are in the 408 area code (San Jose) and you want calls to the 212 area code to hop-off in New York via VoIP to save costs but you do not have any gateway in New York. However you do have an H.323 gateway in Denver and to hop-off from Denver is cheaper than to locally hop-off from San Jose. In this case, you would define the gateway-type prefix 212 for H.323 gateways to always be forced to the Denver zone.

## Configuring VoIP

This section describes the following VoIP configuration tasks:

- Configuring the VoIP Gatekeeper and Gateway
- AAA Configuration

### Configuring the VoIP Gatekeeper and Gateway

All of the Service Provider for VoIP features are configured when setting up the gatekeeper and gateway internetworking configuration. The example configurations provided in this documentation are supplied for reference only.

### Configuration Prerequisites

Before you can configure your Cisco Service Provider features, ensure you have the following installed:

- A voice-capable platform

- VoIP feature card
- Cisco IOS Release 12.0(3)T and VCWare Version 2.5.

These configuration tasks are based on the assumption that all necessary tasks and configurations have been performed as described in the document *Voice Over IP for the Cisco AS5300 Universal Access Server Software Configuration Guide*, which includes the following topics:

- Voice over IP for the Cisco AS5300 Configuration Overview
- Voice over IP for the Cisco AS5300 Configuration Examples
- Voice over IP for the Cisco AS5300 Commands
- Voice over IP for the Cisco AS5300 Debug Commands

**Note** Voice over IP information for the Cisco AS5300 is located on the CCO web site at the following URL:

<http://www.cisco.com/univercd/cc/td/doc/product/access/nubuvoip/voip5300/index.htm>.

This is an example of the configuration steps required to allow the internetworking functionality between the VoIP gateway and the gatekeeper.

See the “Command Reference” section for detailed information about the commands used in this configuration.

Step	Command	Purpose
1	5300# <b>config term</b> 5300-1(config)# <b>gateway</b>	Enables the gateway. To disable the gateway, use the <b>no gateway</b> command in configuration mode. If active calls exist, the gateway cannot be disabled.
2		Configure the interface. Only one interface is allowed to be the gateway interface.  The user can select either the interface that is connected to the gatekeeper, or a loopback interface. The interface that is connected to the gatekeeper is usually a LAN interface (That is, Fast Ethernet, Ethernet, FDDI, or Token-Ring). In this example, the fast Ethernet interface is used.
3	5300-1(config)# %SYS-5-CONFIG_I: Configured from console by console 5300-1(config)# <b>int fa0</b> 5300-1(config-if)# <b>h323 voip interface</b>	Enable the interface as an H.323 Gateway VoIP Interface.  An interface is identified as a Gateway VoIP interface when the following commands are entered in the interface in configuration mode.  To indicate that this interface is no longer a gateway VoIP interface, use the <b>no</b> prefix.

Step	Command	Purpose
4		<p>Specify an H.323 ID for this interface.</p> <p>An H.323 ID specifies the ID used by this gateway when this gateway communicates with the gatekeeper. Usually, this H.323 ID is the name given to the gateway with the gatekeeper domain name appended.</p> <p>For example, if the name of this gateway is <b>voip1</b>, and this gateway is in the domain called <b>vm1lab</b>, then the H.323 ID will be <b>voip1@vm1lab</b>.</p>
5	<pre>5300-1(config)# %SYS-5-CONFIG_I: Configured from console by console 5300-1(config)# int fa0 5300-1(config-if)# h323-gateway voip h323-id voip1@vm1lab</pre>	<p>Specifies the gateway H.323 ID. To disable the H.323 ID, use the <b>no</b> prefix.</p>
6	<pre>5300-1(config)# %SYS-5-CONFIG_I: Configured from console by console 5300-1(config)# int fa0 5300-1(config-if)# h323-gateway voip tech-prefix n#</pre>	<p>Specifies a <i>technology prefix</i>.</p> <p>A <i>technology prefix</i> is used to identify a type of service that this gateway is capable of providing. If a gateway is capable of handling multiple services, specify each service with a <b>tech-prefix</b> command.</p> <p>To disable a <i>technology prefix</i>, use the <b>no</b> prefix.</p>
7	<pre>5300-1(config-if)# h323 voip id &lt;name_of_the_gatekeeper&gt; [multicast   ipaddr A.B.C.D [port &lt;number&gt;]]</pre>	<p>Identifies a gatekeeper. The specified gatekeeper ID must exactly match the gatekeeper ID in the gatekeeper configuration.</p> <p>This is an optional command to identify the name of the gatekeeper that this gateway wants to communicate to. If this command is not given, the gateway will use multicast to find the gatekeeper.</p>
8	<pre>5300-1# sh gateway Gateway voip1@vm1lab is registered to gatekeeper gk1.vm1lab</pre>	<p>Finds the current registration status of the gateway.</p> <p>If the gateway is not registered with any gatekeeper, the <b>show</b> command will return: Gateway voip1@vm1lab is not registered to any gatekeeper</p>

Step	Command	Purpose
9	<pre>300-1# sh dial-peer vo 1234 VoiceOverIpPeer1234 tag = 1234, destination-pattern = 1234', answer-address = ', group = 1234, Admin state is up, Operation state is up, incoming called-number = ', connections/maximum = 0/unlimited, application associated: type = voip, session-target = ras', technology prefix: 8# ip precedence = 0, UDP checksum = disabled, session-protocol = cisco, req-qos = controlled-load, acc-qos = best-effort, fax-rate = voice, codec = g729r8, Expect factor = 10, Icpif = 30, VAD = enabled, Poor QOV Trap = disabled, Connect Time = 0, Charged Units = 0, Successful Calls = 0, Failed Calls = 0, Accepted Calls = 0, Refused Calls = 0, Last Disconnect Cause is "", Last Disconnect Text is "", Last Setup Time = 0.</pre>	<p>Determines the dial peer change.</p> <p>This is an example of a dial-peer that uses RAS.</p>

## Miscellaneous Notes

The differences between this dial-peer and a normal VoIP dial-peer are:

- The session target for this dial-peer uses the keyword **ras**.
- The *technology prefix* is indicated.

## AAA Configuration

The Cisco IOS software AAA accounting user interface can be configured to use the H.323 method as follows:

The **authentication** command line creates a method list named H.323 with RADIUS being its only member.

Also note that the accounting command line looks like a regular RADIUS accounting command line for connection accounting. Connection accounting has to be globally enabled using this command line. Start-stop or stop only methods may be used.

Step	Command	Purpose
1	<pre>5300&gt; enable  Password: &lt;password&gt;  5300#</pre>	<p>Enter enable mode.</p> <p>Enter the password.</p> <p>You have entered enable mode when the prompt changes to 5300#.</p>
2	<pre>5300# config term Enter configuration commands, one per line. End with CNTL/Z. 5300(config)#</pre>	<p>Enter global configuration mode. You have entered global configuration mode when the prompt changes to 5300(config)#.</p>
3	<pre>5300(config)# aaa new-model</pre>	<p>Initiates the AAA script.</p>

Step	Command	Purpose
4	<code>5300(config)# aaa authentication login h323 radius</code>	Configures the router to use the H.323 method list for authentication purposes.
5	<code>5300(config)# aaa accounting connection h323 start-stop radius</code>	Tells the system to use connection based accounting and the H.323 service.
6	<code>5300(config)# radius-server host 171.69.58.104 auth-port 165 acct-port 1646</code>	This command sets the server host IP address, and the ports for both the authentication service and the accounting service.
7	<code>5300(config)# radius-server key testing123</code>	Tests the connection accounting service.
8	<code>5300(config)# end</code>	Ends the configuration session.



### Tips

- Because Cisco security authenticates based on account number, RADIUS is required for the redialer fax application.
- RADIUS is turned on globally, but is only used for services if it is so programmed. Fax hop-on does use it, and a regular session application does not.

---

## Configuration Examples

This section includes the following examples of gateway and gatekeeper configuration:

- Sample AAA Configuration
- Sample Gatekeeper HSRP Configuration
- Sample Gatekeeper Configuration

## Sample AAA Configuration

The Cisco IOS software AAA accounting user interface can be configured to use the H.323 method as follows:

The **authentication** command line creates a method list named H.323 with RADIUS being its only member.

Also note that the accounting command line looks like a regular RADIUS accounting command line for connection accounting. Connection accounting has to be globally enabled using this command line. Start-stop or stop only methods may be used. The following example shows authentication and accounting:

```
aaa new-model
aaa authentication local-override
aaa authentication login default radius
!
gw-accounting h323
!
radius-server host 10.90.1.1 auth-port 1645 acct-port 1646
radius-server key xxx
```

## Sample Gatekeeper HSRP Configuration

In this example, Ethernet 0 is used as the HSRP interface on both gatekeepers and the gatekeeper is configured using either a Cisco 3620 or Cisco 3640 modular access router. The three stages of the sample gatekeeper HSRP configuration are displayed in the proceeding tables:

### Task List

- Configure the Primary Gatekeeper
- Configure the Backup Gatekeeper
- Configure Identical Gatekeeper Modes on Both Gatekeeper 1 and Gatekeeper 2

---

**Note** Enter configuration commands, one per line. End with CNTL/Z, or press the End key.

---

### Configure the Primary Gatekeeper

In this portion of the example, the interface is e0, the HSRP standby group is 1 sharing a virtual address of 172.21.127.55, hello timers are set to 5 seconds, the hold timer is set to 15 seconds, and the standby priority is 110.

```
S36xx# config term
36xx(config-if)# int e0
36xx(config-if)# standby 1 ip 172.21.127.55
36xx(config-if)# standby 1 timers 5 15
36xx(config-if)# standby 1 priority 110
36xx(config-if)# end
```

### Configure the Backup Gatekeeper

In this portion of the example, the interface is e0, the HSRP standby group is 1 sharing a virtual address of 172.21.127.55, hello timers are set to 5 seconds, the hold timer is set to 15 seconds, and the standby priority is 110.

```
36xx# config term.
36xx(config-if)# int e0
36xx(config-if)# standby 1 ip 172.21.127.55
36xx(config-if)# standby 1 timers 5 15
36xx(config-if)# end
```

---

**Note** The configurations are identical except that gk2 has no standby priority configuration, so it assumes the default priority of 100. This means that gk1 has a higher priority.

---

### Configure Identical Gatekeeper Modes on Both Gatekeeper 1 and Gatekeeper 2

In this example, Gatekeeper gk1 is configured using a HSRP virtual address as the gatekeeper's RAS address.:

```
36xx# config term
36xx(config)# g
36xx(config-gk)# zone local gk-sj cisco.com 172.21.127.55
36xx(config-gk)# no shut
```

In this example, Gatekeeper gk2 uses a HSRP virtual address as the gatekeeper's RAS address and uses the same gkname and address as on gk1.

```
36xx# config t
36xx(config-gk)# gatekeeper
36xx(config-gk)# zone local gk-sj cisco.com 172.21.127.55
36xx(config-gk)# no shut
```

---

**Note** The bring-up command **no shut** is issued on both gatekeepers, primary and secondary.

---



#### Tips

If you issue a **show gatekeeper** status command on the two gatekeepers, you will see on gk1:

```
Gatekeeper State: UP
```

However, if this command is issued on gk2, you will see:

```
Gatekeeper State: HSRP STANDBY
```

## Sample Gatekeeper Configuration

This sample gatekeeper configuration uses the E.164 address routing configuration and is based on the following assumptions:

- The gatekeeper domain names used are San Jose and New York

The command syntax for each step uses these domain names and therefore are given for descriptive purposes only. Be sure to determine the local and remote gatekeeper domain names, and the appropriate IP addresses, prior to starting your own configuration.

- The domain name for both gatekeepers on this example is “cisco.com,” although one handles the area code for San Jose and the other handles the area code for New York.
- The gatekeeper in this example is either a Cisco 3620 or Cisco 3640.

The command syntax for each step uses these domain names and therefore are given for descriptive purposes only. Be sure to determine the local and remote gatekeeper domain names, and the appropriate IP addresses, before starting your own configuration.

It is recommended that the gatekeeper ID should be in the form of *gkname.domainname*. This is to avoid ambiguity when a gatekeeper communicates with other gatekeepers in another domain.

---

**Note** This is sample configuration and is only intended as an example of how to use the **zone-prefix** and **gw-type-prefix** commands when configuring gatekeepers. It is not an example of a complete configuration of the gatekeeper.

---

### On the gatekeeper for San Jose:

The gatekeeper for San Jose is enabled by entering the following commands:

```
36xx# config term
36xx(config)# gatekeeper
36xx(config)# zone local gk-sj cisco.com
36xx(config)# zone remote gk-ny cisco.com 172.21.127.27
36xx(config)# zone access gk-sj default direct
36xx(config)# zone prefix gk-sj 1408.....
36xx(config)# zone prefix gk-ny 1212.....
36xx(config)# gw-type-prefix 3# hopoff gk-sj
36xx(config)# gw-type-prefix 4# default-technology
```

### On the gatekeeper for New York:

```
36xx(config)# config term
36xx(config)# gatekeeper
36xx(config)# zone local gk-ny cisco.com
36xx(config)# zone remote gk-sj cisco.com 172.21.1.48
36xx(config)# zone access gk-ny default direct
36xx(config)# zone prefix gk-ny 1212.....
36xx(config)# zone prefix gk-sj1408.....
36xx(config)# gw-type-prefix 3# hopoff gk-ny
36xx(config)# gw-type-prefix 4# default-technology
```

Continuing with this example, in San Jose suppose we have gateways registering with gk-sj as follows:

```
gw-sj2 configured to register with tech prefix 2#
gw-sj3 configured to register with tech prefix 3#
gw-sj4 configured to register with tech prefix 4#
```

Similarly, in New York, gateways are configured to register with gk-ny as follows:

```
gw-ny2 configured to register with tech prefix 2#  
gw-ny3 configured to register with tech prefix 3#  
gw-ny4 configured to register with tech prefix 4#
```

Given the above configuration, in San Jose, a call is presented to the gatekeeper (gk-sj) with the following target address:

### Example 1—Target address 2#12125551212

gk-sj recognizes that 2# is a *tech prefix* (it was not configured as such, but because gw-sj2 registered with it, the gatekeeper now treats 2# as a *tech prefix*) strips that and is left with “12125551212.” This is matched against the *zone prefixes*, and is a match for 1212....., so gk-sj knows that gk-ny handles this. It forwards the whole address “2#12125551212” over to gk-ny, who also looks at the *tech prefix* 2#, and routes this to gw-ny2.

### Example 2—Target address 12125551212

gk-sj checks this against known *tech prefixes*, no match. Checks against zone prefixes, matches on 1212..... for gk-ny, so routes this to gk-ny. gk-ny does not have any local registrations for this address, and there is no *tech prefix* on the address, but his default prefix is 4# and gw-ny4 is registered with 4#, so the call gets routed to gw-ny4.

### Example 3—Target address 3#12125551212

Because this contains the *tech prefix* of 3# and that is defined as a local-hopoff prefix, gk-sj just routes this to gw-sj3, despite the fact that it contains a zone-prefix for New York.

### Example 4—Target address 16505551212

gk-sj looks for a technology prefix match, and fails. Looks for a zone-prefix match and fails again. But succeeds in finding a default gateway prefix of 4#. And succeeds when gw-sj4 is registered with 4#. So, the call gets sent routed out on gw-sj4.

## Command Reference

This section documents new or modified commands. All other commands used with this feature are documented in the Cisco IOS Release 11.3 command references.

- **application**
- **aaa accounting connection h323**
- **application**
- **arq reject-unknown-prefix**
- **audio-prompt load**
- **gateway**
- **gw-accounting**
- **gw-type-prefix**
- **h323-gateway voip h323-id**
- **h323-gateway voip id**
- **h323-gateway voip interface**
- **h323-gateway voip tech-prefix**
- **lrq reject-unknown-prefix**
- **preference**
- **session target**
- **show call application voice**
- **show gatekeeper gw-type-prefix**
- **show gatekeeper status**
- **show gatekeeper zone prefix**
- **show gateway**
- **tech-prefix**
- **zone local**
- **zone prefix**
- **zone remote**

## aaa accounting connection h323

To define the accounting method list H.323 with RADIUS as a method with either **stop-only** or **start-stop** accounting options, use the **aaa accounting connection h323** command. Use the **no** form of this command to disable the use of this accounting method list.

```
aaa accounting connection h323 {stop-only | start-stop} radius
```

```
no aaa accounting connection h323 {stop-only | start-stop} radius
```

### Syntax Description

**stop-only | start-stop** Start-only or stop-only accounting options.

**radius** RADIUS is used as the method.

### Default

No accounting method list.

### Example

```
aaa accounting connection h323 {none | start-stop | stop-only | wait-start} radius
```

### Command Mode

Global configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

The method list has to be called “h323” and is activated for all voice interfaces.

This command line tells the system to create a method list called H.323, which has start-stop RADIUS as its method. The h323 method list is static and is applied by default to all voice interfaces if the **gw-accounting h323** command is also activated.

## aaa authentication login h323 radius

To define a method list called H.323 where RADIUS is a method, use the **aaa authentication login h323 radius** command. Use the **no** form of this command to restore the default.

**authentication login h323 radius**

**no authentication login h323 radius**

### Syntax Description

This command has no keywords or arguments.

### Default

No method list.

### Command Mode

Global configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

This command line registers the H.323 method list (also referred to as an h323 service) which has RADIUS as its only method in the router. The VoIP calls send all their authentication requests through the H.323 service. If this line does not exist in the router configuration VoIP authentication will not take place.

A significant difference in the usage of this command is that with Cisco IOS release 11.3(T) the name of the method list is flexible and can be changed by the user. However, when using method list for configuring AAA with the VoIP service provider software, the method list is a specific name that you should *not* change.

A list is defined using the **aaa authentication login h323 radius** command, and is then applied to an interface. Do not apply this list to any interface for voice authentication. When enabled, using this command applies to all voice interfaces. The function of this command is activated through the IVR application.

## application

To select the session application for Interactive Voice Response, use the **application** command. Use this command when you configure the dial peers.

**application** *name*

### Syntax Description

<i>name</i>	Indicates the name of the IVR script application the call should be handed to.
-------------	--

### Default

None. The call will be handed to the predefined session application.

### Command Mode

Dial peer configuration mode

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### Example

The following example shows how the IVR script named `clid_authen_collect` processes a call.

```
application clid_authen_collect, CallID 90 got event IVR_EV-CALL_SETUP_IND
:
: ivr action: IVR_ACT_CALL_SETUP_ACK
:
: ivr action: IVR_ACT_CALL_PROCEEDING
:
: ivr action: IVR_ACT_CALL_CONNECT

: ivr action: IVR_ACT_CALL_PROCEEDING
:
: ivr action: IVR_ACT_CALL_CONNECT
```

## arq reject-unknown-prefix

To control the behavior of the gatekeeper when it receives an Admission Request (ARQ) that does not match any configured zone prefixes, use the **arq reject-unknown-prefix** command. Use this command to force the gatekeeper to reject such requests. If however, the desired behavior is for the gatekeeper to try to service such requests, then use the **no** form of this command.

```
arq reject-unknown-prefix  
no arq reject-unknown-prefix
```

### Syntax Description

This command has no arguments or keywords.

### Default

No ARQ request rejection.

### Command Mode

Gatekeeper configuration

### Usage Guidelines

This command first appeared in the Cisco IOS Release 11.3(6)NA.

You can use the **arq reject-unknown-prefix** command to control the behavior of the gatekeeper when it receives an Admission Request (ARQ) for a destination E.164 address that does not match any configured zone prefixes.

When an endpoint or gateway initiates an H.323 call, it sends an ARQ to its gatekeeper. The gatekeeper uses the configured list of zone prefixes to determine to which zone the call should be directed. If the called address does not match any known zone prefixes, the gatekeeper will attempt to hairpin the call out through a local gateway with a matching technology prefix. If this is not the desired behavior, then use the **arq reject-unknown-prefix** command to mandate that such calls should be rejected.

This command is typically used either to restrict local gateway calls to a known set of prefixes, or to deliberately fail such calls so that an alternate choice on a gateway's rotary dial-peer can be selected.

### Example

The following example shows how this command affects the behavior of a gatekeeper. Consider a gatekeeper configured as follows:

```
zone local gk408 cisco.com  
zone remote gk415 cisco.com 172.21.139.91  
zone prefix gk408 1408.....  
zone prefix gk415 1415.....
```

In the above example, the gatekeeper manages a zone containing gateways to the 408 area code, and it knows about a peer gatekeeper with gateways to the 415 area code. These zones are configured with the appropriate prefixes so that calls to those area codes hop off in the optimal zone.

If an endpoint makes a call to the 408 area code, the call will be routed out through a local gateway. If the call is to the 415 area code, it will be directed to the gk415 zone and hop off on a gateway there. But if a call is made to, say, the 212 area code, it will also be directed to a local gateway in the gk408 zone.

As a result of the command:

```
arq reject-unknown-prefix
```

a call made to the 212 area code is rejected because the destination address does not match any configured prefixes.

## audio-prompt load

To refresh the audio file in memory, use the **audio-prompt load** command. The router will only load the .au (audio) file when the script initially plays that prompt, or on the router restart. If the .au file is changed, the user must run this EXEC command to reread the file. This will generate an error message if the file is not accessible, or if there is a format error.

**audio-prompt load** *name*

### Syntax Description

<i>name</i>	Indicates the location and name of the .au file to load. It can be loaded from memory, Flash memory, or an FTP server. Presently, with Cisco IOS Release 11.3(6)NA2, the <URL> pointer refers to the directory where Flash memory is stored.
-------------	--

### Default

No default behavior or values.

### Command Mode

Privileged EXEC

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

- The first time the IVR application plays a prompt, it reads it from the URL (or the specified location for the .au file, such as flash or FTP server name) into RAM. Then it plays it from RAM.
- The router is up and running, using the clid\_authen\_collect script. The script is playing out the prompt from flash:enter\_destination.au.
- A sequence of events would be:
  - When the first caller is asked to enter their account and PIN numbers, the enter\_account.au and enter\_pin.au files will be loaded into RAM from Flash memory.
  - When the next call comes in, these prompts are played from the RAM copy.
  - If all callers enter valid account and PIN numbers, then the auth\_failed.au file will not be loaded from Flash memory into RAM memory.

### Example

This example loads (from Flash memory) the audio file flash:enter\_pin.au

```
audio-prompt load flash:enter_pin.au
```

## gateway

To enable the H.323 VoIP gateway, use the **gateway** global configuration command. Use the **no** form of this command to unregister this gateway with the gatekeeper.

**gateway**  
**no gateway**

### Syntax Description

This command has no keywords or arguments.

### Default

The gateway is unregistered.

### Command Mode

Global configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

Use the gateway command to enable H.323 VoIP gateway functionality. After you enable the gateway, it will attempt to discover a gatekeeper by using the H.323 RAS GRQ message.

- If the gateway is enabled, the gateway will attempt to discover a gatekeeper by using the H.323 RAS GRQ message.
- The VoIP gateway is stopped by entering **no gateway**.
- If you enter **no gateway voip**, the VoIP gateway will unregister with the gatekeeper via the H.323 RAS URQ message.

## gw-accounting

To enable gateway specific accounting, use the **gw-accounting** command. Use the no form of this command to disable gateway specific accounting.

**gw-accounting** [**h323** | **syslog**]

**no gw-accounting** [**h323** | **syslog**]

### Syntax Description

<b>h323</b>	(Optional) H.323 method uses RADIUS to output accounting CDRs.
<b>syslog</b>	(Optional) Syslog uses the system logging facility to output CDRs.

### Default

Disable gateway specific accounting.

### Command Mode

Global configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

- This command is used when configuring the AAA accounting application.
- This command line defines a method for doing the accounting and enables the gateway to do the accounting. There are two accounting methods defined.
- Both **h.323** and **syslog** can be enabled at the same time, which causes CDRs to be generated in both methods.

## gw-type-prefix

To configure a technology prefix in the gatekeeper, use the **gw-type-prefix** command. To remove the technology prefix, use the **no** form of the command.

```
gw-type-prefix type-prefix [hopoff gkid] [default-technology]
[[gw ipaddr ipaddr [port]]]
```

```
no gw-type-prefix type-prefix [hopoff gkid] [default-technology]
[[gw ipaddr ipaddr [port]]]
```

### Syntax Description

<i>type-prefix</i>	A technology prefix is recognized and is stripped before checking for the zone prefix. It is strongly recommended that you select technology prefixes that do not lead to ambiguity with zone prefixes. Do this by using the # character to terminate technology prefixes, for example, 3#.
<b>hopoff</b> <i>gkid</i>	(Optional) Specifies the gatekeeper or zone where the call is to hop off, regardless of the zone prefix in the destination address. The <i>gkid</i> argument refers to a zone previously configured using the zone local or zone remote comment.
<b>default-technology</b>	(Optional) Gateways registering with this prefix option are used as the default for routing any addresses that are otherwise unresolved.
<b>gw ipaddr</b> <i>ipaddr</i> [ <i>port</i> ]	(Optional) Indicates that the gateway is incapable of registering technology prefixes. When it registers, it adds the gateway to the group for this type-prefix, just as if it had sent the technology prefix in its registration. This parameter can be repeated to associate more than one gateway with a technology prefix.

### Default

No technology prefix is defined.

### Command Mode

Gatekeeper configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

- More than one gateway can register with the same technology prefix. In such cases, a random selection is made of one of them.
- You do not have to define a technology prefix to a gatekeeper if there are gateways configured to register with that prefix, and if there are no special flags (**hopoff gkid** or **default-technology**) that you want to associate with that prefix.
- You need to configure the gateway type prefix of all remote technology prefixes that will be routed through this gatekeeper.

## Example

The following example specifies 4# as the default technology prefix:

```
gw-type-prefix 4# default-technology
```

## Related Command

**zone prefix**

## h323-gateway voip h323-id

To configure the H.323 name of the gateway identifying this gateway to its associated gatekeeper, use the **h323-gateway voip h323-id** interface configuration command. Use the **no** form of this command to disable this defined gateway name.

```
h323-gateway voip h323-id interface-id  
no h323-gateway voip h323-id interface-id
```

### Syntax Description

<i>interface-id</i>	H.323 name (ID) used by this gateway when this gateway communicates with its associated gatekeeper. Usually, this ID is the name of the gateway with the gatekeeper's domain name appended to the end: name@domain-name.
---------------------	--

### Default

No gateway identification is defined.

### Command Mode

Interface configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### Example

The following example configures Ethernet interface 0/0 as the gateway interface. In this example, the gateway ID is GW13@cisco.com.

```
interface Ethernet0/0  
 ip address 172.9.53.13 255.255.255.0  
 h323-gateway voip interface  
 h323-gateway voip id GK15.cisco.com ipaddr 172.9.53.15 1719  
 h323-gateway voip h323-id GW13@cisco.com  
 h323-gateway voip tech-prefix 13#
```

### Related Commands

```
h323-gateway voip id  
h323-gateway voip interface  
h323-gateway voip tech-prefix
```

## h323-gateway voip id

To define the name and location of the gatekeeper for this gateway, use the **h323-gateway voip id** interface configuration command. Use the **no** form of this command to disable this gatekeeper identification.

```
h323-gateway voip id gatekeeper-id {ipaddr ip-address [port-number] | multicast}
no h323-gateway voip id gatekeeper-id {ipaddr ip-address [port-number] | multicast}
```

### Syntax Description

<i>gatekeeper-id</i>	Indicates the H.323 identification of the gatekeeper. This value must exactly match the gatekeeper ID in the gatekeeper configuration. The recommended format is <i>name.doman-name</i> .
<b>ipaddr</b>	Indicates that the gateway will use an IP address to locate the gatekeeper
<i>ip-address</i>	Defines the IP address used to identify the gatekeeper.
<b>multicast</b>	Indicates that the gateway will use multicast to locate the gatekeeper.
<i>port-number</i>	(Optional) Defines the port number used.

### Default

No gatekeeper identification is defined.

### Command Mode

Interface configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

This command tells the H.323 gateway associated with this interface which H.323 gatekeeper to talk to and where to locate it. The gatekeeper ID configured here must exactly match the gatekeeper ID in the gatekeeper configuration.

### Example

The following example configures Ethernet interface 0/0 as the gateway interface. In this example, the gatekeeper ID is GK15.cisco.com and its IP address is 172.9.53.15 (using port 1719).

```
interface Ethernet0/0
 ip address 172.9.53.13 255.255.255.0
 h323-gateway voip interface
 h323-gateway voip id GK15.cisco.com ipaddr 172.9.53.15 1719
 h323-gateway voip h323-id GW13@cisco.com
 h323-gateway voip tech-prefix 13#
```

Related Commands

**h323-gateway voip h323-id**  
**h323-gateway voip interface**  
**h323-gateway voip tech-prefix**

## h323-gateway voip interface

To configure this interface as an H.323 interface, use the **h323-gateway voip interface** interface configuration command. Use the **no** form of this command to disable H.323 functionality for this interface.

**h323-gateway voip interface**  
**no h323-gateway voip interface**

### Syntax Description

This command has no arguments or keywords.

### Default

Disabled.

### Command Mode

Interface configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### Example

The following example configures Ethernet interface 0.0 as the gateway interface. In this example, the **h323-gateway voip interface** command configures this interface as an H.323 interface.

```
interface Ethernet0/0
 ip address 172.9.53.15 255.255.255.0
 h323-gateway voip interface
 h323-gateway voip id GK15.cisco.com ipaddr 172.9.53.15 1719
 h323-gateway voip h323-id GW15@cisco.com
 h323-gateway voip tech-prefix 13#
```

### Related Commands

**h323-gateway voip h323-id**  
**h323-gateway voip id**  
**h323-gateway voip tech-prefix**

## h323-gateway voip tech-prefix

To define the technology prefix that the gateway will register with the gatekeeper, use the **h323-gateway voip tech-prefix** interface configuration command. Use the **no** form of this command to disable this defined technology prefix.

```
h323-gateway voip tech-prefix prefix  
no h323-gateway voip tech-prefix prefix
```

### Syntax Description

<i>prefix</i>	Defines the numbers used as the technology prefixes. Each technology prefix can contain up to 11 characters. Although not strictly necessary, a pound (#) symbol is frequently used as the last digit in a technology prefix. Valid characters are 0 through 9, the pound (#) symbol, and the asterisk (*).
---------------	---

### Default

Disabled.

### Command Mode

Interface configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

This command defines a technology prefix that the gateway will then register with the gatekeeper. Technology prefixes can be used as a discriminator so that the gateway can tell the gatekeeper that a certain technology is associated with a particular call (for example, 15# could mean a fax transmission), or it can be used like an area code for more generic routing. No standard currently defines what the numbers in a technology prefix mean. By convention, technology prefixes are designated by a pound (#) symbol as the last character.

---

**Note** Cisco gatekeepers use the asterisk (\*) as a reserved character. If you are using Cisco gatekeepers, do not use the asterisk as part of the technology prefix.

---

### Example

The following example configures Ethernet interface 0.0 as the gateway interface. In this example, the technology prefix is defined as 13#.

```
interface Ethernet0/0  
ip address 172.9.53.15 255.255.255.0  
h323-gateway voip interface  
h323-gateway voip id GK15.cisco.com ipaddr 172.9.53.15 1719  
h323-gateway voip h323-id GW15@cisco.com  
h323-gateway voip tech-prefix 13#
```

Related Commands

**h323-gateway voip id**  
**h323-gateway voip interface**  
**h323-gateway voip h323-id**

## lrq reject-unknown-prefix

To control the behavior of the gatekeeper when it receives a Location Request (LRQ) that does not match any configured zone prefixes, use the **lrq reject-unknown-prefix** command. Use this command to force the gatekeeper to reject such requests. If however, the desired behavior is for the gatekeeper to try to service such requests, then use the **no** form of this command.

**lrq reject-unknown-prefix**  
**no lrq reject-unknown-prefix**

### Syntax Description

This command has no arguments or keywords.

### Default

Service requests.

### Command Mode

Gatekeeper configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA.

You can use the **lrq reject-unknown-prefix** command to control the behavior of the gatekeeper when it receives a Location Request (LRQ) that does not match any configured zone prefixes.

When the gatekeeper receives a Location Request asking about an E.164 address, it matches the target address against the list of configured zone prefixes. If the address matches a zone prefix, the behavior is unambiguous and well-defined:

- If the matching zone is local (that is, one controlled by this gatekeeper), the LRQ is serviced
- If the matching zone is remote (that is, one controlled by some other gatekeeper), the LRQ is rejected

However, if the target address does not match any known local or remote zone prefixes, then the default behavior is to attempt to service the call using one of the local zones. This default behavior may not be suitable for all sites, so the **lrq reject-unknown-prefix** command allows you to force the gatekeeper to reject such requests.

### Example

The following example shows how this command affects the behavior of a gatekeeper. The gatekeeper manages two zones, one with gateways with interfaces in the 408 area code, and one with gateways in the 415 area code. These zones are configured with the appropriate prefixes so that calls to those area codes hop off in the optimal zone. If some other zone had been erroneously configured to route calls to the 212 area code to this gatekeeper. When the Location Request arrives, this gatekeeper fails to match the area code, and so the LRQ is rejected.

```
zone local gk408 cisco.com
zone local gk415 cisco.com
zone prefix gk408 1408.....
zone prefix gk415 1415.....
lrq reject-unknown-prefix
```

## preference

To indicate the preference order for matching dial peers in a rotary group, use the **preference** command. It is useful in selecting the desired dial peer when multiple dial peers are matched for a dial string. Use the **no** form of this command if you do not want to assign a preference.

**preference** *value*  
**no preference** *value*

### Syntax Description

*value* Defines an integer value from 0 to 10.

### Default

No preference order is given.

### Command Mode

Dial peer configuration mode

### Usage Guidelines

This command was first appeared in Cisco IOS Release 11.3(7)NA2.

- Use this command with the Rotary Calling Pattern feature.
- The hunting algorithm precedence is configurable. For example, if you wish a call processing sequence to go to destination #A first, then destination B second, and third to destination C; you would assign *preference* (0 being the highest priority) to the destinations in the following order:
  - Preference 0 to A
  - Preference 1 to B
  - Preference 2 to C

### Examples

The following examples show different dial peer configurations using the **preference** command.

#### Example 1

Dialpeer	destpat	preference	session-target
1	4085551048	0 (highest)	jmmurphy-voip
2	408555	0	sj-voip
3	408555	1 (lower)	backup-sj-voip
4	.....	1	0:D (interface)
5	.....	0	anywhere-voip

If the destination number is 4085551048, the order of attempts will be 1,2,3,5,4.

Example 2

Dialpeer	destpat	preference
1	408555	0
2	4085551048	1
3	4085551	0
4	.....4085551.....	0

The number dialed is 4085551048, the order will be 2, 3, 4, 1.

---

**Note** The default behavior is that the longest matching dial peer supersedes the preference value.

---

## session target

To identify the IP address of the destination gatekeeper, use the **session target** command. The field indicating if the RAS protocol is being used has been added. Enter the **no** form of this command to restore the default condition.

```
session target {ipv4:destination-address | dns:[$$$. | $d$. | $e$. | $u$.] host-name |
loopback:rtp | loopback:compressed | loopback:uncompressed | ras}
no session target
```

### Syntax Description

<b>ipv4:destination-address</b>	IP address of the dial peer.
<b>dns:[\$\$\$...] host-name</b>	Indicates that the domain name server will be used to resolve the name of the IP address. Valid entries for this parameter are characters representing the name of the host device.  (Optional) Use one of the following three wildcards with this keyword when defining the session target for VoIP peers: <ul style="list-style-type: none"> <li>• <b>\$\$.</b>—Indicates that the source destination pattern will be used as part of the domain name.</li> <li>• <b>\$d.</b>—Indicates that the destination number will be used as part of the domain name.</li> <li>• <b>\$e.</b>—Indicates that the digits in the called number will be reversed, periods will be added between each digit of the called number, and that this string will be used as part of the domain name.</li> <li>• <b>\$u.</b>—Indicates that the unmatched portion of the destination pattern (such as a defined extension number) will be used as part of the domain name.</li> </ul>
<b>loopback:rtp</b>	Indicates that all voice data will be looped back to the originating source. This is applicable for VoIP peers.
<b>loopback:compressed</b>	Indicates that all voice data will be looped back in compressed mode to the originating source. This is applicable for POTS peers.
<b>loopback:uncompressed</b>	Indicates that all voice data will be looped-back in uncompressed mode to the originating source. This is applicable for POTS peers.
<b>ras</b>	Indicates that the RAS signaling function protocol is being used—meaning that a gatekeeper will be consulted to translate the E.164 address to an IP address.

### Default

No IP address or domain name defined.

### Command Mode

Dial peer configuration

## Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(1)T.

Use the **session target** command to specify a network-specific address or domain name for a dial peer. Whether you select a network-specific address or a domain name depends on the session protocol you select.

The **session target loopback** command is used for testing the voice transmission path of a call. The loopback point will depend on the call origination and the loopback type selected.

The **session target dns** command can be used with or without the specified wildcards. Using the optional wildcards can reduce the number of VoIP dial peer session targets you need to configure if you have groups of numbers associated with a particular router.

Use the **session target ras** command to specify that the RAS protocol is being used to determine the IP address of the session target.

## Examples

The following example configures a session target using DNS for a host, “voice\_router,” in the domain “cisco.com”:

```
dial-peer voice 10 voip
  session target dns:voice_router.cisco.com
```

The following example configures a session target using DNS, with the optional **\$u\$**. wildcard. In this example, the destination pattern has been configured to allow for any four-digit extension, beginning with the numbers 1310222. The optional wildcard **\$u\$**. indicates that the router will use the unmatched portion of the dialed number—in this case, the four-digit extension, to identify the dial peer. As in the previous example, the domain is “cisco.com.”

```
dial-peer voice 10 voip
  destination-pattern 1310222...
  session target dns:$u$.cisco.com
```

The following example configures a session target using dns, with the optional **\$d\$**. wildcard. In this example, the destination pattern has been configured for 13102221111. The optional wildcard **\$d\$**. indicates that the router will use the destination pattern to identify the dial peer in the “cisco.com” domain.

```
dial-peer voice 10 voip
  destination-pattern 13102221111
  session target dns:$d$.cisco.com
```

The following example configures a session target using DNS, with the optional **\$e\$**. wildcard. In this example, the destination pattern has been configured for 12345. The optional wildcard **\$e\$**. indicates that the router will reverse the digits in the destination pattern, add periods between the digits, and then use this reverse-exploded destination pattern to identify the dial peer in the “cisco.com” domain.

```
dial-peer voice 10 voip
  destination-pattern 12345
  session target dns:$e$.cisco.com
```

The following example configures a session target using RAS:

```
dial-peer voice 11 voip
  destination-pattern 13102221111
  session target ras
```

Related Commands

**destination-pattern**  
**session protocol**

## show call application voice

To define the names of the audio files the IVR script will play, the operation of the abort keys, what prompts are used, and caller interaction, the **show call application voice** command.

```
show call application voice [name | summary]  
no show call application voice [name | summary]
```

### Syntax Description

<i>name</i>	(Optional) The name of the desired IVR application.
<b>summary</b>	(Optional) Enter this field to display a one line summary. If the command is entered without summary, a complete detailed description is displayed of the application.

### Command Mode

Privileged EXEC

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

- If the name of a specific application is entered, it will give information about that application.
- If the **summary** keyword is entered a one line summary will be displayed about each application.
- If the command is entered without the **summary**, a detailed description of the entered ivr application is displayed.

### Sample Display

This example shows the output for the `clid_authen_collect` IVR script:

```
sblab115>show call application voice clid_authen_collect  
Application clid_authen_collect has 10 states with 0 calls active  
State start has 1 actions and 5 events  
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=dnis  
  If Event IVR_EV_DEFAULT goto state end  
  If Event IVR_EV_CALL_DIGIT do nothing  
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK  
    and goto state start  
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest  
  If Event IVR_EV_AAA_FAIL goto state get_account  
State end has 1 actions and 3 events  
  Do Action IVR_ACT_END.  
  If Event IVR_EV_DEFAULT goto state end  
  If Event IVR_EV_CALL_DIGIT do nothing  
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY  
    and do nothing
```

```

State get_account has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_account.au
    allowInt=1, pContent=0x60E4C564
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_pin has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_pin.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern pin is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=pin
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_TIMEOUT goto state get_pin count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_pin
State authenticate has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=pin
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_TIMEOUT do nothing count=0
  If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State collect_dest has 4 actions and 8 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_destination.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_DIALPLAN.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state collect_dest
  If Event IVR_EV_TIMEOUT goto state collect_dest count=0
  If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
  If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
  If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
  Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling= account=
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_UP goto state active
  If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing

```

```

State authenticate_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
  Do Action IVR_ACT_PLAY_FAILURE_TONE.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing

sblabl15>show call application voice clid_authen_collect
Application clid_authen_collect has 10 states with 0 calls active
State start has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=dnis
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
    and goto state start
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_AAA_FAIL goto state get_account
State end has 1 actions and 3 events
  Do Action IVR_ACT_END.
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
    and do nothing
State get_account has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_account.au
    allowInt=1, pContent=0x60E4C564
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
    patName=account
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_TIMEOUT goto state get_account count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_account
State get_pin has 4 actions and 7 events
  Do Action IVR_ACT_PLAY.
    URL: flash:enter_pin.au
    allowInt=1, pContent=0x0
  Do Action IVR_ACT_ABORT_KEY. abortKey=*
  Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
  Do Action IVR_ACT_COLLECT_PATTERN. Pattern pin is .+
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_PAT_COL_SUCCESS goto state authenticate
    patName=pin
  If Event IVR_EV_PLAY_COMPLETE do nothing
  If Event IVR_EV_ABORT goto state get_account
  If Event IVR_EV_TIMEOUT goto state get_pin count=0
  If Event IVR_EV_PAT_COL_FAIL goto state get_pin
State authenticate has 1 actions and 5 events
  Do Action IVR_ACT_AUTHENTICATE. accountName=account, pinName=pin
  If Event IVR_EV_DEFAULT goto state end
  If Event IVR_EV_CALL_DIGIT do nothing
  If Event IVR_EV_AAA_SUCCESS goto state collect_dest
  If Event IVR_EV_TIMEOUT do nothing count=0
  If Event IVR_EV_AAA_FAIL goto state authenticate_fail
State collect_dest has 4 actions and 8 events

```

```
Do Action IVR_ACT_PLAY.
    URL: flash:enter_destination.au
    allowInt=1, pContent=0x0
Do Action IVR_ACT_ABORT_KEY. abortKey=*
Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
Do Action IVR_ACT_COLLECT_DIALPLAN.
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_PLAY_COMPLETE do nothing
If Event IVR_EV_ABORT goto state collect_dest
If Event IVR_EV_TIMEOUT goto state collect_dest count=0
If Event IVR_EV_DIAL_COL_SUCCESS goto state place_call
If Event IVR_EV_DIAL_COL_FAIL goto state collect_dest
If Event IVR_EV_TIMEOUT goto state collect_dest count=0
State place_call has 1 actions and 4 events
Do Action IVR_ACT_PLACE_CALL.
    destination= called=
    calling=         account=
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
If Event IVR_EV_CALL_UP goto state active
If Event IVR_EV_CALL_FAIL goto state place_fail
State active has 0 actions and 2 events
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
State authenticate_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY.
    URL: flash:auth_failed.au
    allowInt=0, pContent=0x0
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
State place_fail has 1 actions and 2 events
Do Action IVR_ACT_PLAY_FAILURE_TONE.
If Event IVR_EV_DEFAULT goto state end
If Event IVR_EV_CALL_DIGIT do nothing
```

## show gatekeeper gw-type-prefix

To display the gateway-type prefix table, use the **show gatekeeper gw-type-prefix EXEC** command.

**show gatekeeper gw-type-prefix**

### Syntax Description

This command has no arguments or keywords.

### Command Mode

Privileged EXEC

### Usage Guidelines

This command first appeared in the Cisco IOS Release 11.3 NA.

### Sample Display

The following is sample display from the **show gatekeeper gw-type-prefix** command:

```
router# show gatekeeper gw-type-prefix

(GATEWAYS-TYPE PREFIX TABLE
=====
Prefix: 3#*      (Hopoff- gk408)

Prefix: 4#*      (Default gateway-technology)
Static Configured Gateways:

Prefix: 7#*      (Hopoff gk408)
Static Configured Gateways:
    1.1.1.1:1720
    2.2.2.2:1720
```

Table 1 describes the fields shown in the **show gatekeeper gw-type-prefix** display.

**Table 1** show gatekeeper gw-type-prefix Command Field Descriptions

Field	Description
Prefix:	The tech-prefix defined with the <b>gw-type-prefix</b> command.
(Hopoff gk408)	Calls specifying tech-prefix 3# or 7# will always be routed to zone gk408, regardless of the actual zone prefix in the destination address.
(Default gateway-technology)	The address associated with the technology prefix is a gateway used as the default for routing any addresses that are otherwise unresolveable.
Static Configured Gateways:	Lists all IP addresses and port numbers of statically configured gateways.

## show gatekeeper status

To show overall gatekeeper status that includes authorization and authentication status, zone status, and so on, use the **show gatekeeper status** EXEC command.

```
show gatekeeper status
```

### Syntax Description

This command has no arguments or keywords.

### Command Mode

Privileged EXEC

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3 NA.

### Sample Display

The following is sample display from the **show gatekeeper status** command:

```
router# show gatekeeper status

Gatekeeper State: UP
Zone Name: gk-px4.cisco.com
Accounting: DISABLED
Security: DISABLED
```

Table 2 describes the fields shown in the **show gatekeeper status** display.

**Table 2 show gatekeeper status Command Field Descriptions**

Field	Description
Gatekeeper State	<ul style="list-style-type: none"> <li>UP is operational.</li> <li>DOWN is administratively shut down.</li> <li>INACTIVE is administratively enabled, that is, the no shutdown command has been issued but no local zones have been configured.</li> <li>HSRP STANDBY indicates the gatekeeper is on hot standby and will take over if the currently active gatekeeper fails.</li> </ul>
Zone Name	Zone name.
Accounting	Authorization and accounting status.
Security	Security status.

## show gatekeeper zone prefix

To display the **zone prefix** table, use the **show gatekeeper zone prefix EXEC** command.

**show gatekeeper zone prefix**

### Syntax Description

This command has no arguments or keywords.

### Command Mode

Privileged EXEC

### Usage Guidelines

This command first appeared in the Cisco IOS Release 11.3 NA.

### Sample Display

The following is sample display from the **show gatekeeper zone prefix** command:

```
5300# show gatekeeper zone prefix

      ZONE PREFIX TABLE
      =====
GK-NAME          E164-PREFIX
-----          -
gk.zone13       212.....
gk.zone14       415.....
gk.zone14       408.....
```

Table 3 describes the fields shown in the **show gatekeeper status** display.

**Table 3 show gatekeeper status command Field Descriptions**

Field	Description
GK-NAME	The gatekeeper name.
E164-PREFIX	The E.164 prefix and a dot that acts as a wildcard for matching each remaining number in the telephone number.

## show gateway

To display the current gateway status, use the **show gateway** command.

```
show gateway
```

### Syntax Description

This command has no keywords or arguments.

### Command Mode

Privileged EXEC

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### Sample Display

```
gte-as5300-2# show gateway  
Gateway voip2@vml1lab is registered to Gatekeeper gk1.vml1lab
```

## tech-prefix

To specify a particular technology prefix be prepended to the destination pattern of a specific dial peer, use the **tech-prefix** dial peer configuration command. Use the **no** form of this command to disable the defined technology prefix for this dial peer.

**tech-prefix** *number*  
**no tech-prefix** *number*

### Syntax Description

<i>number</i>	Defines the numbers used as the technology prefix. Each technology prefix can contain up to 11 characters. Although not strictly necessary, a pound (#) symbol is frequently used as the last digit in a technology prefix. Valid characters are 0 through 9, the pound (#) symbol, and the asterisk (*).
---------------	---

### Default

No technology prefix is defined.

### Command Mode

Dial peer configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

Technology prefixes are used to distinguish between gateways having specific capabilities within a given zone. In the exchange between the gateway and the gatekeeper, the technology prefix is used to select a gateway after the zone has been selected. Use the **tech-prefix** command to define technology prefixes.

Technology prefixes can be used as a discriminator so that the gateway can tell the gatekeeper that a certain technology is associated with a particular call (for example, 15# could mean a fax transmission), or it can be used like an area code for more generic routing. No standard defines what the numbers in a technology prefix mean; by convention, technology prefixes are designated by a pound (#) symbol as the last character.

In most cases, there is a dynamic protocol exchange between the gateway and the gatekeeper that enables the gateway to inform the gatekeeper about technology prefixes and where to forward calls. If, for some reason, that dynamic registry feature is not in effect, you can statically configure the gatekeeper to query the gateway for this information by configuring the **gw-type-prefix** command on the gatekeeper. Use the **show gatekeeper gw-type-prefix** to display how the gatekeeper has mapped the technology prefixes to local gateways.

---

**Note** Cisco gatekeepers use the asterisk (\*) as a reserved character. If you are using Cisco gatekeepers, do not use the asterisk as part of the technology prefix.

---

## Example

The following example defines a technology prefix of 14# for the specified dial peer. In this example, the technology prefix means that the H.323 gateway will ask the RAS gatekeeper to direct calls using the technology prefix of 14#.

```
dial-peer voice 10 voip
 destination-pattern 14...
 tech-prefix 14#
```

## Related Commands

**gw-type-prefix**  
**show gatekeeper gw-type-prefix**

## zone local

To specify a zone controlled by a gatekeeper, use the **zone local** gatekeeper configuration command. To remove a zone controlled by a gatekeeper, use the **no** form of this command. This command can also be used to change the IP address used by the gatekeeper.

```
zone local gatekeeper-name domain-name [rasIPAddress]
no zone local gatekeeper-name domain-name
```

### Syntax Description

<i>gatekeeper-name</i>	The gatekeeper's name or zone name. This is usually the fully domain-qualified host name of the gatekeeper. For example, if the domain-name is cisco.com, the gatekeeper-name might be gk1.cisco.com. However, if the gatekeeper is controlling multiple zones, the gatekeeper-name for each zone should be some unique string that has a mnemonic value.
<i>domain-name</i>	The domain name served by this gatekeeper.
<i>rasIPAddress</i>	(Optional) The IP address of one of the interfaces on the gatekeeper. When the gatekeeper responds to gatekeeper discovery messages, it signals the endpoint or gateway to use this address in future communications. Setting this address for one local zone makes it the address used for all local zones.

### Default

No local zone is defined.

---

**Note** The gatekeeper cannot operate without at least one local zone definition. Without local zones, the gatekeeper goes to an inactive state when the **no shutdown** command is issued.

---

### Command Mode

Gatekeeper configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3 NA.

Multiple local zones can be defined. The gatekeeper manages all configured local zones. Intrazone and interzone behavior remains the same (zones are controlled by the same or different gatekeepers.)

Only one rasIPAddress argument can be defined for all local zones. You cannot configure each zone to use a different RAS IP address. If you define this in the first zone definition, you can omit it for all subsequent zones, which automatically pick up this address. If you set it in a subsequent **zone local** command, it also changes the RAS address of all previously configured local zones. After it is defined, you can change it by re-issuing any **zone local** command with a different rasIPAddress argument.

If the rasIPaddress argument is an HSRP virtual address, it automatically puts the gatekeeper into HSRP mode. In this mode, the gatekeeper assumes STANDBY or ACTIVE status according to whether the HSRP interface is on STANDBY or ACTIVE status.

You cannot remove a local zone if there are endpoints or gateways registered in it. To remove the local zone, shut down the gatekeeper first, which forces unregistration.

Multiple zones are controlled by multiple logical gatekeepers on the same Cisco IOS release.

### Example

The following example creates a zone controlled by a gatekeeper in the domain called cisco.com:

```
zone local gk1.cisco.com cisco.com
```

### Related Commands

```
show gatekeeper zone statue  
zone remote
```

## zone prefix

To configure the gatekeeper with knowledge of its own and any remote zone's prefixes, use the **zone prefix** gatekeeper configuration command. To remove knowledge of zone prefixes, use the **no** form of this command.

```
zone prefix gatekeeper-name e164-prefix  
no zone prefix gatekeeper-name e164-prefix
```

### Syntax Description

<i>gatekeeper-name</i>	The name of a local or remote gatekeeper, which must have been defined using the <b>zone local</b> or <b>zone remote</b> command.
<i>e164-prefix</i>	An E.164 prefix in standard form followed by dots (.) that each represent a number in the E.164 address.  For example, 212..... is matched by 212 and any seven numbers.

### Default

No knowledge of its own or any other zone's prefix is defined.

### Command Mode

Gatekeeper configuration

### Usage Guidelines

Although a dot representing each digit in an E.164 address is the preferred configuration method, you may also enter an asterisk (\*) to match any number of digits.

A gatekeeper may handle more than one zone prefix, but a zone prefix cannot be shared by more than one gatekeeper. If you have defined a zone prefix as being handled by a gatekeeper, and now define it as being handled by a second gatekeeper, the second assignment will cancel the first.

When a zone handles several prefixes, all gateways in that zone constitute a common pool which can be used to hop off to any of those prefixes. You may however wish to partition your gateways by prefix, for instance you have a gateway which interfaces to the 408 area code, and another which interfaces to the 415 area code, and for cost reasons you want each gateway only to be used for calls to its area code. In that case, you can define several local zones on the gatekeeper, each responsible for a prefix, and have each gateway register to the zone handling its prefix. For example, you can define local zone gk-408 handling prefix 408..... and local zone gk-415 handling 415..... and have the gateway interfacing to the 408 area code register with gk-408, and the gateway with the 415 interface register to gk-415.

### Example

The following example matches the 212 area code and any seven digits as the zone prefix for gk-ny:

```
zone prefix gk-ny 212.....
```

Related Commands

**zone local**  
**zone remote**

## zone remote

To specify statically a remote zone if DNS is unavailable or undesirable, use the **zone remote** gatekeeper configuration command. To remove the remote zone, use the **no** form of this command.

```
zone remote other-gatekeeper-name other-domain-name other-gatekeeper-ip-address  
[port-number]
```

```
no zone remote other-gatekeeper-name other-domain-name other-gatekeeper-ip-address  
[port-number]
```

### Syntax Description

<i>other-gatekeeper-name</i>	Name of the remote gatekeeper.
<i>other-domain-name</i>	Domain name of the remote gatekeeper.
<i>other-gate-keeper-ip-address</i>	IP address of the remote gatekeeper.
<i>port-number</i>	(Optional) RAS signaling port number for the remote zone. Value ranges from 1 to 65535. If this is not set, the default is the well known RAS port number 1719.

### Default

No remote zone is defined. DNS will locate the remote zone.

### Command Mode

Gatekeeper configuration

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3 NA.

All gatekeepers do not have to be in DNS. For those that are not, use the **zone remote** command so that the local gatekeeper knows how to access them. In addition, you may want to improve call response time slightly for frequently accessed zones. If the **zone remote** command is configured for a particular zone, you do not need to make a DNS lookup transaction.

### Example

The following example configures the local gatekeeper to reach targets of the form *xxx.cisco.com* by sending queries to the gatekeeper named *sj3.cisco.com* at IP address 1.2.3.4:

```
zone remote sj3.cisco.com cisco.com 1.2.3.4
```

### Related Commands

```
show gatekeeper zone statue  
zone local
```

## Debug Commands

This section describes new and modified debug commands. All other commands used with this feature are documented in the *Cisco IOS Release 11.3 Debug Command Reference*.

- **debug cch323 h225**
- **debug cch323 h245**
- **debug cch323 ras**
- **debug h225**
- **debug ras**
- **debug voip aaa**
- **debug voip ccapi**
- **debug voip ivr**

## debug cch323 h225

To trace of the state transition of the H.225 state machine based on the processed event, use the **debug cch323 h225** privileged EXEC command. Use the **no** form of this command to disable debugging output.

[no] **debug cch323 h225**

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### State Descriptions

The state definitions of the different states of the H.225 state machine are as follows:

- **H225\_IDLE**—This is the initial state of the H.225 state machine. The H.225 state machine is in this state before issuing a call setup request (for the outbound IP call case) or ready to receive an incoming IP call.
- **H225\_SETUP**—This is the call setup state. The state machine transitions to this state after sending out a call setup request, or after the reception of an incoming call indication.
- **H225\_ALERT**—This is the call alerting state. The state machine transitions to this state after sending the alerting message or after the reception of an alerting message from the peer.
- **H225\_CALLPROC**—This is the call proceeding state.
- **H225\_ACTIVE**—This is the Call connected state. In this state, the call is active. The state machine transitions to this state after sending the connect message to the peer or after the reception of the connect message from the peer.
- **H225\_WAIT\_FOR\_ARQ**—This is the state where the H.225 state machine is waiting for the completion of the ARQ process from the RAS state machine.
- **H225\_WAIT\_FOR\_DRQ**—This is the state where the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.
- **H225\_WAIT\_FOR\_H245**—This is the state where the H.225 state machine is waiting for the success or failure from the H.245 state machine.

### Events Description

The event definitions of the different events of the H.225 state machine are as follows:

- **H225\_EVENT\_NONE**— No event.
- **H225\_EVENT\_ALERT**—This event indicates the H.225 state machine to send an alerting message to the peer.
- **H225\_EVENT\_ALERT\_IND**—This event indicates the H.225 state machine that an alerting message is received from the peer.
- **H225\_EVENT\_CALLPROC**—This event indicates the H.225 state machine to send a call proceeding message to the peer
- **H225\_EVENT\_CALLPROC\_IND**—This event indicates the H.225 state machine that a call proceeding message is received from the peer.
- **H225\_EVENT\_REJECT**—This event indicates the H.225 state machine to reject the call setup request from the peer.

- H225\_EVENT\_REJECT\_IND—This event indicates the H.225 state machine that a call setup request to the peer is rejected.
- H225\_EVENT\_RELEASE—This event indicates the H.225 state machine to send a release complete message to the peer.
- H225\_EVENT\_RELEASE\_IND—This event indicates the H.225 state machine that a release complete message is received from the peer.
- H225\_EVENT\_SETUP—This event indicates the H.225 state machine to send a setup message to the peer.
- H225\_EVENT\_SETUP\_IND—This event indicates the H.225 state machine that a setup message is received from the peer.
- H225\_EVENT\_SETUP\_CFM—This event indicates the H.225 state machine to send a connect message to the peer.
- H225\_EVENT\_SETUP\_CFM\_IND—This event indicates the H.225 state machine that a connect message from the peer.
- H225\_EVENT\_RAS\_SUCCESS—This event indicates the H.225 state machine that the pending RAS operation is successful.
- H225\_EVENT\_RAS\_FAILED—This event indicates the H.225 state machine that the pending RAS operation failed.
- H225\_EVENT\_H245\_SUCCESS—This event indicates the H.225 state machine that the pending H.245 operation is successful.
- H225\_EVENT\_H245\_FAILED—This event indicates the H.225 state machine that the pending H.245 operation failed.

## Sample Display

```

Router# debug cch323 h225
20:59:17:Set new event H225_EVENT_SETUP
20:59:17:H225 FSM:received event H225_EVENT_SETUP while at state H225_IDLE
20:59:17:Changing from H225_IDLE state to H225_SETUP state
20:59:17:cch323_h225_receiver:received msg of type SETUPCFM_CHOSEN
20:59:17:H225 FSM:received event H225_EVENT_SETUP_CFM_IND while at state
H225_SETUP
20:59:17:Changing from H225_SETUP state to H225_ACTIVE state
20:59:17:Set new event H225_EVENT_H245_SUCCESS
20:59:17:H225 FSM:received event H225_EVENT_H245_SUCCESS while at state
H225_ACTIVE
20:59:20:Set new event H225_EVENT_RELEASE
20:59:20:H225 FSM:received event H225_EVENT_RELEASE while at state
H225_ACTIVE
20:59:20:Changing from H225_ACTIVE state to H225_WAIT_FOR_DRQ state
20:59:20:Set new event H225_EVENT_RAS_SUCCESS
20:59:20:H225 FSM:received event H225_EVENT_RAS_SUCCESS while at state
H225_WAIT_FOR_DRQ
20:59:20:Changing from H225_WAIT_FOR_DRQ state to H225_IDLE state

```

## debug cch323 h245

To trace the state transition of the H.245 state machine based on the processed events, use the **debug cch323 h245** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug cch323 h245**

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

The H.245 state machines include the following three state machines:

- Master Slave Determination state machine
- Capability Exchange state machine
- Open Logical Channel state machine

### State Definitions

The definitions are listed:

- H245\_MS\_NONE— This is the initial state of the master slave determination state machine.
- H245\_MS\_WAIT—In this state, a Master Slave Determination message is sent, waiting for the reply.
- H245\_MS\_DONE— The result is in.
- H245\_CAP\_NONE—This is the initial state of the capabilities exchange state machine.
- H245\_CAP\_WAIT—In this state, a cap exchange message is sent, waiting for reply.
- H245\_CAP\_DONE—The result is in.
- H245\_OLC\_NONE—This is the initial state of the open logical channel state machine.
- H245\_OLC\_WAIT: OLC message sent, waiting for reply.
- H245\_OLC\_DONE: OLC done.

### Event Definitions

- H245\_EVENT\_MSD—Send MSD message
- H245\_EVENT\_MS\_CFM—Send MSD acknowledge message
- H245\_EVENT\_MS\_REJ—Send MSD reject message
- H245\_EVENT\_MS\_IND— Received MSD message
- H245\_EVENT\_CAP—Send CAP message
- H245\_EVENT\_CAP\_CFM—Send CAP acknowledge message
- H245\_EVENT\_CAP\_REJ—Send CAP reject
- H245\_EVENT\_CAP\_IND—Received CAP message
- H245\_EVENT\_OLC—Send OLC message
- H245\_EVENT\_OLC\_CFM—Send OLC acknowledge message

- H245\_EVENT\_OLC\_REJ—Send OLC reject message
- H245\_EVENT\_OLC\_IND—Received OLC message

## Sample Display

```

Router# debug cch323 h245
20:58:23:Changing to new event H245_EVENT_MSD
20:58:23:H245 MS FSM:received event H245_EVENT_MSD while at state
H245_MS_NONE
20:58:23:changing from H245_MS_NONE state to H245_MS_WAIT state
20:58:23:Changing to new event H245_EVENT_CAP
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP while at state
H245_CAP_NONE
20:58:23:changing from H245_CAP_NONE state to H245_CAP_WAIT state
20:58:23:cch323_h245_receiver:received msg of type
M_H245_MS_DETERMINE_INDICATION
20:58:23:Changing to new event H245_EVENT_MS_IND
20:58:23:H245 MS FSM:received event H245_EVENT_MS_IND while at state
H245_MS_WAIT
20:58:23:cch323_h245_receiver:received msg of type
M_H245_CAP_TRANSFER_INDICATION
20:58:23:Changing to new event H245_EVENT_CAP_IND
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_IND while at state
H245_CAP_WAIT
20:58:23:cch323_h245_receiver:received msg of type
M_H245_MS_DETERMINE_CONFIRM
20:58:23:Changing to new event H245_EVENT_MS_CFM
20:58:23:H245 MS FSM:received event H245_EVENT_MS_CFM while at state
H245_MS_WAIT
20:58:23:changing from H245_MS_WAIT state to H245_MS_DONE state
20:58:23:cch323_h245_receiver:received msg of type M_H245_CAP_TRANSFER_CONFIRM
20:58:23:Changing to new event H245_EVENT_CAP_CFM
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_CFM while at state
H245_CAP_WAIT
20:58:23:changing from H245_CAP_WAIT state to H245_CAP_DONE state
20:58:23:Changing to new event H245_EVENT_OLC
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC while at state
H245_OLC_NONE
20:58:23:changing from H245_OLC_NONE state to H245_OLC_WAIT state
20:58:23:cch323_h245_receiver:received msg of type
M_H245_UCHAN_ESTABLISH_INDICATION
20:58:23:Changing to new event H245_EVENT_OLC_IND
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_IND while at state
H245_OLC_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_UCHAN_ESTAB_ACK
20:58:23:Changing to new event H245_EVENT_OLC_CFM
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_CFM while at state
H245_OLC_WAIT
20:58:23:changing from H245_OLC_WAIT state to H245_OLC_DONE state

```

## debug cch323 ras

To trace the state transition of the RAS state machine based on the processed events, use the **debug cch323 ras** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug cch323 ras**

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

RAS operates in two state machines. One global state machine controls the overall RAS operation of the Gateway. The other state machine is a per call state machine that controls the active calls.

### State Definitions

The state definitions of the different states of the RAS state machine follow:

- CCH323\_RAS\_STATE\_NONE—This is the initial state of the RAS state machine.
- CCH323\_RAS\_STATE\_GRQ—The state machine is in the GRQ state. In this state, the gateway is in the process of discovering a gatekeeper.
- CCH323\_RAS\_STATE\_RRQ—The state machine is in the RRQ state. In this state, the gateway is in the process of registering with a gatekeeper.
- CCH323\_RAS\_STATE\_IDLE—The global state machine is in the idle state.
- CCH323\_RAS\_STATE\_URQ—The state machine is in the URQ state. In this state, the gateway is in the process of unregistering with a gatekeeper.
- CCH323\_RAS\_STATE\_ARQ—The per call state machine is in the process of admitting a new call.
- CCH323\_RAS\_STATE\_ACTIVE—The per call state machine is in the call active state.
- CCH323\_RAS\_STATE\_DRQ—The per call state machine is in the process of disengaging an active call.

### Event Definitions

These are the event definitions of the different states of the RAS state machine:

- CCH323\_RAS\_EVENT\_NONE—Nothing
- CCH323\_RAS\_EVENT\_GWUP—Gateway is coming up
- CCH323\_RAS\_EVENT\_GWDWN—Gateway is going down
- CCH323\_RAS\_EVENT\_NEWCALL:—New call
- CCH323\_RAS\_EVENT\_CALLDISC—Call disconnect
- CCH323\_RAS\_EVENT\_GCF—Received GCF
- CCH323\_RAS\_EVENT\_GRJ—Received GRJ
- CCH323\_RAS\_EVENT\_ACF—Received ACF
- CCH323\_RAS\_EVENT\_ARJ—Received ARJ
- CCH323\_RAS\_EVENT\_SEND\_RRQ—Send RRQ

- CCH323\_RAS\_EVENT\_RCF—Received RCF
- CCH323\_RAS\_EVENT\_RRJ—Received RRJ
- CCH323\_RAS\_EVENT\_SEND\_URQ—Send URQ
- CCH323\_RAS\_EVENT\_URQ—Received URQ
- CCH323\_RAS\_EVENT\_UCF—Received UCF
- CCH323\_RAS\_EVENT\_SEND\_UCF—Send UCF
- CCH323\_RAS\_EVENT\_URJ—Received URJ
- CCH323\_RAS\_EVENT\_BCF—Received BCF
- CCH323\_RAS\_EVENT\_BRJ—Received BRJ
- CCH323\_RAS\_EVENT\_DRQ—Received DRQ
- CCH323\_RAS\_EVENT\_DCF—Received DCF
- CCH323\_RAS\_EVENT\_SEND\_DCF—Send DCF
- CCH323\_RAS\_EVENT\_DRJ—Received DRJ
- CCH323\_RAS\_EVENT\_IRQ—Received IRQ
- CCH323\_RAS\_EVENT\_IRR—Send IRR
- CCH323\_RAS\_EVENT\_TIMEOUT—Message timeout

## Sample Display

```

Router# debug cch323 ras
20:58:49:Changing to new event CCH323_RAS_EVENT_SEND_RRQ
cch323_run_ras_sm:received event CCH323_RAS_EVENT_SEND_RRQ while at
CCH323_RAS_STATE_IDLE state
cch323_run_ras_sm:changing to CCH323_RAS_STATE_RRQ state
cch323_ras_receiver:received msg of type RCF_CHOSEN
cch323_run_ras_sm:received event CCH323_RAS_EVENT_RCF while at CCH323_RAS_STATE_RRQ
state
cch323_run_ras_sm:changing to CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_NEWCALL while at
CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_ARQ
cch323_ras_receiver:received msg of type ACF_CHOSEN
20:58:59:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_ACF while at
CCH323_RAS_STATE_ARQ state
20:58:59:cch323_percall_ras_sm:changing to new state
CCH323_RAS_STATE_ACTIVE
20:59:02:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_CALLDISC while
at CCH323_RAS_STATE_ACTIVE state
20:59:02:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_DRQ
cch323_ras_receiver:received msg of type DCF_CHOSEN
20:59:02:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_DCF while at
CCH323_RAS_STATE_DRQ state
20:59:02:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_IDLE
20:59:04:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_IRR while at
CCH323_RAS_STATE_ACTIVE state
20:59:04:cch323_percall_ras_sm:changing to new state
CCH323_RAS_STATE_ACTIVE

```

## debug h225

To display additional information about the actual contents of H.225 RAS messages, use the **debug h225** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug h225 {asn1 | events}**

### Syntax Description

<b>asn1</b>	Indicates that only the ASN.1 contents of any H.225 message sent or received will be displayed.
<b>events</b>	Indicates that key Q.931 events that occur when placing an H.323 call from one gateway to another will be displayed.

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2

Both versions of the debug H225 command display information about H.225 messages. H.225 messages are used to exchange RAS information between gateways and gatekeepers as well as to exchange Q.931 information between gateways.

The **debug h225 events** command displays key Q.931 events that occur when placing an H.323 call from one gateway to another. Q.931 events are carried in H.225 messages. This command enables you to monitor Q.931 state changes such as setup, alert, connected, and released.

---

**Note** Although the debug information includes the hexadecimal output of the entire H.225 message, only the key state changes are decoded.

---

The **debug h225 asn1** command displays the ASN.1 contents of any H.225 message sent or received that contains ASN.1 content. Not all H.225 messages contain ASN.1 content. Some messages contain both Q.931 information and ASN.1 information; if you enter this command, only ASN.1 information will be displayed.

## Sample Display

The following sample display for the **debug h225 events** command shows a call being placed from gateway GW13 to gateway GW14. Before the call was placed, the gateway exchanged RAS messages with the gatekeeper. Because RAS messages do not contain Q.931 information, these messages do not appear in this output.

```
Router# debug h225 events
H.225 Event Messages debugging is on
Router#

*Mar 2 02:47:14.689:      H225Lib::h225TConn:connect in progress on socket [2]
*Mar 2 02:47:14.689:      H225Lib::h225TConn:Q.931 Call State is initialized to be
[Null].
*Mar 2 02:47:14.697:Hex representation of the SETUP TPKT to
send.0300004D080200DC05040380C0A36C0991313323313333303070099131342331343330307E00260500
80060008914A000102004B1F5E5D8990006C000000005BF7454000C070000000000000
*Mar 2 02:47:14.701:
*Mar 2 02:47:14.701:      H225Lib::h225SetupRequest:Q.931 SETUP sent from socket [2]
*Mar 2 02:47:14.701:      H225Lib::h225SetupRequest:Q.931 Call State changed to [Call
Initiated].
*Mar 2 02:47:14.729:Hex representation of the received
TPKT03000021080280DC013401017E0012050340060008914A000100000109350E2B28
*Mar 2 02:47:14.729:
*Mar 2 02:47:14.729:      H225Lib::h225RecvData:Q.931 ALERTING received from socket [2]
*Mar 2 02:47:14.729:      H225Lib::h225RecvData:Q.931 Call State changed to [Call
Delivered].
*Mar 2 02:47:17.565:Hex representation of the received
TPKT03000034080280DC07040380C0A37E0023050240060008914A0001000109350E2B2802004B1F5E5D899
0006C00000000005BF7454
*Mar 2 02:47:17.569:
*Mar 2 02:47:17.569:      H225Lib::h225RecvData:Q.931 CONNECT received from socket [2]
*Mar 2 02:47:17.569:      H225Lib::h225RecvData:Q.931 Call State changed to [Active].
*Mar 2 02:47:23.273:Hex representation of the received
TPKT0300001A080280DC5A080280107E000A050500060008914A0001
*Mar 2 02:47:23.273:
*Mar 2 02:47:23.273:      H225Lib::h225RecvData:Q.931 RELEASE COMPLETE received from
socket [2]
*Mar 2 02:47:23.273:      H225Lib::h225RecvData:Q.931 Call State changed to [Null].
*Mar 2 02:47:23.293:Hex representation of the RELEASE COMPLETE TPKT to
send.0300001A080200DC5A080280107E000A050500060008914A0001
*Mar 2 02:47:23.293:
*Mar 2 02:47:23.293:      H225Lib::h225TerminateRequest:Q.931 RELEASE COMPLETE sent
from socket [2]. Call state changed to [Null].
*Mar 2 02:47:23.293:      H225Lib::h225TClose:TCP connection from socket [2] closed
```

The following output shows the same call being placed from gateway GW13 to gateway GW14 using the **debug h225 asn1** command. The output is very long but you can track the following information:

- The admission request to the gatekeeper.
- The admission confirmation from the gatekeeper.
- The ASN.1 portion of the H.225/Q.931 setup message from the calling gateway to the called gateway.
- The ASN.1 portion of the H.225/Q.931 setup response from the called gateway, indicating that the call has proceeded to alerting state.
- The ASN.1 portion of the H.225/Q.931 message from the called gateway, indicating that the call has been connected.

- The ASN.1 portion of the H.225/Q.931 message from the called gateway, indicating that the call has been released.
- The ANS.1 portion of the H.225 RAS message from the calling gateway to the gatekeeper, informing it that the call has been disengaged.
- The ASN.1 portion of the H.225 RAS message from the gatekeeper to the calling gateway, confirming the disengage request.
- The ASN.1 portion of the H.225/Q.931 release complete message sent from the called gateway to the calling gateway.

```

Router# debug h225 asn1
H.225 ASN1 Messages debugging is on
Router#

value RasMessage ::= admissionRequest :
*Mar 2 02:48:18.445: {
*Mar 2 02:48:18.445:   requestSeqNum 03320,
*Mar 2 02:48:18.445:   callType pointToPoint :NULL,
*Mar 2 02:48:18.445:   callModel direct :NULL,
*Mar 2 02:48:18.445:   endpointIdentifier "60D6BA4C00000001",
*Mar 2 02:48:18.445:   destinationInfo
*Mar 2 02:48:18.445:   {
*Mar 2 02:48:18.445:     e164 : "14#14300"
*Mar 2 02:48:18.445:   },
*Mar 2 02:48:18.449:   srcInfo
*Mar 2 02:48:18.449:   {
*Mar 2 02:48:18.449:     e164 : "13#13300"
*Mar 2 02:48:18.449:   },
*Mar 2 02:48:18.449:   bandwidth 0640,
*Mar 2 02:48:18.449:   callReferenceValue 0224,
*Mar 2 02:48:18.449:   conferenceID '4B1F5E5D899000720000000005C067A4'H,
*Mar 2 02:48:18.449:   activeMC FALSE,
*Mar 2 02:48:18.449:   answerCall FALSE
*Mar 2 02:48:18.449: }
*Mar 2 02:48:18.449:25800CF7 00F00036 00300044 00360042 00410034 00430030 00300030
00300030
00300030 00310103 80470476 33010380 46046633 40028000 E04B1F5E 5D899000
72000000 0005C067 A400
29000CF7 40028000 0109350E 06B80077
value RasMessage ::= admissionConfirm :
*Mar 2 02:48:18.469: {
*Mar 2 02:48:18.469:   requestSeqNum 03320,
*Mar 2 02:48:18.469:   bandwidth 0640,
*Mar 2 02:48:18.469:   callModel direct :NULL,
*Mar 2 02:48:18.469:   destCallSignalAddress ipAddress :
*Mar 2 02:48:18.469:   {
*Mar 2 02:48:18.469:     ip '0109350E'H,
*Mar 2 02:48:18.469:     port 01720
*Mar 2 02:48:18.469:   },
*Mar 2 02:48:18.469:   irrFrequency 0120
*Mar 2 02:48:18.473: }
*Mar 2 02:48:18.473:value H323-UserInformation ::=
*Mar 2 02:48:18.481: {
*Mar 2 02:48:18.481:   h323-uu-pdu
*Mar 2 02:48:18.481:   {
*Mar 2 02:48:18.481:     h323-message-body setup :
*Mar 2 02:48:18.481:     {
*Mar 2 02:48:18.481:       protocolIdentifier { 0 0 8 2250 0 1 },
*Mar 2 02:48:18.481:       sourceInfo
*Mar 2 02:48:18.481:       {
*Mar 2 02:48:18.481:         terminal
*Mar 2 02:48:18.481:         {
*Mar 2 02:48:18.481:           },

```

```

*Mar 2 02:48:18.481:          mc FALSE,
*Mar 2 02:48:18.481:          undefinedNode FALSE
*Mar 2 02:48:18.481:        },
*Mar 2 02:48:18.481:        activeMC FALSE,
*Mar 2 02:48:18.481:        conferenceID '4B1F5E5D899000720000000005C067A4'H,
*Mar 2 02:48:18.481:        conferenceGoal create :NULL,
*Mar 2 02:48:18.485:        callType pointToPoint :NULL,
*Mar 2 02:48:18.485:        sourceCallSignalAddress ipAddress :
*Mar 2 02:48:18.485:          {
*Mar 2 02:48:18.485:            ip '00000000'H,
*Mar 2 02:48:18.485:            port 00
*Mar 2 02:48:18.485:          }
*Mar 2 02:48:18.485:        }
*Mar 2 02:48:18.485:      }
*Mar 2 02:48:18.485:    }
*Mar 2 02:48:18.485:00800600 08914A00 0102004B 1F5E5D89 90007200 00000005 C067A400
0C070000
00000000 00
value H323-UserInformation ::=
*Mar 2 02:48:18.525:{
*Mar 2 02:48:18.525: h323-uu-pdu
*Mar 2 02:48:18.525: {
*Mar 2 02:48:18.525:   h323-message-body alerting :
*Mar 2 02:48:18.525:   {
*Mar 2 02:48:18.525:     protocolIdentifier { 0 0 8 2250 0 1 },
*Mar 2 02:48:18.525:     destinationInfo
*Mar 2 02:48:18.525:     {
*Mar 2 02:48:18.525:       mc FALSE,
*Mar 2 02:48:18.525:       undefinedNode FALSE
*Mar 2 02:48:18.525:     },
*Mar 2 02:48:18.525:     h245Address ipAddress :
*Mar 2 02:48:18.525:     {
*Mar 2 02:48:18.525:       ip '0109350E'H,
*Mar 2 02:48:18.525:       port 011050
*Mar 2 02:48:18.525:     }
*Mar 2 02:48:18.525:   }
*Mar 2 02:48:18.525: }
*Mar 2 02:48:18.525:}
*Mar 2 02:48:18.525:value H323-UserInformation ::=
*Mar 2 02:48:22.753:{
*Mar 2 02:48:22.753: h323-uu-pdu
*Mar 2 02:48:22.753: {
*Mar 2 02:48:22.753:   h323-message-body connect :
*Mar 2 02:48:22.753:   {
*Mar 2 02:48:22.753:     protocolIdentifier { 0 0 8 2250 0 1 },
*Mar 2 02:48:22.753:     h245Address ipAddress :
*Mar 2 02:48:22.753:     {
*Mar 2 02:48:22.753:       ip '0109350E'H,
*Mar 2 02:48:22.753:       port 011050
*Mar 2 02:48:22.753:     },
*Mar 2 02:48:22.753:     destinationInfo
*Mar 2 02:48:22.753:     {
*Mar 2 02:48:22.753:       terminal
*Mar 2 02:48:22.753:       {
*Mar 2 02:48:22.753:         },
*Mar 2 02:48:22.757:       mc FALSE,
*Mar 2 02:48:22.757:       undefinedNode FALSE
*Mar 2 02:48:22.757:     },
*Mar 2 02:48:22.757:     conferenceID '4B1F5E5D899000720000000005C067A4'H
*Mar 2 02:48:22.757:   }
*Mar 2 02:48:22.757: }
*Mar 2 02:48:22.757:}
*Mar 2 02:48:22.757:value H323-UserInformation ::=
*Mar 2 02:48:27.109:{
*Mar 2 02:48:27.109: h323-uu-pdu

```

```

*Mar 2 02:48:27.109: {
*Mar 2 02:48:27.109:   h323-message-body releaseComplete :
*Mar 2 02:48:27.109:     {
*Mar 2 02:48:27.109:       protocolIdentifier { 0 0 8 2250 0 1 }
*Mar 2 02:48:27.109:     }
*Mar 2 02:48:27.109: }
*Mar 2 02:48:27.109:}
*Mar 2 02:48:27.109:value RasMessage ::= disengageRequest :
*Mar 2 02:48:27.117: {
*Mar 2 02:48:27.117:   requestSeqNum 03321,
*Mar 2 02:48:27.117:   endpointIdentifier "60D6BA4C00000001",
*Mar 2 02:48:27.117:   conferenceID '4B1F5E5D899000720000000005C067A4'H,
*Mar 2 02:48:27.121:   callReferenceValue 0224,
*Mar 2 02:48:27.121:   disengageReason normalDrop :NULL
*Mar 2 02:48:27.121: }
*Mar 2 02:48:27.121:3C0CF81E 00360030 00440036 00420041 00340043 00300030 00300030
00300030
00300031 4B1F5E5D 89900072 00000000 05C067A4 00E020
400CF8
value RasMessage ::= disengageConfirm :
*Mar 2 02:48:27.133: {
*Mar 2 02:48:27.133:   requestSeqNum 03321
*Mar 2 02:48:27.133: }
*Mar 2 02:48:27.133:value H323-UserInformation ::=
*Mar 2 02:48:27.133:{
*Mar 2 02:48:27.133: h323-uu-pdu
*Mar 2 02:48:27.133: {
*Mar 2 02:48:27.133:   h323-message-body releaseComplete :
*Mar 2 02:48:27.133:     {
*Mar 2 02:48:27.133:       protocolIdentifier { 0 0 8 2250 0 1 }
*Mar 2 02:48:27.133:     }
*Mar 2 02:48:27.133: }
*Mar 2 02:48:27.133:}
*Mar 2 02:48:27.133:05000600 08914A00 01
.

```

## debug ras

To display the types and addressing of RAS messages sent and received, use the **debug voip ccapi** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug ras**

### Usage Guidelines

The debug output lists the message type using mnemonics defined in ITU-T specification H.225.

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### Sample Display

In the following output, gateway GW13.cisco.com sends a RAS registration request message (RRQ) to gatekeeper GK15.cisco.com at IP address 172.9.53.15. GW13.cisco.com then receives a registration confirmation (RCF) message from the gatekeeper. If there is no response, it could mean that the gatekeeper is offline or improperly addressed. If you receive a reject message (RRJ), it could mean that the gatekeeper is unable to handle another gateway or that the registration information is incorrect.

```
Router# debug ras
*Mar 13 19:53:34.231:      RASLib::ras_sendto:msg length 105 from
                        172.9.53.13:8658 to 1.9.53.15:1719
*Mar 13 19:53:34.231:      RASLib::RASSendRRQ:RRQ (seq# 36939) sent
                        to 172.9.53.15
*Mar 13 19:53:34.247:      RASLib::RASRecvData:successfully rcvd
                        message of length 105 from 172.9.53.15:1719
*Mar 13 19:53:34.251:      RASLib::RASRecvData:RCF (seq# 36939) rcvd
                        from [172.9.53.15:1719] on sock [0x6168356C
```

## debug voip aaa

To enable debugging messages for gateway aaa to be output to the system console, use the **debug voip aaa** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug voip aaa**

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

## debug voip ccapi

To debug the call control API, use the **debug voip ccapi** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug voip ccapi**

### Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(6)NA2.

### Sample Display

```
Router# show debug
voip:
  voip ccAPI function enter/exit debugging is on
Oct 9 17:39:20.267:cc_api_call_setup_ind (vdbPtr=0x60ED5134,
callInfo={called=3001, calling=4004, fdest=0 peer_tag=1},
callID=0x6104B374)
Oct 9 17:39:20.275:cc_process_call_setup_ind (event=0x60D45CF0) handed
call to app "sess"
Oct 9 17:39:20.279:ccAppInitialize (name=App for callId 3
, appHandle=0x6103DD44)
Oct 9 17:39:20.279:ccCallSetContext (callID=0x3, context=0x6103DD3C)
Oct 9 17:39:20.279:ccCallSetupAck (callID=0x3)
Oct 9 17:39:20.279:ccGenerateTone (callID=0x3 tone=8)
Oct 9 17:39:20.279:ccCallApp (callID=0x3)
Oct 9 17:39:20.279:ccCallSetContext (callID=0x3, context=0x60DC4594)
00:11:31:%RADIUS-6-SERVERALIVE:Radius server 171.69.184.73 is
responding
again (previously dead).
Oct 9 17:39:22.808:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=1, mode=0)
Oct 9 17:39:23.069:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=1, mode=0)
Oct 9 17:39:23.399:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=5, mode=0)
Oct 9 17:39:23.652:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=1, mode=0)
Oct 9 17:39:24.041:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=0, mode=0)
Oct 9 17:39:24.294:cc_api_call_digit (vdbPtr=0x60ED5134, callID=0x3,
digit=0, mode=0)
Oct 9 17:39:24.294:ccCallAppReturn (callID=0x3)
Oct 9 17:39:24.294:ccCallApp (callID=0x3)
Oct 9 17:39:24.294:ccCallSetContext (callID=0x3, context=0x6105DC90)
Oct 9 17:39:24.294:ccCallProceeding (callID=0x3, prog_ind=0x0)
Oct 9 17:39:24.294:ccCallSetupRequest (peer=0x60FE4068, dest=,
params=0x6105DB70 mode=0, *callID=0x60D50978)
Oct 9 17:39:24.294:callingNumber=4004, calledNumber=115100,
redirectNumber=
Oct 9 17:39:24.294:accountNumber=, finalDestFlag=0,
guid=3c85.5d28.2861.0004.0000.0000.000a.8dfc
Oct 9 17:39:24.294:peer_tag=115
Oct 9 17:39:24.294:ccIFCallSetupRequest:(vdbPtr=0x60D4A268, dest=,
callParams={called=115100, calling=4004, fdest=0, voice_peer_tag=115},
mode=0x0)
Oct 9 17:39:24.294:ccCallSetContext (callID=0x4, context=0x6105DD78)
Oct 9 17:39:26.350:cc_api_call_alert(vdbPtr=0x60D4A268, callID=0x4,
prog_ind=0x8, sig_ind=0x0)
Oct 9 17:39:26.350:ccCallAlert (callID=0x3, prog_ind=0x8, sig_ind=0x0)
Oct 9 17:39:26.350:ccConferenceCreate (confID=0x60D509C8, callID1=0x3,
```

```
callID2=0x4, tag=0x0)
Oct 9 17:39:26.350:cc_api_bridge_done (confID=0x1, srcIF=0x60D4A268,
srcCallID=0x4, dstCallID=0x3, disposition=0, tag=0x0)
Oct 9 17:39:26.350:cc_api_bridge_done (confID=0x1, srcIF=0x60ED5134,
srcCallID=0x3, dstCallID=0x4, disposition=0, tag=0x0)
Oct 9 17:39:26.350:cc_api_caps_ind (dstVdbPtr=0x60D4A268,
dstCallId=0x4,srcCallId=0x3, caps={codec=0x7, fax_rate=0x7F, vad=0x3})
Oct 9 17:39:26.350:cc_api_caps_ind (dstVdbPtr=0x60ED5134,
dstCallId=0x3,srcCallId=0x4, caps={codec=0x4, fax_rate=0x2, vad=0x2})
Oct 9 17:39:26.350:cc_api_caps_ack (dstVdbPtr=0x60ED5134,
dstCallId=0x3,srcCallId=0x4, caps={codec=0x4, fax_rate=0x2, vad=0x2})
Oct 9 17:39:26.350:cc_api_caps_ack (dstVdbPtr=0x60D4A268,
dstCallId=0x4,srcCallId=0x3, caps={codec=0x4, fax_rate=0x2, vad=0x2})
Oct 9 17:39:26.430:cc_api_call_connected(vdbPtr=0x60D4A268,
callID=0x4)
Oct 9 17:39:26.430:ccCallConnect (callID=0x3)
Oct 9 17:39:26.430:ccCallAppReturn (callID=0x3)
Oct 9 17:39:26.430:ccCallSetContext (callID=0x4, context=0x6103DD3C)
Oct 9 17:39:30.683:cc_api_call_disconnected(vdbPtr=0x60D4A268,
callID=0x4, cause=0x10)
Oct 9 17:39:30.683:ccCallDisconnect (callID=0x4, cause=0x10 tag=0x0)
Oct 9 17:39:30.683:ccConferenceDestroy (confID=0x1, tag=0x0)
Oct 9 17:39:30.687:cc_api_bridge_done (confID=0x1, srcIF=0x60D4A268,
srcCallID=0x4, dstCallID=0x3, disposition=0 tag=0x0)
Oct 9 17:39:30.727:cc_api_call_disconnect_done(vdbPtr=0x60D4A268,
callID=0x4, disp=0, tag=0x0)
Oct 9 17:39:30.727:cc_api_bridge_done (confID=0x1, srcIF=0x60ED5134,
srcCallID=0x3, dstCallID=0x4, disposition=0 tag=0x0)
Oct 9 17:39:30.727:ccCallDisconnect (callID=0x3, cause=0x10 tag=0x0)
Oct 9 17:39:30.779:cc_api_call_disconnect_done(vdbPtr=0x60ED5134,
callID=0x3, disp=0, tag=0x0)
00:11:42:%LINK-3-UPDOWN:Interface Serial0:18, changed state to down
```

## debug voip ivr

To debug the IVR application, use the **debug voip ivr** privileged EXEC command. Use the **no** form of this command to disable debugging output.

**[no] debug voip ivr [states | error | all]**

### Syntax Description

<b>states</b>	(Optional) Displays verbose information about how IVR is handling each call.
<b>error</b>	(Optional) Displays information only if an error occurred.
<b>all</b>	(Optional) Displays both <b>states</b> and <b>error</b> messages.

### Usage Guidelines

IVR debug messages will be displayed when a call is being actively handled by the IVR scripts. Error output should only occur if something is not working or an error condition has been raised. States output supplies information about the current status of the IVR script and the different events which are occurring in that state.

This command first appeared in Cisco IOS Release 11.3(6)NA2.

## What to Do Next

For more information on the Voice Service Provider, refer to the *Voice over IP for the Cisco AS5300 Software Configuration Guide*.

For more information on the gatekeeper, refer to the *Configuring H.323 VoIP Gatekeeper for Cisco Access Platforms* feature module for Cisco IOS Release 11.3(7)NA.

For more information on the gateway, refer to the *Configuring H.323 VoIP Gateway for Cisco Access Platforms* feature module for Cisco IOS Release 11.3(7)NA.

The following information is supplied as background information only. It is intended to supplement basic understanding for these features and is not all inclusive on the subject matter.

You can supplement this feature module with *Release Notes for Cisco AS5300 for Cisco IOS Release 12.0T*, which documents the release from which this special release is derived.