Cisco IOS Tcl IVR and VoiceXML Application Guide

Cisco IOS Release 15.4(1)T

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

Text Part Number: OL-11176-02

Licensing Information

The following license agreement covers the XML parser code that is used by the VoiceXML interpreter and is contained in the Cisco IOS software image.

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

Alternatively, the contents of this file may be used under the terms of the GNU General Public License (the "GPL"), in which case the provisions of the GPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the GPL and not to allow others to use your version of this file under the MPL, indicate your decision by deleting the provisions above and replacing them with the notice and other provisions required by the GPL. If you do not delete the provisions above, a recipient may use your version of this file under either the MPL or the GPL.

The Original Code is expat.

The Initial Developer of the Original Code is James Clark.

Portions created by James Clark are Copyright (C) 1998, 1999

James Clark. All Rights Reserved.

Contributor(s): Jenny Yao from Cisco Systems, Inc.

Modification of the source code made by Jenny Yao is controlled by definition of "TARGET_CISCO".

Cisco Modification of Expat Source Code

File Modified	**Description of Modification**
hashtable.h	Remove include file <stddef.h>.
xmldef.h	Include Cisco IOS <master.h> file.
hashtable.c	Change filename from hashtable.c to xmlhashtable to avoid a generic filename.
xmlparse.c	Remove some variables defined but not used, like errorProcessor and internalEnc. Type cast (int) for sizeof() during arithmetic operation. Add new routine XML_realloc(), replacing realloc() in this file because usage of realloc() could cause memory leakage.
xmltok.c	Add argument type void for function with empty argument. Type cast (int) for sizeof() during arithmetic operation.
xmltok.h	Add argument type void for function with empty argument.
xmltok_impl.c	Initialize variable open to 0.
filexmltok_impl.h	Remove file including for <stddef.h>.

Cisco IOS Tcl IVR and VoiceXML Application Guide

CONTENTS

CHAPTER 1 **Cisco IOS Tcl IVR and VoiceXML Feature Roadmap** 1-1

Platforms and Cisco IOS Software Images 1-2

Cisco IOS Tcl IVR and VoiceXML Feature List 1-2

CHAPTER 2 **Overview of Cisco IOS Tcl IVR and VoiceXML Applications** 2-1

Contents 2-2

Prerequisites for Implementing Tcl IVR and VoiceXML Applications 2-2

Memory Requirements for Interpreting a VoiceXML Document 2-2

Gateway Prerequisite Configuration 2-2

VCWare Version on Cisco AS5300 2-3

Call Admission Control 2-3

HTTP Server 2-4

Installing Apache PHP Server Software 2-4

Enabling Chunked Transfer 2-5

Additional References 2-5

Related Documents 2-5

Related Websites 2-7

Standards 2-7

MIBs 2-8

RFCs 2-8

Technical Assistance 2-9

CHAPTER 3 **Configuring Basic Functionality for Tcl IVR and VoiceXML Applications** 3-1

Contents 3-1

Prerequisites for Configuring Basic Functionality for Tcl IVR and VoiceXML Applications 3-2

Recommended Knowledge 3-2

VoiceXML Document Development 3-2

Restrictions for Configuring Basic Functionality for Tcl IVR and VoiceXML Applications 3-3

General Restrictions 3-3

DTMF Relay Restrictions 3-4

ISDN Overlap Restrictions 3-4

Information About Configuring Basic Functionality for Tcl IVR and VoiceXML Applications 3-4

Benefits 3-5

Feature Design of Voice Applications 3-6

VoiceXML for Cisco IOS Feature Overview **3-6**

Tcl IVR 2.0 Overview **3-8**

MGCP Scripting Overview **3-9**

Call Handling Between Tcl and VoiceXML Applications **3-9**

HTTP Client Support **3-11**

HTTP over Secure Socket Layer **3-13**

Cisco IOS Release 12.3(14)T and Later Releases Voice Application CLI Structure Changes **3-13**

Service and Package Parameters **3-13**

Parameter Groups **3-14**

Upgrading to Cisco IOS Release 12.3(14)T **3-15**

Obsolete, Modified, and Replaced Commands **3-15**

New Commands **3-17**

VoiceXML Before Version 2.0 and Version 2.0 Behavior **3-19**

VoiceXML 2.0 Changes **3-20**

Obsolete Elements **3-20**

New Elements **3-20**

New Attributes **3-20**

New Shadow Variables **3-21**

New Properties **3-21**

New Session Variables **3-21**

Error Events **3-21**

Behavior Changes **3-23**

New Functionality **3-24**

New Restrictions **3-24**

How to Configure Basic Functionality for a Tcl IVR or VoiceXML Application **3-24**

Loading a Service onto the Gateway **3-25**

Verifying Loading of Service **3-26**

Troubleshooting Tips **3-28**

Defining a Package on the Gateway **3-28**

Verifying Package Definition **3-29**

Troubleshooting Tips **3-30**

Configuring Service Parameters **3-31**

Prerequisites **3-31**

Configuring Package Parameters **3-32**

Prerequisites **3-32**

Using Parameterspaces **3-33**

Verifying Parameterspace Configuration **3-36**

Troubleshooting Tips **3-36**

Defining Parameter Groups **3-36**

Verifying Parameter Group Definition **3-37**

BETA DRAFT - CISCO CONFIDENTIAL

Troubleshooting Tips 3-38
Using Parameter Groups 3-38
Verifying Parameter Group Configuration 3-39
Troubleshooting Tips 3-40
Configuring an Inbound Application 3-40
Role of Dial Peers in Configuring Voice Applications 3-40
How Voice Applications are Matched to Called Numbers 3-41
Direct Inward Dialing (DID) Behavior 3-42
Troubleshooting Tips 3-43
Verifying an Inbound Application Configuration 3-44
Verifying the Gateway Configuration by Using a Sample Application 3-45
Troubleshooting Dial Peer Matching 3-49
Troubleshooting Dial Peer Configuration 3-50
Configuring an Outbound Application 3-52
Prerequisites 3-52
Outbound Voice Applications 3-52
Call Scenario for an Outbound Application 3-53
Configuring an Outbound VoIP Dial Peer for Call Transfers 3-55
Verifying the Outbound Application Configuration 3-58
Troubleshooting Tips 3-59
Configuring VoiceXML with the SIP Phone 3-60
Configuring a DNIS Map for VoiceXML Applications 3-61
DNIS Maps 3-61
Using Cisco IOS Software or Text Files for DNIS Maps 3-61
DNIS Map Limits 3-62
URLs in DNIS Maps 3-62
Verifying DNIS Map Configuration 3-64
Modifying HTTP Client Settings 3-66
Configuring HTTPS 3-67
Configuring an Exclusive Trustpoint and Cipher Suites 3-67
Configuring the HTTP Client to Obtain a Certificate 3-68
Verifying HTTP Client Settings 3-69
Configuration Examples for Tcl IVR and VoiceXML Applications 3-71
Tcl IVR 2.0 Examples 3-71
GW1 IVR Configuration: Example 3-71
GW2 IVR Configuration: Example 3-74
Inbound VoiceXML Application: Example 3-77
Outbound VoiceXML Application with DNIS Map: Example 3-79
Where to Go Next 3-81

Additional References 3-82

CHAPTER 4 **Configuring Audio File Properties for Tcl IVR and VoiceXML Applications** 4-1

Contents 4-1

Prerequisites for Audio Files 4-2

Restrictions for Audio Files 4-2

Recording and Playback Restrictions 4-2

Codec Restrictions 4-3

Information About Audio File Properties for Tcl and VoiceXML Applications 4-4

Audio File Playout Methods 4-4

Dynamic Prompts 4-5

Volume and Rate Controls for Audio Prompts using VoiceXML 4-6

How to Configure Audio File Properties for Applications 4-6

Specifying a New Language Module for Dynamic Prompts 4-7

Verifying Configured Languages 4-8

Troubleshooting Tips 4-8

Setting Language and Location of Audio Files for Dynamic Prompts 4-8

Verifying Language and Location of Audio Files for Dynamic Prompts 4-10

Setting Memory Recording Limits 4-11

Troubleshooting Tips 4-12

Verifying Prompt Playout 4-12

Configuring Audio Prompt Streaming 4-13

Configuring RTSP Live Streaming 4-13

Modifying Codec Complexity on the Cisco 3600 Series 4-13

Configuration Examples for Audio Files 4-16

Where to Go Next 4-16

Additional References 4-17

CHAPTER 5 **Configuring VoiceXML Voice Store and Forward** 5-1

Contents 5-1

Prerequisites for VoiceXML Voice Store and Forward 5-2

Restrictions for VoiceXML Voice Store and Forward 5-2

Information About VoiceXML Store and Forward 5-3

VoiceXML Voice Store and Forward 5-3

VoiceXML Audio Recording Scenario 5-4

Audio File Formats Supported for Recording and Playback 5-5

VoiceXML Recording Locations 5-5

Codec Support for Audio Recording 5-5

BETA DRAFT - CISCO CONFIDENTIAL

Codec Mappings in Audio Recordings 5-7
Correction Utility for Audio File Headers 5-7
.AU File Format Correction 5-7
.WAV File Format Corrections 5-8
How to Configure VoiceXML Voice Store and Forward 5-8
Configuring the On-Ramp Gateway for VoiceXML Voice Store and Forward 5-8
On-Ramp Gateway for VoiceXML Voice Store and Forward 5-9
Configuring the Interface Type for Sending Voice Mail 5-9
Configuring the Sending MTA 5-9
Configuring the POTS Dial Peer 5-11
Verifying the On-Ramp Gateway Configuration 5-12
Troubleshooting Recording Configuration 5-15
Configuring the Off-Ramp Gateway to Place a Call 5-16
Off-Ramp Gateway Placing a Call 5-16
Off-Ramp Mail Trigger Call Scenario 5-17
MDNs 5-18
DSNs 5-18
Downloading the Tcl Script for Off-Ramp Mail Application 5-19
Loading the Mail Application onto the Gateway 5-19
Configuring the Interface Type for Receiving Voice Mail 5-21
Configuring the Receiving MTA for Voice Store and Forward 5-21
Configuring the POTS Dial Peer 5-22
Configuring the MMoIP Dial Peer 5-23
Verifying the Off-Ramp Gateway Configuration 5-24
Configuration Examples for VoiceXML Voice Store and Forward 5-27
VoiceXML Voice Store and Forward on Cisco AS5300 Example 5-27
VoiceXML Voice Store and Forward Off-Ramp on Cisco 3600 Series Example 5-30
Where to Go Next 5-32
Additional References 5-32

CHAPTER 6 **Configuring ASR and TTS Properties** 6-1
Contents 6-1
Prerequisites for Using ASR and TTS 6-2
Restrictions for External ASR and TTS Servers 6-3
Information About Speech Recognition and Synthesis 6-4
How to Configure External Server Properties 6-5
Specifying ASR and TTS Media Server Locations 6-5
Troubleshooting Tips 6-6
Verifying the Media Server Locations 6-8

Troubleshooting ASR and TTS Server Functionality 6-8
Specifying MRCP v2 ASR and TTS Media Server Locations 6-10
Verifying the MRCP v2 Media Server Locations 6-12
Troubleshooting MRCP v2 ASR and TTS Server Functionality 6-14
Setting MRCP Client History Limits 6-17
Configuration Examples for ASR and TTS 6-18
ASR and TTS: Example 6-18
MRCP v2 ASR and TTS: Example 6-20
Where to Go Next 6-22
Additional References 6-22

CHAPTER 7 Configuring Fax Detection for VoiceXML 7-1
Contents 7-1
Prerequisites for Fax Detection for VoiceXML 7-2
Restrictions for Fax Detection for VoiceXML 7-2
Information About Fax Detection for VoiceXML 7-2
How to Configure Fax Detection for VoiceXML 7-3
Configuring Fax Detection for VoiceXML 7-3
Verifying VoiceXML Fax Detection Configuration 7-4
Configuration Examples for Fax Detection for VoiceXML 7-6
Fax Detection for VoiceXML with T.37 Store-and-Forward Fax Example 7-6
Where to Go Next 7-8
Additional References 7-9

CHAPTER 8 Configuring Media Inactive Call Detection (Silent Call Detection) 8-1
Overview of the Media Inactivity Detection Feature 8-2
Prerequisites for Media Inactive Call Detection 8-2
New Command-Line Interface in Cisco IOS Release 12.4(4)T 8-3
Restrictions for Media Inactive Call Detection 8-4
Information About Media Inactive Call Detection 8-4
Functionality of Media Inactive Call Detection in Cisco IOS Release 12.4(4)T 8-4
Modifications to Information Tags and Internal Error Codes 8-5
Configuring CallFeature Parameters for Media Inactive Call Detection 8-7
Configuring Media Inactivity Parameters at the Package Level 8-7
Configuring Media Inactivity Parameters at the Service or Application Level 8-9
Configuring Media Inactivity Parameters at the Dial-Peer Level 8-10
Configure Long Duration Call Detection in Global Configuration Mode 8-12
Configuring Long Duration Call Detection in Dial-Peer Configuration Mode 8-12

BETA DRAFT - CISCO CONFIDENTIAL

Verifying the Configuration 8-13

Examples 8-15

CHAPTER 9 **Configuring Telephony Call-Redirect Features** 9-1

Contents 9-1

Prerequisites 9-2

Restrictions 9-2

Information About Telephony Call-Redirect Features 9-2

Benefits 9-3

GTD Parameters 9-3

Release-to-Pivot 9-3

Two B-Channel Transfer 9-4

ETSI Call Transfer 9-5

How to Configure Telephony Call-Redirect Features 9-5

Configuring Call-Transfer Method for Voice Applications 9-5

Configuring the Parameter Routing Method 9-5

Configuring the Call Rerouting Method 9-7

Verifying the Call-Transfer Method for Voice Applications 9-8

Troubleshooting Tips 9-8

Enabling RTPvt 9-8

Enabling TBCT for Trunk Groups 9-9

Configuring Outbound Dial Peer for TBCT Calls 9-10

Configuring TBCT Call Limits 9-11

Terminating Billing for Active TBCT Calls 9-12

Verifying TBCT 9-12

Enabling ETSI Call Transfer 9-14

Configuration Examples 9-16

TBCT Trunk Group Example 9-16

TBCT with Notify on Clear Example 9-17

ETSI Call Transfer Example 9-17

Where to Go Next 9-18

Additional References 9-18

CHAPTER 10 **Configuring Tcl IVR 2.0 Session Interaction** 10-1

Contents 10-1

Prerequisites for Session Interaction 10-2

Restrictions for Session Interaction 10-2

Information About Session Interaction 10-2

Tcl IVR 2.0 Session Interaction and Service Registry **10-2**
Benefits of Session Interaction **10-3**
How to Configure Session Interaction **10-3**
Starting a New Tcl IVR 2.0 Application Instance (Session) **10-3**
Starting an Application Instance in the Configuration **10-4**
Starting an Application Instance in Privileged EXEC Mode **10-5**
Verifying That an Application Instance is Running **10-6**
Troubleshooting Tips **10-6**
Stopping an Application Instance **10-7**
Stopping an Application Instance in the Configuration **10-7**
Stopping an Application Instance in Privileged EXEC Mode **10-8**
Configuration Examples for Session Interaction **10-9**
Tcl IVR Application Sessions Example **10-9**
Cisco IOS Command Output **10-13**
Where to Go Next **10-14**
Additional References **10-15**

CHAPTER **11** **Configuring SIP and TEL URL Support** **11-1**
Finding Feature Information **11-1**
Contents **11-1**
Prerequisites for Configuring SIP and TEL URL Support **11-2**
Restrictions for Configuring SIP and TEL URL Support **11-2**
Information About SIP and TEL URL Support **11-2**
SIP and TEL URL Support for Voice Applications **11-2**
Benefits of SIP and TEL URL Support **11-3**
How to Configure SIP and TEL URL Support **11-3**
Configuring a Voice Class URI **11-3**
Configuring a Voice Class SIP URI **11-3**
Configuring a Voice Class TEL URL **11-6**
Configuring an Inbound Dial Peer to Match on a URI **11-7**
Prerequisites **11-8**
Configuring an Inbound Dial Peer to Match the URI in H.323 Calls **11-8**
Configuring an Inbound Dial-Peer to Match the URI on SIP Calls **11-9**
Verifying Dial-Peer Configuration for an Incoming URI Based Dial-Peer Match **11-12**
Configuring an Outbound Dial Peer for URI Destinations **11-15**
Prerequisites **11-15**
Restrictions **11-15**
Verifying Dial-Peer Configuration for an Outgoing URI **11-16**

BETA DRAFT - CISCO CONFIDENTIAL

Troubleshooting Tips 11-19
Troubleshooting URI Matching 11-19
Configuration Examples for SIP and TEL URL Support 11-21
Example: Outbound Dial-Peer Originating Gateway 11-21
Example: URIs with Header Passing Using SIP Protocol and H.323 Protocol 11-22
Example: Outbound Dial-Peer Terminating Gateway 11-26
Example: URIs with Header Passing Using SIP Protocol and H.323 Protocol 11-27
Where to Go Next 11-30
Additional References 11-31
Related Documents 11-31
MIBs 11-31
Technical Assistance 11-31
Feature Information for Configuring SIP and TEL URL Support 11-31

CHAPTER 12 **Monitoring and Troubleshooting Voice Applications** 12-1
Contents 12-2
Prerequisites for Voice Application Monitoring and Troubleshooting 12-2
Restrictions for Voice Application Monitoring and Troubleshooting 12-2
Information About Voice Application Monitoring and Troubleshooting Enhancements 12-2
Description of Voice Application Monitoring and Troubleshooting Features 12-3
Counters and Gauges for Voice Application Statistics 12-4
Monitoring Levels for Voice Applications 12-5
Application Instance Statistics 12-7
Application Interface Statistics 12-8
Benefits of Voice Application Monitoring and Troubleshooting Enhancements 12-8
Guidelines for Enabling Statistics and Event Logging for Voice Applications 12-9
Throttling Mechanism for Event Logging 12-9
Memory Requirements for Writing Event Logs to FTP 12-9
How to Configure Monitoring for Voice Applications 12-10
Enabling Event Logging and Statistics Globally for Voice Applications 12-10
Enabling Event Logging for a Specific Voice Application or Interface 12-12
Monitoring Voice Applications for Active Calls 12-14
Monitoring Voice Applications for Terminated Calls 12-15
Examples for Monitoring Voice Applications 12-16
Monitoring Voice Call Legs 12-20
What To Do Next 12-21
Clearing Event Logs and Statistics for Application Instances and Interfaces 12-22
Displaying Event Logs for Applications or Call Legs in Real-Time 12-22
Modifying Event Log Settings for Application Instances 12-23

Modifying Event Log Settings for Application Interfaces 12-25
Modifying Event Log Settings for Call Legs 12-26
Restrictions 12-26
Modifying Event Log History Limits 12-27
Configuration Examples for Monitoring Voice Applications 12-29
Enabling Event Logs and Statistics Globally: Example 12-29
Customizing Event Logs and Statistics Example 12-31
Additional References 12-32

INDEX

Cisco IOS Tcl IVR and VoiceXML Feature Roadmap

This guide collects together in one place information about Cisco IOS voice features having to do with Cisco IOS Tcl IVR and VoiceXML. It is written for developers and network administrators who are installing, configuring, and maintaining a Tcl or VoiceXML application on a Cisco voice gateway. This guide describes the tasks related to the Cisco IOS software implementation of these applications. It contains the following:

Module	Description
Overview of Cisco IOS Tcl IVR and VoiceXML Applications	Overview of the features, summarizes supported platforms, standards, MIBs, and RFCs, lists related features and documentation, and contains restrictions and prerequisites.
Configuring Basic Functionality for Tcl IVR and VoiceXML Applications	Basic steps for configuring Tcl and VoiceXML applications on the Cisco gateway.
Configuring Audio File Properties for Tcl IVR and VoiceXML Applications	Steps for configuring language modules and location of dynamic prompts, streaming of audio files, and recording limits.
Configuring VoiceXML Voice Store and Forward	Steps for configuring the VoiceXML Voice Store and Forward feature.
Configuring ASR and TTS Properties	Steps for implementing speech recognition and speech synthesis for voice applications.
Configuring Fax Detection for VoiceXML	Steps for configuring the Fax Detection for VoiceXML feature.
Configuring Media Inactive Call Detection (Silent Call Detection)	Steps for configuring the Media Inactive Call Detection (Silent Call Detection) feature.
Configuring Telephony Call-Redirect Features	Steps for configuring call transfer, Release-to-Pivot (RTPvt), and ISDN Two B-Channel Transfer (TBCT).
Configuring Tcl IVR 2.0 Session Interaction	Steps for starting instances of Tcl IVR 2.0 applications.
Configuring SIP and TEL URL Support	Steps for enabling voice applications to match incoming calls and place outbound calls to a Session Initiation Protocol (SIP) or telephone (TEL) URL.

Monitoring and Troubleshooting Voice Applications	Steps for using event logs and statistics introduced in the Voice Application Monitoring and Troubleshooting Enhancements feature to enable detailed monitoring of voice application instances and call legs.
	Support for MGCP scripting.

This chapter describes how to access Cisco Feature Navigator. It also lists and describes, by Cisco IOS release, Cisco IOS Tcl IVR and VoiceXML features for that release.

Note For information about the full set of Cisco IOS voice features, see the entire Cisco IOS Voice Configuration Library—including library preface, glossary, and other documents—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm

Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

Cisco IOS Tcl IVR and VoiceXML Feature List

Table 1-1 lists Cisco Tcl IVR and VoiceXML features by Cisco IOS release. Features that are introduced in a particular release are available in that and subsequent releases.

Note For releases earlier than Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_administration_guide_book09186a00804436fd.html

Table 1-1 Cisco IOS Tcl IVR and VoiceXML Features by Cisco IOS Release

Release	Features Introduced in That Release[1]	Feature Description	Feature Documentation
15.4(1)T	Enhanced VoiceXML	Provides support to prevent call loops.	Configuring VoiceXML with the SIP Phone
12.1(3)T	Tcl IVR Version 2.0	Tcl IVR Version 2.0 was introduced.	Configuring Basic Functionality for Tcl IVR and VoiceXML Applications
12.2(2)T	Enhanced Multi-Language Feature	Provides support for multiple languages for Tcl IVR.	Configuring Audio File Properties for Tcl IVR and VoiceXML Applications

Table 1-1 Cisco IOS Tcl IVR and VoiceXML Features by Cisco IOS Release (continued)

Release	Features Introduced in That Release[1]	Feature Description	Feature Documentation
12.2(2)XB	VoiceXML for Cisco IOS	Provides support for VoiceXML documents on Cisco IOS platforms.	Configuring Basic Functionality for Tcl IVR and VoiceXML Applications
	Fax Detection for VoiceXML	Provides support for fax detection on the Cisco IOS VoiceXML gateway to determine which calls are voice or fax.	Configuring Fax Detection for VoiceXML
	VoiceXML Record Element	Provides support for the VoiceXML 1.0 <record> element for recording speech to a choice of four destinations, including local memory on the Cisco gateway.	Configuring Audio File Properties for Tcl IVR and VoiceXML Applications
12.2(11)T	VoiceXML Voice Store and Forward	Allows streaming-based voice recording and playback features for various media.	Configuring VoiceXML Voice Store and Forward
	Speech Recognition and Synthesis for Voice Applications	Provides support for automatic speech recognition (ASR) and text-to-speech (TTS) capabilities for VoiceXML and Tcl applications on Cisco voice gateways.	Configuring ASR and TTS Properties
	VoiceXML Media Volume and Rate Controls	Provides controls for the rate and volume of prompts during playback by using Cisco attributes in the VoiceXML document.	Configuring Audio File Properties for Tcl IVR and VoiceXML Applications
12.3(1)	Voice Application Call Control Enhancements	Provides RTPvt, TBCT, and GTD support.	Configuring Telephony Call-Redirect Features
12.3(4)T	Tcl IVR 2.0 Session Interaction	Allows different instances of Tcl IVR applications (sessions) to communicate with other sessions on the same gateway and for applications to dynamically bridge call legs between different sessions	Configuring Tcl IVR 2.0 Session Interaction
	SIP and TEL URL Support	Enables Cisco gateways to direct incoming calls to a voice application based on the URL and Tcl IVR 2.0 and VoiceXML applications to place outbound calls to a Session Initiation Protocol (SIP) or telephone (TEL) URL.	Configuring Tcl IVR 2.0 Session Interaction

Table 1-1 Cisco IOS Tcl IVR and VoiceXML Features by Cisco IOS Release (continued)

Release	Features Introduced in That Release[1]	Feature Description	Feature Documentation
12.3(8)T	Voice Application Monitoring and Troubleshooting Enhancements	Enables detailed monitoring of voice application instances and call legs.	Monitoring and Troubleshooting Voice Applications
	Voice Application HTTP Client Cookie Support	Implements HTTP cookie support for Cisco IOS VoiceXML applications.	Configuring Basic Functionality for Tcl IVR and VoiceXML Applications
	ETSI Call Transfer	Provides support for European Telecommunications Standards Institute (ETSI) explicit call transfer functionality on Cisco IOS gateways.	ETSI Call Transfer
12.3(14)T	New Cisco Voice Application Command-Line Interface Structure	Implements a new command-line interface structure for configuring Tcl and IVR applications.	Cisco IOS Release 12.3(14)T and Later Releases Voice Application CLI Structure Changes
	New Cisco Voice Application Command-Line Interface Structure	Implements a new command-line interface structure for configuring Tcl and IVR applications.	Cisco IOS Release 12.3(14)T and Later Releases Voice Application CLI Structure Changes
	HTTP Client API for Tcl IVR	Provides support for Tcl IVR applications to retrieve data from or post data to an HTTP server.	*Tcl IVR API Version 2.0 Programmer's Guide*
	Support for Additional Tcl 8.3.4 Commands	Provides support for additional Tcl 8.3.4 commands.	*Tcl IVR API Version 2.0 Programmer's Guide*
12.4(4)T	Media Inactive Call Detection (Silent Call Detection)	Enhances Cisco IOS behavior for detection and monitoring calls before disconnecting when an inactive condition is detected.	Cisco IOS Release 12.3(14)T and Later Releases Voice Application CLI Structure Changes
12.4(11)T	VoiceXML 2.0	Provides support for the *VoiceXML Version 2.0* W3C Recommendation (March 16, 2004) on Cisco IOS voice gateway VoiceXML browsers.	Configuring Basic Functionality for Tcl IVR and VoiceXML Applications
12.4(15)T	VoiceXML 2.1	Provides support for the *VoiceXML Version 2.1* W3C Candidate Recommendation (June 13, 2005) on Cisco IOS voice gateway VoiceXML browsers.	Configuring Basic Functionality for Tcl IVR and VoiceXML Applications

1. Features that are introduced in a particular release are available in that and subsequent releases.

Overview of Cisco IOS Tcl IVR and VoiceXML Applications

Tcl and VoiceXML applications on the Cisco gateway provide Interactive Voice Response (IVR) features and call control functionality such as call forwarding, conference calling, and voice mail.

IVR systems provide information through the telephone in response to user input in the form of spoken words or dual tone multifrequency (DTMF) signaling. The Cisco voice gateway allows an IVR application to be used during call processing. A Cisco voice gateway can have several IVR applications to accommodate many different services, and you can customize the IVR applications to present different interfaces to various callers.

Voice applications can be developed using one or both of these two scripting languages:

- Tcl IVR 2.0—Tcl-based scripting with a proprietary Cisco API. Provides extensive call control capabilities, signaling, and GTD manipulation.
- VoiceXML—Standards-based markup language for voice browsers. Existing web server and application logic can be used for VoiceXML applications, requiring less time and money to build infrastructure and perform development than traditional proprietary IVR systems require.

For information on developing and implementing a VoiceXML document or Tcl script for use with your voice application, see the following guides:

- *Cisco VoiceXML Programmer's Guide*
- *Tcl IVR API Version 2.0 Programmer's Guide*

For more information about Cisco IOS voice features, see the entire Cisco IOS Voice Configuration Library—including library preface and glossary, feature documents, and troubleshooting information—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco IOS Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Contents

- Prerequisites for Implementing Tcl IVR and VoiceXML Applications, page 2
- Additional References, page 5

Prerequisites for Implementing Tcl IVR and VoiceXML Applications

The following sections describe the prerequisites necessary for configuring a Tcl IVR or VoiceXML application on the Cisco gateway:

- Memory Requirements for Interpreting a VoiceXML Document, page 2
- Gateway Prerequisite Configuration, page 2
- VCWare Version on Cisco AS5300, page 3
- Call Admission Control, page 3
- HTTP Server, page 4

Note When developing and configuring a voice application, also refer to the *Cisco VoiceXML Programmer's Guide* or the *Tcl IVR API Version 2.0 Programmer's Guide*.

Memory Requirements for Interpreting a VoiceXML Document

For the smallest document, the minimum memory that the Cisco IOS software and VoiceXML interpreter uses for a call is approximately 128 KB. The maximum memory allowed is approximately 380 KB. This is allocated as follows:

- The underlying system, including the telephony signaling software and JavaScript Expressions context, requires approximately 120 KB for a call.
- Each VoiceXML document can use a maximum of 65 KB of internal memory. The amount of memory that each document requires cannot be calculated by counting tags or lines in a VoiceXML document, but generally the memory required correlates to the size of the document.

Gateway Prerequisite Configuration

Step 1 Establish a working IP network using the Cisco gateway.

Step 2 Configure VoIP on the gateway, including voice ports and dial peers.

For information on configuring IP networking, VoIP, and ISDN, refer to the appropriate software configuration guide for your Cisco gateway:

- *Software Configuration Guide for Cisco 2600 Series, Cisco 3600 Series and Cisco 3700 Series Routers*
- Cisco 2800 Series Software Configuration
- Cisco 3800 Series Software Configuration
- *Cisco AS5300 Software Configuration Guide*
- *Cisco AS5350 and Cisco AS5400 Universal Gateway Software Configuration Guide*
- *Cisco AS5800 Operations, Administration, Maintenance, and Provisioning Guide*
- *Cisco AS5850 Operations, Administration, Maintenance, and Provisioning Guide*

Step 3 For Tcl IVR 2.0 applications, download any necessary certified Cisco Tcl scripts or audio files from the following location:

http://www.cisco.com/cgi-bin/tablebuild.pl/tclware

Step 4 For Voice Store and Forward, the mail application Tcl script, named *voicemail-offramp.tcl*, must be downloaded to your TFTP server. See the "Downloading the Tcl Script for Off-Ramp Mail Application" section on page 19 for instructions on how to download the mail application script.

Step 5 On Cisco 3600 series gateways, the codec type used for recordings must be compatible with the codecs supported by the **codec complexity** command on the gateway. The setting of the **codec complexity** command determines the supported codecs. See the "Modifying Codec Complexity on the Cisco 3600 Series" section on page 13 for instructions on how to change the codec complexity.

VCWare Version on Cisco AS5300

Cisco VCWare release required for the Cisco AS5300 VFC:

- Release 10.26a or later with DSPWare 4.0.26

Note You can verify your VCWare version by using the **show vfc** *slot* **version vcware** privileged EXEC command, where *slot* is the slot number (0 to 2) of the VFC card.

For more information on VCWare versions and installation:

- Overview of the VCWare version required for this feature: *Release Notes for Cisco VCWare on Cisco AS5300 Universal Access Servers/Voice Gateways*
- Descriptions of the different VCWare releases: *Cisco VCWare Compatibility Matrix for the Cisco AS5300 Universal Access Server/Voice Gateway*
- Download instructions: *Voice over IP for the Cisco AS5300*, "VFC Management" section.

Call Admission Control

Call admission control for calls handled by Tcl 1.0, Tcl IVR 2.0, and VoiceXML applications can be configured by setting the percentages of memory and CPU utilization that are optimal for the Cisco voice gateway and for the particular application scenario. When the percentage levels are set, incoming calls are denied whenever the current system CPU or memory usage (or a combination of these) exceeds the resource thresholds. This denial prevents the gateway from overloading.

The default is 89% for the CPU and 98% for RAM. However, on a high-end platform such as the Cisco AS5400, resource thresholds can be increased to accommodate more calls. To configure call admission control and set the optimal system CPU and RAM usage thresholds, use the **call treatment** and **call threshold** commands.

Tip The following is a recommended configuration for extreme performance conditions. The voice gateway rejects calls if the configured thresholds for CPU usage and memory usage for call treatment are exceeded.

```
Router(config)# call treatment on
Router(config)# call threshold global cpu-5sec low 50 high 80 treatment
```

```
Router(config)# call threshold global total-mem low 80 high 90 treatment
```

For detailed instructions on configuring call admission control, refer to the *Trunk Connections and Conditioning Features* document, Cisco IOS Voice Configuration Library, Release 12.3.

HTTP Server

The web server must support Hypertext Transfer Protocol (HTTP) 1.1. Cisco voice applications are tested for compatibility with web servers running Apache software; compatibility with other web servers is not verified. For instructions on installing Apache PHP software for use with Cisco voice applications, see the following procedures.

For information about the supported HTTP 1.1 features, see the "HTTP Client Support" section on page 11.

Installing Apache PHP Server Software

Step 1 Untar the php-4.0.4pl1.tar file.

Step 2 Untar the apache_1.3.17.tar file to a new directory.

Step 3 Modify the php-4.0.4pl1/sapi/apache/mod_php4.c file:

Change the line "if ((retval = setup_client_block(r, REQUEST_CHUNKED_ERROR)))" to "if ((retval = setup_client_block(r, REQUEST_CHUNKED_DECHUNK)))"

Step 4 cd to php-4.0.4pl1 directory.

Run configure, specifying the path to apache sources
(for example, /configure apache=/local/http/apache_1.3.17)

un "make" followed by "make install" from the preceding directory.

Step 5 cd to apache directory (for example, cd /local/http/apache_1.3.17).

Execute the following:

- configure prefix=/local/http/apache
- make
- make install

Step 6 Start your HTTP server in su mode:

/local/http/apache/bin/apachectl start

Enabling Chunked Transfer

When recording audio files to an HTTP server, audio data is moved in chunks from the gateway using the "chunked" Transfer-Encoding method. To accept streaming recording, the chunked transfer capability must be enabled on the web server. To enable chunked transfer on the Apache PHP server, perform the following steps:

Step 1 Turn on the following flag:

./php-4.0.3p11/sapi/apache/mod_php4.c

Step 2 Recompile the Apache PHP server code.

For more information, refer to the Apache Software Foundation website.

Additional References

The following sections provide references related to Cisco IOS Tcl IVR and VoiceXML.

Related Documents

Related Topic	Document Title
Cisco IOS	
Cisco IOS configuration examples	Cisco Systems Technology Support website at http://cisco.com/en/US/tech/index.html. Choose a technology category and subsequent hierarchy of subcategories. Under Configure, click **Configuration Examples and Tech Notes**.
Cisco IOS debug command reference	*Cisco IOS Debug Command Reference, Release 12.4T* at http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123tcr/123dbr/index.htm
Cisco IOS troubleshooting information	*Cisco IOS Voice Troubleshooting and Monitoring Guide* at http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/voipt_c/index.htm
Cisco IOS voice command reference	*Cisco IOS Voice Command Reference, Release 12.3T* at http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123tcr/123tvr/index.htm
Cisco IOS Voice Configuration Library preface and glossary	Cisco IOS Voice Configuration Library at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.
API guides	
VoiceXML elements and attributes	*Cisco VoiceXML Programmer's Guide*
Tcl verbs and attributes	*Tcl IVR Version 2.0 Programmer's Guide*
Hardware installation guides	
Cisco AS5300 hardware installation instructions	*Hardware installation documents for Cisco AS5300*
Cisco AS5350 hardware installation instructions	*Hardware installation documents for Cisco AS5350*
Cisco AS5400 hardware installation instructions	*Hardware installation documents for Cisco AS5400*
Software configuration guides	
Cisco 3600 series prerequisite software configuration	*Software Configuration Guide for Cisco 2600 Series, Cisco 3600 Series and Cisco 3700 Series Routers*
Cisco 2800 series prerequisite software configuration	*Cisco 2800 Series Software Configuration*
Cisco 3800 series prerequisite software configuration	*Cisco 3800 Series Software Configuration*

Related Topic	Document Title
Cisco AS5300 prerequisite software configuration	*Cisco AS5300 Software Configuration Guide*
VCWare releases	*Cisco VCWare Compatibility Matrix for the Cisco AS5300 Universal Access Server/Voice Gateway*
VCWare version requirements	*Release Notes for Cisco VCWare on Cisco AS5300 Universal Access Servers/Voice Gateways*
VCWare download instructions	*Voice over IP for the Cisco AS5300, "VFC Management" section.*
Cisco AS5350 prerequisite software configuration	*Cisco AS5350 and Cisco AS5400 Universal Gateway Software Configuration Guide*
Cisco AS5800 prerequisite software configuration	*Cisco AS5800 Operations, Administration, Maintenance, and Provisioning Guide*
Cisco AS5850 prerequisite software configuration	*Cisco AS5850 Operations, Administration, Maintenance, and Provisioning Guide*
Dial peer configuration	*Dial Peer Configuration*, Cisco IOS Voice Configuration Library, Release 12.3
H.323 gateway configuration	*Cisco IOS H.323 Configuration Guide*, Release 12.3
MGCP configuration	*Cisco IOS MGCP and Related Protocols Configuration Guide*, Release 12.3
SIP configuration	*Cisco IOS SIP Configuration Guide*, Release 12.3
Fax Relay and store and forward fax configuration	*Cisco IOS Fax Services over IP Application Guide*, Release 12.3
AAA configuration	"*Authentication, Authorization, and Accounting (AAA)*" chapter in *Cisco IOS Security Configuration Guide*, Release 12.3
VSAs used for call detail records (CDRs)	*RADIUS Vendor-Specific Attributes Voice Implementation Guide*
Codec support	*VoIP—Understanding Codecs: Complexity, Support, MOS, and Negotiation*
Call Transfer	
Call transfer and forward configuration	*Cisco IOS Call Transfer and Call Forwarding Supplementary Services*
GTD configuration	*GTD for GKTMP Using SS7 Interconnect for Voice Gatekeeper Version 2.0*
GTD configuration	*R2 and ISUP Transparency for Voice Gateways Version 2.0*
AAA configuration	*Cisco IOS Security Configuration Guide*, Release 12.3
Cisco SC2200 configuration	*Cisco SC2200 Signaling Controller* documentation
Miscellaneous	
Release notes	*Cisco IOS Release 12.3 Cross-Platform Release Notes*
Network module specifications for Cisco 3600 series	*Cisco Network Modules Hardware Installation Guide*

Related Websites

Related Topic	Title and Location
RTSP live streaming	Helix audio streaming server http://www.realnetworks.com
Speech recognition and synthesis software	Nuance Communications http://www.nuance.com
Speech recognition software	SpeechWorks International, Inc. http://www.speechworks.com
Voice browser activities' documentation	World Wide Web Consortium http://www.w3.org
Vovida RTSP server configuration	Vovida.org http://www.vovida.org
Web server software	Apache Software Foundation http://www.apache.org

Standards

Standards	Title
ANSI T1.661 SS7	*Release To Pivot*
ANSI T1.688 SS7	*Facility Request To Pivot*
ECMA-262	*ECMAScript Language Specification*, 3rd edition August 1998
ITU-T H.450.2	*Call transfer supplementary service for H.323*
ITU-T H.450.3	*Call diversion supplementary service for H.323*
Telcordia GR-2857-CORE	*Generic Requirements for SS7 Release to Pivot Network Capability*
Telcordia GR-2865-CORE	*Generic Requirements for ISDN PRI Two B-Channel Transfer*
Telcordia GR-3016-CORE	*Operator Services Generic Requirements for the Use of Signaling System 7 (SS7) Release to Pivot (RTP) Phase II Network Capability*
VoiceXML 2.1	W3C Candidate Recommendation (June 13, 2005)

MIBs

MIBs	MIBs Link
• CISCO-VOICE-DIAL-CONTROL-MIB • CISCO-VOICE-DNIS-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 821	*Simple Mail Transfer Protocol*
RFC 822	*Standard for the Format of ARPA Internet Text Messages*
RFC 1341	*MIME (Multipurpose Internet Mail Extensions)*, June 1992
RFC 1652	*SMTP Service Extension for 8bit-MIME Transport*
RFC 1869	*SMTP Service Extensions*
RFC 1891	*SMTP Service Extension for Delivery Status Notifications*
RFC 1892	*Multipart/Report Content Type for the Reporting of Mail System Administrative Messages*
RFC 1893	*Enhanced Mail System Status Codes*
RFC 1894	*An Extensible Message Format for Delivery Status Notifications*
RFC 1896	*The Text/Enriched MIME Content-Type*
RFC 2034	*SMTP Service Extension for Returning Enhanced Error Codes*
RFC 2045	*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*
RFC 2046	*Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*
RFC 2047	*MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*
RFC 2068	*Hypertext Transfer Protocol—HTTP/1.1*, January 1997
RFC 2109	*HTTP State Management Mechanism*
RFC 2197	*SMTP Service Extension for Command Pipelining*
RFC 2298	*An Extensible Message Format for Message Disposition Notifications*
RFC 2326	*Real-Time Streaming Protocol*, April 1998
RFC 2368	*The Mailto URL Scheme*, July 1998
RFC 2616	*Hypertext Transfer Protocol HTTP/1.1*, June 1999

Technical Assistance

Description	Link
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/public/support/tac/home.shtml
Developers using this guide may be interested in joining the Cisco Developer Support Program. This program was created to provide you with a consistent level of support that you can depend on while leveraging Cisco interfaces in your development projects.	http://www.cisco.com/en/US/products/svcs/ps3034/ps5408/ps5418/serv_home.html developer-support@cisco.com

Configuring Basic Functionality for Tcl IVR and VoiceXML Applications

This chapter explains the basic tasks required for loading and configuring a Tcl IVR or VoiceXML application on the Cisco gateway.

For more information about this and related Cisco IOS voice features, see the following:

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/products/ps6441/prod_configuration_guide09186a0080565f8a.html.

Note For releases earlier than Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco IOS Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_administration_guide_book09186a00804436fd.html

Feature History for Basic Functionality for Tcl IVR and VoiceXML Applications

This chapter includes basic Tcl IVR and VoiceXML application features. For a feature history of all Tcl IVR and VoiceXML features, see "Cisco IOS Tcl IVR and VoiceXML Feature List" on page 2.

Contents

- Prerequisites for Configuring Basic Functionality for Tcl IVR and VoiceXML Applications, page 2
- Restrictions for Configuring Basic Functionality for Tcl IVR and VoiceXML Applications, page 3
- Information About Configuring Basic Functionality for Tcl IVR and VoiceXML Applications, page 4
- How to Configure Basic Functionality for a Tcl IVR or VoiceXML Application, page 24
- Configuration Examples for Tcl IVR and VoiceXML Applications, page 71
- Where to Go Next, page 81
- Additional References, page 82

Prerequisites for Configuring Basic Functionality for Tcl IVR and VoiceXML Applications

- Recommended Knowledge, page 2
- VoiceXML Document Development, page 2

Recommended Knowledge

Before configuring a Tcl or VoiceXML application on the Cisco voice gateway, we recommend you have the following knowledge:

- For working with VoiceXML applications and writing VoiceXML documents:
  - Knowledge of web page development
  - Familiarity with the *VoiceXML 2.1* W3C Candidate Recommendation (June 13, 2005)
  - Knowledge of VoiceXML application programming
  - Familiarity with the *Cisco VoiceXML Programmer's Guide*
- For working with Tcl applications and writing Tcl IVR 2.0 scripts:
  - Familiarity with Tcl Version 7.1 or later
  - Knowledge of Tcl application programming
  - Familiarity with the *Tcl IVR API Version 2.0 Programmer's Guide*
- For setting up a web application environment:
  - Experience with web application administration
  - Knowledge of languages and protocols such as HTML and HTTP
- For configuring the Cisco voice gateway:
  - Experience with the prerequisite configuration of the Cisco voice gateway as described in the "Gateway Prerequisite Configuration" section on page 2
  - Familiarity with Cisco IVR and VoIP functionality

Tip For more resources, see the "Additional References" section on page 5.

VoiceXML Document Development

To define your voice application, you must write a VoiceXML document using a web-authoring tool. The document must be installed on a web or file server. A VoiceXML document can also call for the gateway to interact with various web applications (servlets and CGI executables), in which case you must also supply these web applications.

Note Place all VoiceXML documents behind a firewall.

Restrictions for Configuring Basic Functionality for Tcl IVR and VoiceXML Applications

The following restrictions apply to Tcl and VoiceXML applications:

- General Restrictions, page 3
- DTMF Relay Restrictions, page 4
- ISDN Overlap Restrictions, page 4

General Restrictions

- Not all *VoiceXML Version 2.1* features are supported, and there are modifications to features. See the *Cisco VoiceXML Programmer's Guide*.
- The web server must support HTTP 1.1. Cisco voice applications are tested for compatibility with web servers running Apache software; compatibility with other web servers is not verified.
- Tcl IVR 2.0 is not backward compatible with Tcl IVR 1.0. Tcl IVR 1.0 verbs and Tcl IVR 2.0 verbs cannot be mixed in a script.
- Incoming VoIP calls can only be transferred to POTS dial peers. Transfers between VoIP call legs are not supported.
- There is no DSP on the IP leg, so a voice application cannot initiate a tone.
- RTSP multicast sessions are not supported by the Cisco IOS RTSP client.
- HTTP must be used to dynamically load VoiceXML documents. Using FTP or TFTP to dynamically load documents may adversely impact gateway performance.
- W3C Semantic Markup Language (SML) is not supported. The media server providing TTS must use the vendor-specific markup language.
- A separate RTP stream for speech recognition is supported on the Cisco 3660 only when the codec complexity is set to high on the voice card. It is not supported for medium complexity codecs.
- These restrictions apply to speech recognition on an IP call leg:
  - The **codec** command must be set to G.711 u-law in the VoIP dial peer.
  - The **dtmf-relay** command must be set to rtp-nte if DTMF input is required.
  - The **no vad** command must be configured in the VoIP dial peer. The ASR server performs voice activity detection (VAD) so for accurate results, VAD should not be configured on the gateway.

Note Cisco VoiceXML features in Cisco IOS Release 12.2(11)T are based on the *W3C VoiceXML Version 2.0 Working Draft* (October 23, 2001). If you are running a release earlier than Cisco IOS Release 12.2(11)T and require information about the implementation of the VoiceXML Version 1.0 Specification, see the *Cisco VoiceXML Reference for VoiceXML 1.0*.

In Cisco IOS Release 12.4(15)T and later releases, VoiceXML 1.0 is not supported.

DTMF Relay Restrictions

To collect digits over an IP call leg, DTMF Relay must be configured and negotiated on the IP call leg by using the **dtmf-relay** command in the VoIP dial peer. The supported methods are:

- For H.323, use:
  - Cisco Proprietary RTP (cisco-rtp)
  - H245 Alphanumeric IE (h245-alphanumeric)
  - H245 Signal IE (h245-signal)
- For SIP, use RTP Named Telephone Event RFC 2833 (rtp-nte).

For more information about DTMF Relay, see the *Cisco IOS Voice Configuration Library*, Release 12.4.

ISDN Overlap Restrictions

Although Cisco routers can receive an ISDN call in en bloc or overlap modes, the Cisco VoiceXML application does not yet support overlap mode. When configured for en bloc mode, the setup message should contain all necessary addressing information to route the call. In overlap mode, the setup message does not contain the complete address. Additional information messages are required from the calling side to complete the called address. The Cisco VoiceXML application does not currently detect signals in the form of INFO messages; therefore, overlap mode fails.

As a workaround, in overlap mode, the dial-peer configuration cannot contain the port as a matching parameter. The following is an example:

```
dial-peer voice 101 pots
  application my_vxml
  incoming called-number .........
  direct-inward-dial
  port 1/0:15
  forward-digits 10
```

If the port is configured, the dial peer matches the port number and the call is routed to application my_vxml before the full DNIS is collected. By removing the configured port, the Telephony Service Provider layer handles the overlap and the application receives the full DNIS.

In the dial peer example above, the workaround is valid only if there are 10 digits in the overlap. After the dial peer matches the 10 digits and hands the call off to the Cisco VoiceXML application, additional digits in the form of INFO messages are not processed.

Information About Configuring Basic Functionality for Tcl IVR and VoiceXML Applications

To configure Tcl and VoiceXML applications, you must understand the following concepts:

- Benefits, page 5
- Feature Design of Voice Applications, page 6
- VoiceXML for Cisco IOS Feature Overview, page 6
- Tcl IVR 2.0 Overview, page 8

- MGCP Scripting Overview, page 9
- Call Handling Between Tcl and VoiceXML Applications, page 9
- HTTP Client Support, page 11
- HTTP over Secure Socket Layer, page 13
- Cisco IOS Release 12.3(14)T and Later Releases Voice Application CLI Structure Changes, page 13
- New Commands, page 17
- VoiceXML Before Version 2.0 and Version 2.0 Behavior, page 19
- VoiceXML 2.0 Changes, page 20

Benefits

- Enables integration between Tcl and VoiceXML and the development of hybrid applications.
- Provides TTS and ASR support through VoiceXML and a distributed server farm.
- Implements AAA authentication and authorization.
- Enables service providers to offer an enhanced unified communications service, in which a subscriber uses the same number for both voice and fax messages. The flexibility of the application allows personalized services to be configured for different customers or for different called numbers.
- Supports all standard telephony signaling: H.323, Media Gateway Control Protocol (MGCP), and Session Initiation Protocol (SIP).

Tcl IVR 2.0

- Extensive call control capabilities, signaling, and GTD manipulation.

VoiceXML

- The design of VoiceXML is based on the client/server model and therefore provides similar benefits. A web browser can request a VoiceXML document from an HTTP web server just as it requests an HTML web page. The ability to use existing HTTP web servers to generate "voice web" pages provides VoIP service providers and their customers with important benefits:
  - Existing web server and application logic can be used for VoiceXML applications, requiring service providers less time and money to build infrastructure and perform development than traditional proprietary IVR systems require.
  - Hosting of VoiceXML applications can be added to the services offered customers. Customers have an open, extensible method for customizing their voice applications.
  - Existing web development skills can be transferred to those developing voice applications. A large number of developers and system integrators with these skills are already available.
- Supports features described in the *VoiceXML 2.1* W3C Candidate Recommendation (June 13, 2005).

Note The gateway cannot support behavior defined in versions earlier than *W3C VoiceXML Version 2.0* and behavior defined in *W3C VoiceXML Version 2.0* or later for different calls simultaneously.

Feature Design of Voice Applications

Tcl and VoiceXML applications on the Cisco gateway provide Interactive Voice Response (IVR) features and call control capabilities such as call forwarding and voice mail.

IVR systems provide information over a telephone in response to user input in the form of spoken words or dual-tone multifrequency (DTMF) signaling. For example, when a user makes a call with a prepaid calling card or debit card, an IVR application prompts the caller to enter a specific type of information, such as an account number. After playing the voice prompt, the IVR application collects the predetermined number of touch tones (digit collection), forwards the collected digits to a server for storage and retrieval, and then places the call to the destination phone or system. Call records can be kept and a variety of accounting functions performed.

The Cisco voice gateway allows voice applications to be used during call processing. Typically, application scripts contain both executable files and audio files that interact with the system software. Tcl scripts and VoiceXML documents can be stored in any of the following locations: TFTP, FTP, or HTTP servers, Flash memory of the gateway, or on the removable disks of the Cisco 3600 series. The audio files that they reference can be stored in any of these locations, and on Real Time Streaming Protocol (RTSP) servers. A Cisco voice gateway can have several voice applications to accommodate many different services, and you can customize the voice applications to present different interfaces to the various callers. IP phones can also originate calls to a gateway running a voice application.

Voice applications on Cisco gateways can be developed using a choice of two scripting languages:

- Tcl IVR 2.0—Tcl-based scripting with a proprietary Cisco API.
- VoiceXML—Standards-based markup language for voice browsers.

Applications can also be developed using a hybrid of both Tcl IVR and VoiceXML. The following sections describe the basic features of Cisco IOS Tcl IVR and VoiceXML applications:

- VoiceXML for Cisco IOS Feature Overview, page 6
- Tcl IVR 2.0 Overview, page 8
- MGCP Scripting Overview, page 9

VoiceXML for Cisco IOS Feature Overview

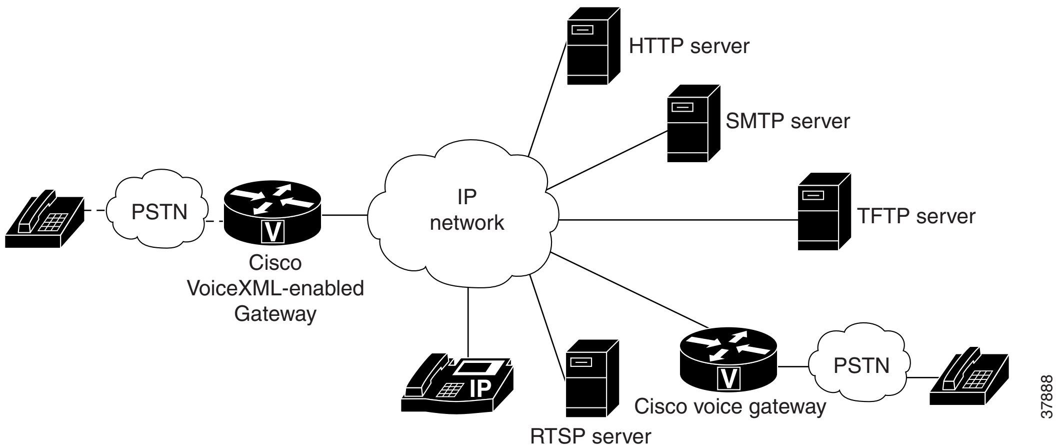
Applications written in Voice eXtensible Markup Language (VoiceXML) provide access through a voice browser to content and services over the telephone, just as Hypertext Markup Language (HTML) provides access through a web browser running on a PC. The universal accessibility of the telephone and its ease of use makes VoiceXML applications a powerful alternative to HTML for accessing the information and services of the World Wide Web.

The VoiceXML for Cisco IOS feature provides a platform for interpreting VoiceXML documents. When a telephone call is made to the Cisco VoiceXML-enabled gateway, VoiceXML documents are downloaded from the servers, providing content and services to the caller, typically in the form of prerecorded audio in an IVR application. Customers can access online business applications over the telephone, providing for example, stock quotes, sports scores, or bank balances.

VoiceXML brings the advantages of web-based development and content delivery to voice applications. It is similar to HTML in its simplicity and in its presentation of information. The VoiceXML for Cisco IOS feature is based on the *VoiceXML 2.1* W3C Candidate Recommendation (June 13, 2005) and is designed to provide web developers great flexibility and ease in implementing VoiceXML applications.

Figure 3-1 shows components that can be configured as a part of a VoiceXML application installed on a Cisco voice gateway:

Figure 3-1 VoiceXML for Cisco IOS Application Components



For information on developing a VoiceXML document for implementing an application on the Cisco voice gateway, see the *Cisco VoiceXML Programmer's Guide*.

VoiceXML Call Scenario Example

The following is an example call scenario for a VoiceXML application:

1. The caller dials a number and is connected through the PSTN or the IP network to a Cisco voice gateway that is configured as a VoiceXML-enabled gateway.

   For instance, the caller could be connected to a business providing sports scores over the telephone.

2. The Cisco voice gateway associates the dialed number with the appropriate VoiceXML document, residing on a web server.

   For example, this business uses a VoiceXML document on an HTTP server to provide sports scores.

3. The voice gateway runs the VoiceXML document and responds to the caller's input by playing the appropriate audio content.

   For instance, an application might play a prerecorded prompt that asks the caller to press a specific DTMF key ("Press 2 for the results of tonight's National League Playoff Game") to hear a spoken sport score ("Giants 4, Mets 0").

4. The VoiceXML application could also transfer the caller to another department such as customer service.

   For example, the application, after playing the score, might prompt the caller with the message: "If you sign up for a year's service now, you will be entered in the drawing for two tickets to this year's World Series. Press 5 to contact one of our agents."

VoiceXML Document Loop Security

The VoiceXML for Cisco IOS feature provides safeguards against denial of service attacks that use infinite looping VoiceXML documents.

A maximum of ten loops are permitted per VoiceXML session to help prevent disruption of the system by a malicious looping program. Loops are counted when a VoiceXML document transits to another dialog within a document or goes to another document using <submit> or <goto> without any user interaction. If a document goes to another dialog or another document ten times without any prompts or digit collection, the session is aborted.

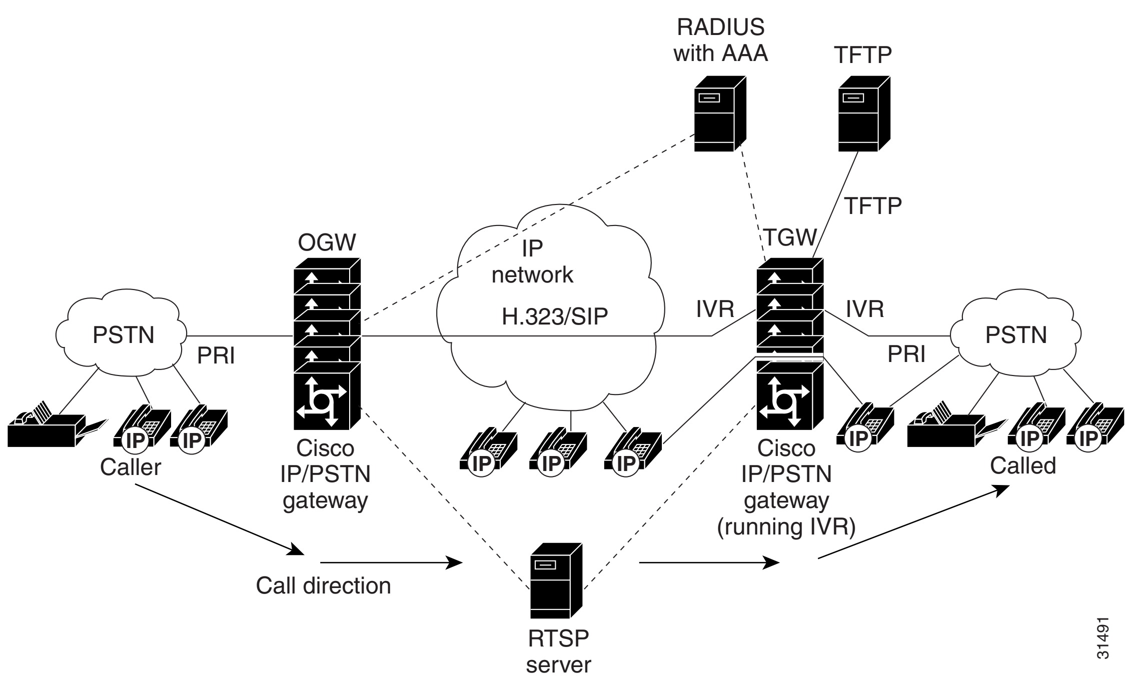
The loop count includes both before <disconnect> and after <disconnect> events. After <disconnect>, the VoiceXML document is mostly unrestricted if there is no user interaction.

Tcl IVR 2.0 Overview

Tcl IVR Version 2.0 uses Tcl scripts to gather data and to process accounting information. For example, a Tcl IVR script can play an audio prompt that asks callers to enter a specific type of information, such as a personal identification number (PIN). After playing the audio prompt, the Tcl IVR application collects the predetermined number of touch tones and sends the collected information to an external server for caller authentication and service authorization.

Figure 3-2 displays a Tcl IVR application on the gateway.

Figure 3-2 IVR Control of Tcl Scripts on an IP Call Leg



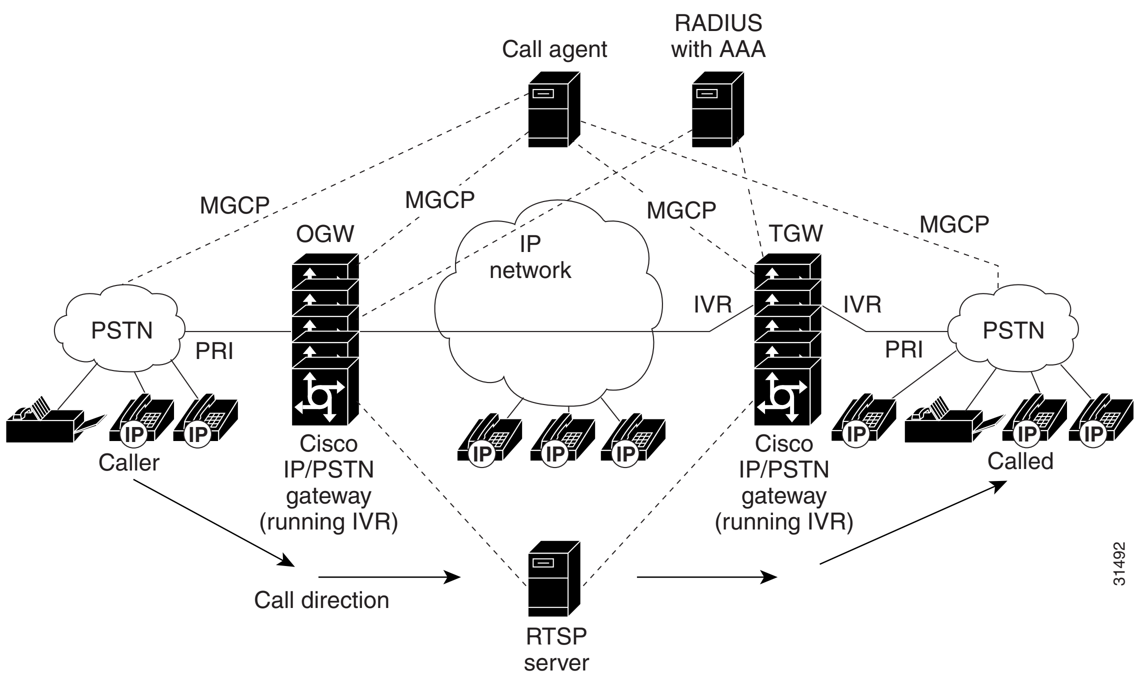
For information on developing Tcl scripts for voice applications, see the *Tcl IVR API Version 2.0 Programmer's Guide.*

MGCP Scripting Overview

MGCP scripting allows external call agents (CAs) to instruct the Cisco gateway to run a voice application on a PSTN or VoIP call leg. For example, you can request and collect the PIN and account number from a caller. These applications can be written in Tcl 2.0 or VoiceXML.

Figure 3-3 shows the MGCP CA controlling the application scripts.

Figure 3-3 MGCP Control of Voice Application Scripts



In Figure 3-3, the RTSP server is configured to interact with gateways that have voice applications installed and running. The RADIUS server also interacts with the gateways to provide authentication, authorization, and accounting (AAA).

For specific information about the VoiceXML implementation of MGCP, see "" on page 1. For instructions on how to configure your gateway for MGCP, see the *Cisco IOS MGCP and Related Protocols Guide*, Release 12.3.

Call Handling Between Tcl and VoiceXML Applications

Cisco voice gateways use special call-handling software applications. Some applications are contained in Cisco IOS software, others are defined dynamically by using the application configuration submodes. Applications that are defined dynamically can use Tcl scripts or VoiceXML documents.

When an application hands off calls to another application by placing a call through a dial peer to an outbound application, it performs a *transfer*. When a Tcl script hands off a call directly to a VoiceXML document or to another Tcl script, it performs a *handoff*. Transfer is performed through the dial plan by using Cisco IOS software, transferring a call to any number associated with that application. Typically, an inbound dial peer links to an application which may transfer calls through an outbound dial peer to an outbound application. Transfer supports a large database of number-to-URL mappings. In comparison, a handoff is done using Tcl. In a handoff, a Tcl 2.0 application can pass a handoff string. An application passing a handoff string is not supported by the transfer function.

To display a list of Tcl or VoiceXML applications that are currently configured or installed on the gateway, use the **show call application voice summary** command.

Following are some of the applications that come with Cisco IOS software:

- session—Similar to the DEFAULT application, except that it is written in Tcl 2.0. This is a basic application that performs the DID function or supplies a secondary dial tone to the caller.
- fax_hop_on—Collects digits from the redialer, such as account number and destination number. When a call is placed to an H.323 network, the set of fields (configured in the call information structure) are “entered,” “destination,” and “account.”
- clid_authen—Authenticates the call with automatic number identification (ANI) and DNIS numbers, collects the destination data, and makes the call.
- clid_authen_collect—Authenticates an incoming call using ANI or DNIS information, or, if that fails, collects dialed digits.
- clid_authen_npw—Performs as clid_authen, but uses a null password when authenticating, rather than DNIS numbers.
- clid_authen_col_npw—Performs as clid_authen_collect, but uses a null password and does not use or collect DNIS numbers.
- clid_col_npw_3—Performs as clid_authen_col_npw except with that script, if authentication with the digits collected (account and PIN) fails, the clid_authen_col_npw script just plays a failure message (auth_failed.au) and then hangs up. The clid_col_npw_3 script allows two failures, then plays the retry audio file (auth_retry.au) and collects the account and PIN again.

  The caller can interrupt the message by entering digits for the account number, triggering the prompt to tell the caller to enter the PIN. If authentication fails the third time, the script plays the audio file auth_fail_final.au, and hangs up.
- Default (DEFAULT)—This simple application outputs dial tone when a call comes in, collects digits, and places a call to the dialed number. Similar to the *session* application, except that it is included in Cisco IOS software.

For a complete list of Tcl scripts that can be downloaded from Cisco.com, check the following location:

http://www.cisco.com/cgi-bin/tablebuild.pl/tclware

You must have a Cisco.com login to access the above site. Qualified users can establish an account on Cisco.com by following the directions at http://tools.cisco.com/RPF/register/register.do. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you.

Transfer and Handoff Restrictions

- Tcl 2.0 applications can transfer (and handoff) calls to VoiceXML applications and other Tcl applications.
- VoiceXML cannot directly hand off calls to Tcl 2.0 applications because handoff is a Tcl scripting function and there is no equivalent in VoiceXML. VoiceXML can transfer calls to Tcl 2.0 by using Cisco IOS software, as described in the “Configuring an Outbound Application” section on page 52.
- When Tcl 2.0 or VoiceXML applications transfer calls to a telephone number, the outbound dial peer can specify a Tcl 2.0 or VoiceXML application to accept the transfer.
- Transfer (and handoff) to or from Tcl 1.0 applications is not supported.

- Transfer (and handoff) to or from the default application is not supported.

For more information on VoiceXML and Tcl coding, see the *Cisco VoiceXML Programmer's Guide* and the *Tcl IVR API Version 2.0 Programmer's Guide*, respectively.

HTTP Client Support

In general, HTTP is the preferred protocol for loading VoiceXML applications and audio prompts. The HTTP client code is implemented in Cisco IOS software specifically for this purpose. The Cisco File System (IFS) protocols (FTP, TFTP) were implemented for loading images, and saving and restoring configurations, so there are limits to the efficiency and number of concurrent loads. HTTP was developed for efficiency over the web; it has mechanisms to determine how long a file is considered valid in cache, and to determine if a cached version is still valid. With TFTP, the only way to determine if a cached version is valid is by reloading the entire file.

Pages that are loaded through a pointer within a document using TFTP are not cached on the gateway, and TFTP should not be used for loading these dynamic documents. For example, the application attribute of the <vxml> tag and the next attribute in the <goto> tag should not use IFS protocols in the URI. These documents should use HTTP.

Note Place all VoiceXML documents behind a firewall.

Table 3-1 lists the HTTP 1.1 client features supported by the Cisco gateway.

Table 3-1 HTTP 1.1 Feature Support

Feature	Supported?	Description
HTTP 1.1 client	Yes	HTTP 1.1 client functionality as required by the VoiceXML Forum's 1.0 Specification (refer also to RFC 2616—HTTP 1.1, June 1999).
HTTP and TFTP protocols for web server interchange	Yes	VoiceXML uses HTTP or TFTP protocols to interact with the web server. These are text-based protocols and exchanges are not encrypted.
HTTP caching	Yes	HTTP caching is supported.
HTTP cookies	Yes	HTTP cookies are supported by the HTTP 1.1 client in Cisco IOS Release 12.3(8)T and later releases.
HTTP proxy	No	There is no mechanism to redirect requests to an HTTP proxy on behalf of users. All HTTP request messages are sent directly to the server specified in the request URI.
HTTP/S	No	HTTP/S is not supported by the HTTP 1.1 client.
Secure shared use of HTTP client by multiple callers	Yes	Secure shared use of the HTTP client by multiple callers through caching of shared documents, not including documents generated as a result of an individual submit from caller input.

Cisco IOS software comes with a default set of HTTP 1.1 client parameters for file caching and connection timeouts. We recommend the default settings. To modify the default HTTP client settings, see the "Modifying HTTP Client Settings" section on page 66.

Supported HTTP 1.1 Headers

The Cisco HTTP 1.1 client supports the following formats for message headers:

Request Header

```
Accept: text/vxml, text/x-vxml, application/vxml, application/x-vxml,
application/voicexml, application/x-voicexml, application/octet-stream, text/plain,
text/html, audio/basic, audio/wav
Connection: closed/Keep-Alive
Content-Length:
Content-Type:
Host:
If-Modified-Since:
If-NoneMatch:
Transfer-Encoding:
User-Agent:
User-Agent: Cisco-IOS-family/Version Sub-Product-Name
User-Agent: Cisco-IOS-C5300/12.2(20011107:234726) VoiceXML/1.0
```

The following example shows a GET request message sent to the server tennis.cisco.com for the request URL: http://tennis.cisco.com/vxml/test/init.vxml.

```
GET /vxml/test/init.vxml HTTP/1.1
Host: tennis.cisco.com
Content-Type: application/x-www-form-urlencoded
Connection: Keep-Alive
Accept: text/vxml; level = 1, text/plain, text/html, audio/basic
User-Agent: Cisco-IOS-C5300/12.2(20011107:234726) VoiceXML/1.0
```

Response Header

```
Age:
Cache-Control:
Connection:
Content-Length:
Content-Type:
Date:
ETag:
Expires:
Keep-Alive: timeout=5, max=10
Last-Modified:
Location:
Pragma:
Transfer-Encoding:
```

Caching Refresh Value

The freshness lifetime of entries stored in the HTTP client cache is determined by one of the following values, in the order listed:

1. max_age_value
2. expire_value less the date_value
3. If the above information is not available, the gateway uses one of the following:
   a. If the last_modified_value is available, the freshness_lifetime is ten percent of the difference between the date_value and the last_modified_value.
   b. Otherwise, the freshness_lifetime is the value configured on the gateway by using the **http client cache refresh** command. If this command is not configured, the default value set by the gateway is 86,400 sec (24 hr).

HTTP over Secure Socket Layer

HTTP over Secure Socket Layer (SSL) (HTTPS) provides the capability to connect to the Cisco IOS HTTP server securely. HTTPS uses SSL to provide device authentication and data encryption. HTTPS is a secure message-oriented communications protocol designed for use with HTTP.

The HTTPS feature allows you to load VoiceXML application documents and scripts and play audio prompt files from the HTTP server using a secure socket interface. The HTTPS encryption enables you to securely store documents, such as recorded audio prompts, onto the HTTP server without the transmitted documents being exposed to public viewing or listening.

Cisco IOS Release 12.3(14)T and Later Releases Voice Application CLI Structure Changes

The **call application voice** command structure for configuring Tcl and IVR applications has been restructured for Cisco IOS Release 12.3(14)T and later to provide easier configuration of application parameters than the earlier CLI structure.

Note For releases earlier than Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco IOS Tcl IVR and VoiceXML Application Guide* at:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

The new configuration structure introduces the application configuration mode, which provides the following configuration submodes for parameters:

- Service parameter configuration submode: Use the **service** command in application configuration mode to load and configure a standalone application, such as a debit card script.
- Package parameter configuration submode: Use the **package** command in application configuration mode to load and configure a package. A package is a linkable set of C or Tcl functions that provide functionality invoked by applications or other packages. They are not standalone. For example, a debit card application may use multiple language translation packages, such as English and French. These language translation packages can also be used by other applications without having to modify the package for each application using it.
- Dial-peer parameter configuration submode: Use the **dial-peer** command in configuration mode to configure parameters for the services configured under this particular dial peer or packages, which such services use. Parameters under the dial-peer submode can only be configured using the **paramspace** command (described later).
- Group parameter configuration submode: Use the **group-params** command in application configuration mode to configure parameter groups. These groups of parameters can then be used by a service or by another package. Parameters under the group submode can only be configured using the **paramspace** command (described later).

Service and Package Parameters

Each service or package contains its own parameters that are available for configuration. Parameters can be configured in the following ways:

- Package—Parameters are configured for the package that registered the parameter

- Service—Parameters are configured for a service
- Dial-peer—Parameters are configured, and can be used by a service, which is configured under this dial peer or a package, which such services use
- Group—A group of parameters is configured, and can be used by a service or another package

Parameter Precedence

In cases where the same parameter is configured in more than one way, parameters configured at the dial-peer level override parameters configured at the services level, which override parameters configured at the package level.

For example, if the gateway receives a call and the script tries to access a certain parameter, the dial peer is first checked for the individual parameter definition or group parameter definition. If the parameter is configured at the dial-peer level, that value is used. If it is not configured there, the service level is checked and used if configured. If the parameter is not configured at the service level, the package level is checked and used if configured (applies to packages only).

Parameter Namespace or Parameterspace

To avoid problems with applications or packages using the same parameter names, the *parameter namespace,* or *parameterspace* concept is introduced. When a service or a package is defined on the gateway, its parameter namespace is automatically defined. This is known as the service or package's local parameterspace, or "myparameterspace." The name of that local parameterspace is the same as the name of a service or package as configured. When you use the **param** command to configure a service or package's parameters, the parameters available for configuration are those contained in the local parameterspace.

If you want to configure a parameter registered under a parameter space other than a local one or when the local parameter space is undefined (under param-group or dial-peer configuration submodes) use the **paramspace** command. This allows parameters registered under, for example, a package's paramspace to be configured on the configuration levels other than package: service and dial-peer.

Parameter Namespace Mapping

If you want to use parameter definitions found in a different parameterspace, you can use the use **paramspace parameter-namespace** command to map the package's parameters to a different parameterspace. This allows that package to use the parameter definitions found in the new parameterspace. Its original local parameterspace gets mapped or substituted by a new parameterspace, and its original local parameterspace is no longer available. See the example in "Using Parameterspaces" section on page 33 for more information.

Parameter Groups

The **group-params** command allows you to define groups of parameters so that a group of parameters can be used by multiple services or packages. Parameter groups are defined globally and once a group is defined, it is available for all services or packages to use.

Local parameterspace is undefined under a group configuration submode, therefore all parameters must be configured with explicit specification of parameter space this parameter belongs to by using the **paramspace** command. Groups can contain definitions of parameters under different parameterspaces.

The definition of the group is global (its definition does not belong to any specific service or package). However once defined, a group can be configured (used) at any of the three levels: service, package, or dial-peer, using the **group-name** command.

In cases where a parameter is configured individually and in a parameter group, the individual parameter definition is given precedence.

Upgrading to Cisco IOS Release 12.3(14)T

When a router is upgraded to Cisco IOS Release 12.3(14)T, automatic conversion of the entire application configuration from the old **call application voice** command structure to the new application configuration structure occurs. You do not have to make any configuration changes when upgrading.

Obsolete, Modified, and Replaced Commands

The existing **call application voice** and related commands have been moved to the application configuration mode. This creates a new configuration mode just for applications, and allows multiple parameters to be configured with fewer commands. Some commands are replaced, modified, or obsolete. Table 3-2 lists these commands.

Note The **show call application voice** command is not changed.

For more information on individual commands, see the *Cisco IOS Voice Command Reference, Release 12.4T* at http://cisco.com/en/US/products/ps6441/products_command_reference_book09186a00804973c0.html

Table 3-2 Replaced Call Application Commands

Command in Cisco IOS Release 12.3(11)T	Command Cisco IOS Release 12.3(14)T	
	Command	**Mode**
call accounting-template voice	**call accounting-template**	Global configuration and application configuration
call application alternate	**service**	Global application configuration
call application cache reload time	**cache reload time**	Application configuration global
call application event-log	**event-log**	Application configuration monitor
call application event-log dump ftp	**event-log dump ftp**	Application configuration monitor
call application event-log error-only	**event-log error-only**	Application configuration monitor
call application event-log max-buffer-size	**event-log max-buffer-size**	Application configuration monitor
call application global	**global**	Application configuration
call application history session event-log save-exception-only	**history session event-log save-exception-only**	Application configuration monitor
call application history session max-records	**history session max-records**	Application configuration monitor
call application history session retain-timer	**history session retain-timer**	Application configuration monitor
call application interface event-log	**interface event-log**	Application configuration monitor
call application interface event-log dump ftp	**interface event-log dump ftp**	Application configuration monitor

Table 3-2 Replaced Call Application Commands (continued)

Command in Cisco IOS Release 12.3(11)T	Command Cisco IOS Release 12.3(14)T	
	Command	**Mode**
call application interface event-log error-only	**interface event-log error only**	Application configuration monitor
call application interface event-log max-buffer-size	**interface event-log max-buffer-size**	Application configuration monitor
call application interface max-server-records	**interface max-server-records**	Application configuration monitor
call application interface stats	**interface stats**	Application configuration monitor
call application session start (global configuration)	**session start**	Application configuration
call application stats	**stats**	Application configuration monitor
call application voice	**application** **service** **package** **param**	Global configuration Application configuration Application configuration Application configuration
call application voice access-method	**param access-method**	Application configuration
call application voice account-id-method	**param account-id-method**	Application configuration
call application voice accounting enable	param accounting-enable	Application configuration
call application voice accounting-list	param accounting-list	Application configuration
call application voice accounting-template	call accounting-template	Global configuration or application configuration
call application voice authen-list	**param authen-list**	Application configuration
call application voice authen-method	param authen-method	Application configuration
call application voice authentication enable	**param authentication enable**	Application configuration
call application voice default disc-prog-ind-at-connect	**param convert-discpi-after-connect** or **paramspace session_xwork convert-discpi-after-connect**	Application configuration
call application voice dsn-script	**param dsn-script**	Application configuration
call application voice event-log	**param event-log** or **paramspace appcommon event-log**	Application configuration
call application voice fax-dtmf	**param fax-dtmf**	Application configuration
call application voice global-password	**param global-password**	Application configuration
call application voice language	**paramspace language location**	Application configuration

Table 3-2 Replaced Call Application Commands (continued)

Command in Cisco IOS Release 12.3(11)T	Command Cisco IOS Release 12.3(14)T	
	Command	**Mode**
call application voice mail-script	**param mail-script**	Application configuration
call application voice mode	**param mode**	Application configuration
call application voice pin-len	**param pin-len**	Application configuration
call application voice prompt	**param prompt**	Application configuration
call application voice redirect-number	param redirect-number	Application configuration
call application voice retry-count	**param retry-count**	Application configuration
call application voice security trusted	**param security** or **paramspace appcommon security**	Application parameter configuration
call application voice set-location	**paramspace language location**	Application configuration
call application voice transfer mode	**paramspace callsetup mode** or **param mode**	Application configuration
call application voice transfer reroute-mode	**paramspace callsetup reroutemode** or **param reroutemode**	Application configuration
call application voice uid-length	param uid-len	Application configuration
call application voice voice-dtmf	**param voice-dtmf**	Application configuration
call application voice warning-time	**param warning-time**	Application configuration
call language voice	**param language**	Application configuration

New Commands

Table 3-3 lists new commands and modes in Cisco IOS Release 12.3(14)T.

Table 3-3 New Commands and Modes in Cisco IOS Release 12.3(14)T

Command	Mode
application (global)	Global configuration
cache reload time	Global application configuration
event-log	Application configuration monitor
event-log dump ftp	Application configuration monitor
event-log error-only	Application configuration monitor
event-log max-buffer-size	Application configuration monitor
global (application configuration)	Application configuration
group-params	Application configuration

Table 3-3 New Commands and Modes in Cisco IOS Release 12.3(14)T (continued)

Command	Mode
history session event-log save-exception-only	Application configuration monitor
history session max-records	Application configuration monitor
history session retain-timer	Application configuration monitor
interface event-log	Application configuration monitor
interface event-log dump ftp	Application configuration monitor
interface event-log error only	Application configuration monitor
interface event-log max-buffer-size	Application configuration monitor
interface max-server-records	Application configuration monitor
interface stats	Application configuration monitor
package	Application configuration
package appcommon	Application configuration
package callsetup	Application configuration
package language	Application configuration
package session_xwork	Application configuration
param	Application configuration
param access-method	Application configuration
param account-id-method	Application configuration
param accounting enable	Application configuration
param accounting-list	Application configuration
param authen-list	Application configuration
param authen-method	Application configuration
param authentication enable	Application configuration
param convert-discpi-after-connect	Application configuration
param dsn-script	Application configuration
param event-log	Application configuration
param fax-dtmf	Application configuration
param global-password	Application configuration
param language	Application configuration
param mail-script	Application configuration
param mode	Application configuration
param pin-len	Application configuration
param prompt	Application configuration
param redirect-number	Application configuration
param reroutemode	Application configuration
param retry-count	Application configuration
param security	Application configuration

Table 3-3 New Commands and Modes in Cisco IOS Release 12.3(14)T (continued)

Command	Mode
param uid-len	Application configuration
param voice-dtmf	Application configuration
param warning-time	Application configuration
paramspace appcommon security	Application configuration
paramspace appcommon security	Application configuration
paramspace callsetup mode	Application configuration
paramspace callsetup reroutemode	Application configuration
paramspace language location	Application configuration
paramspace session_xwork convert-discpi-after-connect	Application configuration
service	Application configuration
session start	Application configuration
stats	Application configuration monitor

Table 3-4 lists new commands in Cisco IOS Release 12.4(15)T.

Table 3-4 New Commands in Cisco IOS Release 12.4(15)T

Command	Mode
debug mrcp	Privileged EXEC
http client secure-ciphersuite	Global configuration
http client secure-trustpoint	Global configuration
show call active media	User EXEC and privileged EXEC
show call history media	User EXEC and privileged EXEC
show http client secure status	User EXEC and privileged EXEC

For more information on individual commands, see the *Cisco IOS Voice Command Reference, Release 12.4T* at
http://www.cisco.com/en/US/products/ps6441/products_command_reference_book09186a00804973c0.html

VoiceXML Before Version 2.0 and Version 2.0 Behavior

The default VoiceXML behavior is compatible with versions earlier than *W3C VoiceXML 2.0*. You can enable *W3C VoiceXML 2.0* behavior by entering the **vxml version 2.0** command in global configuration mode. This command enables the following features:

- An audio error event, error.badfetch, is not thrown when an audio file cannot be played, for instance, because the file is in an unsupported format, the src attribute references an invalid URI, or the expr attribute evaluates to an invalid URI.
- Support for the beep attribute of the <record> element.

- Blind transfer compliant with *W3C VoiceXML 2.0* and not the same as consultation transfer.
- A semantic error is generated if an undeclared variable is used. You must declare variables before using them.

Use the **no vxml version 2.0** command to revert to the default behavior.

You can turn on the audio error feature only, which is compatible with versions earlier than *W3C VoiceXML 2.0*, by entering the **vxml audioerror** command in global configuration mode. Use the **no vxml audioerror** command to disable this feature. The **vxml audioerror** command overrides the **vxml version 2.0** command, so that if both commands are entered, the audio error event will be thrown when an audio file cannot be played.

VoiceXML 2.0 Changes

The following sections describe the changes made in VoiceXML Version 2.0:

- Obsolete Elements, page 20
- New Elements, page 20
- New Attributes, page 20
- New Shadow Variables, page 21
- New Properties, page 21
- New Session Variables, page 21
- Error Events, page 21
- Behavior Changes, page 23
- New Functionality, page 24
- New Restrictions, page 24

Obsolete Elements

The <cisco-puts> and <cisco-putvar> elements are obsolete. Use the <log> and <value expr> elements for logging and debugging.

New Elements

Support was added for the <metadata> element as a means of specifying metadata information using a schema.

New Attributes

- Support was added for the <vxml> element for these attributes: xml:base, xmlns, xmlns:xsi, and xsi:schemaLocation.

Note Both base and xml:base are supported for backward compatibility.

- Support was added for the xml:base attribute of the <prompt> element.
- The transferaudio attribute was added to the <transfer> element.

- The aai and aaiexpr attributes were added to allow data passing with the <transfer> element.
- Support was added for the beep attribute of the <record> element.
- The bargeintype attribute was added to the <prompt> element with values "speech" and "hotword".
- Support was added for the accept attribute with the value "aaproximately" in the <choice> and <option> elements.

New Shadow Variables

- The maxtime shadow variable was added to the <record> element.
- The <transfer> utterance shadow variable is set to the DTMF result, if the transfer was terminated by DTMF input.
- The shadow variable name$.inputmode was added as DTMF when a transfer is terminated by long pound.

New Properties

- The bargeintype property was added.
- The fetchaudiodelay and fetchaudiominimum properties were added to fetch properties.

New Session Variables

The following session variables were added:

- session.connection.protocol.name
- session.connection.protocol.version
- session.connection.redirect
- session.connection.originator
- session.connection.local.uri
- session.connection.remote.uri

Error Events

The following sections describe changes in error events thrown.

- Different Error Events Thrown, page 21
- New Error Events Thrown, page 22
- Error Events No Longer Thrown, page 23

Different Error Events Thrown

In these cases, a different error is thrown in VoiceXML 2.0 than was thrown in versions before VoiceXML 2.0:

- The error.unsupported.builtin event is thrown if an unsupported builtin type is detected.

  In versions before VoiceXML 2.0, the error.badfetch event was thrown.

- The error.badfetch event is thrown if the nextitem or expitem attributes in the <goto> element result in a nonexistent form item.

  In versions before VoiceXML 2.0, the error.semantic event was thrown.
- The error.badfetch event is thrown if the next or expr attributes evaluate to an incorrect uniform resource identifier (URI).

  In versions before VoiceXML 2.0, the error.semantic event was thrown.
- The errror.connection.baddestination event is thrown if an invalid URI is found in the <transfer> element.

  In versions before VoiceXML 2.0, the error.badfetch event was thrown.
- The error.semantic is thrown if neither the dest nor the destexpr attribute is specified in the <transfer> element.

  In versions before VoiceXML 2.0, the error.badfetch event was thrown.
- The error.semantic error is thrown for an illegal property value.

  In versions before VoiceXML 2.0, the error.badfetch was thrown for some illegal properties.
- If no audio input or output resource is available, the error.noresource event is thrown.

New Error Events Thrown

In these cases, an error is thrown in VoiceXML 2.0, whereas no error was thrown in versions before VoiceXML 2.0

- The error.badfectch event is thrown if the <filled> element inside the <field> element specified a mode.
- The error.badfetch event is thrown if the namelist attribute of the <filled> element references a control item variable.
- The error.badfetch event is thrown if a <menu> element's dtmf attribute is set to true and a <choice> element contained a choice other than 0 ,* or #.
- The error.semantic event is thrown if any variable in the namelist attribute of the <clear> element is undeclared.
- The error.badfetch event is thrown if both inline and external script is specified for the same element.
- The error.badfetch event is thrown if neither inline nor external script is specified.
- The error.badfetch event is thrown if both inline and external grammar is specified for the same element.
- The error.badfetch event is thrown if neither inline nor external grammar is specified for ASR grammar.

Note Empty grammar is allowed only for regex grammar.

- The error.badfetch event is thrown if the catch event has an empty string.
- The error.semantic event is thrown if variable names, including field names, contain a dot. For example, the field name "a.b" is illegal.
- The error.unsupportedformat event is thrown when an invalid grammar media type is used.
- Unsupported synthesis language results in an error.unsupported.language event being thrown.
- The error.noauthorization event is thrown if the caller has insufficient permission to make a transfer.

Error Events No Longer Thrown

In these cases, no error is thrown in VoiceXML 2.0, whereas an error was thrown in versions before VoiceXML 2.0.

- Branching within the document is allowed in the <submit> element. The document is reloaded even if only a fragment URI is specified.

  In versions before VoiceXML 2.0, the error.semantic event was thrown.

- The content attribute in the <meta> element is no longer a mandatory attribute.

  In versions before VoiceXML 2.0, the error.badfetch event was thrown.

- If the <menu> element's dtmf attribute is set, and if the <choice> element specified a number greater than 9, then no error is thrown. However, any key press greater than 9 will not be matched.

  In versions before VoiceXML 2.0, the error.semantic event was thrown.

Behavior Changes

- The <choice> event handler without control transition causes the <menu> element to be reexecuted.
- Errors which occur during form item transition in the <goto> element, are handled in the dialog scope.
- Speech or DTMF <grammar> elements in a <field> element with a specified built-in type are in addition to the built-in grammars; they do not override them.
- No variables, conditions or counters are reset when using the nextitem attribute of the <goto> element.
- The right catch handler is selected depending on the correct match count rather than selecting the handler in the most local scope.
- The value of the <initial> variable is initialized to boolean "true" instead of "defined".
- Reloading of scripts depends on the maxage and maxstale properties.
- When a <catch> element is executed, variables are resolved and thrown events are handled relative to the scope where the original event originated, not relative to the scope that contains the <catch> element.
- The link grammar inside the <initial> element was activated.
- Use of the <enumerate> element is allowed within the prompts and the catch elements associated with <menu> elements and with <field> elements that contain <option> elements.
- The namelist attribute of the <clear> element is allowed to specify variables other than form item variables which need to be reset.
- Whitespaces are accepted in DTMF sequences specified in the dtmf attributes in the <choice>, <option> and <link> elements.
- The xmlns attribute is added to inline grammars if it is not already specified.
- If the universals property is set to all, then exit and cancel grammars in addition to help will be generated by default.
- The form item variable of the <transfer> element is undefined for blind transfer.
- The platform is disconnected immediately when blind transfer takes place. The connection status is not available for blind transfer, although some error conditions may be reported.
- The transfer form item variable is undefined if connection.disconnect.transfer is thrown.

- If an implementation does not support a specific object, it throws an error.unsupported.objectname.
- In the <audio> element, if the audio file cannot be played and the content of the element is empty, no audio is played and no error event is thrown.
- When the expr attribute of the <audio> element evaluates to ECMAScript undefined, the content of the element ignored.
- When fetchaudio is specified, all the queued prompts are played out, then the next document is executed.

New Functionality

- Unsupported language is indicated in the message variable of a <throw> element.
- The value of the expr attribute of the <exit> element is an ECMAScript expression.
- Support was added for the <script> element with UTF-8 and UTF-16 charsets.
- Support was added for referencing grammars and scripts dynamically using the srcexpr attribute.
- Interpretation of grammars in the <record> element was added.
- Support was added for required audio formats for the <audio> and <record> elements.
- Support was added for hotword grammar in the <transfer> element.
- Scansoft property SWI_ in NLS is ignored.
- Support was added for submit recording using multipart and form-data without chunking.
- When an event is done, and there is no transition to other dialog, the control should go back to the Form Interpretation Algorithm (FIA). Instead, VoiceXML is terminated.

New Restrictions

- An error.semantic is thrown if the number of form items selected continuously without any asynchronous operations, such as media play, ASR, or TTS, exceeds 50.

How to Configure Basic Functionality for a Tcl IVR or VoiceXML Application

This section describes the basic procedures for loading a Tcl or VoiceXML application onto the Cisco gateway and assigning the application to a dial peer.

- Loading a Service onto the Gateway, page 25
- Verifying Loading of Service, page 26
- Defining a Package on the Gateway, page 28
- Verifying Package Definition, page 29
- Configuring Service Parameters, page 31
- Configuring Package Parameters, page 32
- Using Parameterspaces, page 33
- Verifying Parameterspace Configuration, page 36
- Defining Parameter Groups, page 36

- Verifying Parameter Group Definition, page 37
- Using Parameter Groups, page 38
- Verifying Parameter Group Configuration, page 39
- Configuring an Inbound Application, page 40
- Verifying an Inbound Application Configuration, page 44
- Verifying the Gateway Configuration by Using a Sample Application, page 45
- Configuring an Outbound Application, page 52
- Configuring an Outbound VoIP Dial Peer for Call Transfers, page 55
- Verifying the Outbound Application Configuration, page 58
- Configuring VoiceXML with the SIP Phone, page 60
- Configuring a DNIS Map for VoiceXML Applications, page 61
- Verifying DNIS Map Configuration, page 64
- Modifying HTTP Client Settings, page 66
- Configuring HTTPS, page 67
- Verifying HTTP Client Settings, page 69

Loading a Service onto the Gateway

This section describes how to load a VoiceXML document or Tcl script onto the Cisco gateway. A service is a standalone application. The application service name can then be configured under a dial-peer to provide services.

Tip If you download a Tcl script from the Cisco Software Center at http://www.cisco.com/public/sw-center/index.shtml, be sure to review the ReadMe file that is included with the script. It may contain additional script-specific information, such as configuration parameters and user interface descriptions. You must have an account on Cisco.com to access the Software Center. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. application
4. service *service-name location*
5. **end**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
    application
Example: Router(config)# application
```

Step 4 Load a VoiceXML document or Tcl script and define its application name:

```
    service service-name location
```

- *service-name*—Name that identifies the voice application. This is a user-defined name and does not have to match the script name.
- *location*—Directory and filename of the Tcl script or VoiceXML document in URL format. For example:
  - Flash memory (flash:*filename*)
  - TFTP server (tftp://../*filename*)
  - HTTP server (http://../*filename*)
  - HTTPS server (https://../*filename*)
  - built-in applications (builtin:*filename*)

```
Example: Router(config-app)# service fax_detect flash:app_fax_detect.2.1.2.2.tcl
```

Step 5 (Optional) Exit the submode and return to privileged EXEC mode.

```
    end
Example: Router(config-app)# end
```

Verifying Loading of Service

To verify the service was loaded, perform the following steps.

SUMMARY STEPS

1. **show running-config**
2. **show call application voice** *service-name*
3. **show call application voice summary**

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that the service is loaded onto the gateway:

```
!
 service fax_detect flash:app_fax_detect.2.1.2.2.tcl
!
```

Step 2 You can also use the **show call application voice** command to verify that the service is loaded onto the gateway. The following example shows output for the service named *fax_detect*:

```
Router# show call application voice fax_detect
Script Name : fax_detect
       URL  : flash:app_fax_detect.2.1.2.2.tcl
```

```
       Type : Service
       State: 3
       Life : Configured
       Calls: 0

  Parameters registered under fax_detect namespace:
  name                 type  default value   description
  pin-len              I     4               the number of digits in PIN
  retry-count          I     3               the number of attempts to reenter PIN
  redirect-number      S                     the telephone number where a call is
redirected to
  uid-len              I     10              the number of digits in UID
  warning-time         I     30              the time (in secs) within which a user is
warned before the calling time expires (call terminates)
Script Code Begin:
--------------------------------

Tcl Script  version 2.0 - 2.1
.
.
.
```

Step 3 Use the **show call application voice summary** command to view a list of all services and packages configured on the gateway:

```
Router# show call application voice summary
SERVICES (standalone applications):
  name                 description

  session              builtin:app_session_script.tcl
  fax_hop_on           builtin:app_fax_hop_on_script.tcl
  clid_authen          builtin:app_clid_authen_script.tcl
  clid_authen_collect  builtin:app_clid_authen_collect_script.tcl
  clid_authen_npw      builtin:app_clid_authen_npw_script.tcl
  clid_authen_col_npw  builtin:app_clid_authen_col_npw_script.tcl
  clid_col_npw_3       builtin:app_clid_col_npw_3_script.tcl
  clid_col_npw_npw     builtin:app_clid_col_npw_npw_script.tcl
  session_service      builtin:Session_Service.C
  DEFAULT              Default system session application
  lib_off_app          Libretto Offramp
  app_debitcard        flash:app_debitcard.2.0.2.4.tcl
  fax_detect           flash:app_fax_detect.2.1.2.2.tcl

 PACKAGES:
  name                 description

  tcl20base            builtin:tcl20base_package.C
  media                builtin:package_media.C
  vxmlbase             builtin:vxmlbase_package.C
  destination          builtin:Destination.C
  english              builtin:package_english.C
  httpios              builtin:package_httpios.C
  callsetup            builtin:CallSetup.C
  session_xwork        builtin:Session_XWork.C
  chinese              builtin:package_chinese.C
  consult              builtin:Consult.C
  tclmodule            builtin:TclModule.C
  simple               flash:simple.tcl
  consultresp          builtin:ConsultResp.C
  tclcore              builtin:tclcore_package.C
  pkg3                 flash:pkg3.tcl
  vxmlmodule           builtin:VxmlModule.C
  spanish              builtin:package_spanish.C
  digitcollect         builtin:DigitCollect.C
```

Tip If you have modified the VoiceXML document or Tcl script, you must perform the steps in "Loading a Service onto the Gateway" section on page 25 to load the new version of the script onto the gateway.

Troubleshooting Tips

If the service does not successfully load onto the gateway, Table 3-5 lists some possible causes and the actions that you can take.

Table 3-5 Service Does Not Load onto Gateway

Possible Causes	Suggested Actions
Cisco gateway cannot access the external server to download the associated VoiceXML document or Tcl script.	Ping the corresponding server to make sure that the gateway has connectivity.
Tcl script or VoiceXML document contains an error that prevents it from being loaded.	Use the **debug voip ivr** command and retry the steps in "Loading a Service onto the Gateway" section on page 25. The **debug voip ivr** command provides information about why the application could not be loaded. For example, this command displays an error message if the script contains bad syntax. See the *Cisco IOS Debug Command Reference, Release 12.4T* for more information on debug commands.

Defining a Package on the Gateway

This section describes how to define a package on the gateway. A package is a set of linkable C or Tcl functions that provide functionality invoked by applications or other packages. They are not standalone.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. application
4. package *package-name location*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
```

```
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
    application
Example: Router(config)# application
```

Step 4 Define a package:

package *package-name location*

- *package-name*—Name that identifies the package.
- *location*—Directory and filename of the package in URL format. For example, Flash memory (flash:*filename*), a TFTP (tftp://../*filename*) or an HTTP server (http://../*filename*) are valid locations.

```
Example: Router(config-app)# package language http://server-1/language_translate.tcl
```

Verifying Package Definition

To verify the package is defined, perform the following steps.

SUMMARY STEPS

1. **show running-config**
2. **show call application voice** *package-name*
3. **show call application voice summary**

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that the package is defined on the gateway:

```
!
application
!
 package language http://server-1/language_translate.tcl
!
```

Step 2 You can also use the **show call application voice** command to verify that the package is defined on the gateway. The following example shows output for the package named *callsetup*:

```
Router# show call application voice callsetup
Script Name : callsetup
       URL  : builtin:CallSetup.C
       Type : Package
       State: 3
       Life : Builtin
       Calls: 0

  Parameters registered under callsetup namespace:
  name                 type  default value   description
  mode                 S     rotary          functional mode of this package
  reroutemode          S     rotary          call reroute handling option
Script Code Begin:
--------------------------------

Built in C Package implementing the CallSetup functionality
--------------------------------
```

Step 3 Use the **show call application voice summary** command to view a list of all services and packages configured on the gateway:

```
Router# show call application voice summary
SERVICES (standalone applications):
  name                  description

  session               builtin:app_session_script.tcl
  fax_hop_on            builtin:app_fax_hop_on_script.tcl
  clid_authen           builtin:app_clid_authen_script.tcl
  clid_authen_collect   builtin:app_clid_authen_collect_script.tcl
  clid_authen_npw       builtin:app_clid_authen_npw_script.tcl
  clid_authen_col_npw   builtin:app_clid_authen_col_npw_script.tcl
  clid_col_npw_3        builtin:app_clid_col_npw_3_script.tcl
  clid_col_npw_npw      builtin:app_clid_col_npw_npw_script.tcl
  session_service       builtin:Session_Service.C
  DEFAULT               Default system session application
  lib_off_app           Libretto Offramp
  app_debitcard         flash:app_debitcard.2.0.2.4.tcl
  fax_detect            flash:app_fax_detect.2.1.2.2.tcl

 PACKAGES:
  name                  description

  tcl20base             builtin:tcl20base_package.C
  media                 builtin:package_media.C
  vxmlbase              builtin:vxmlbase_package.C
  destination           builtin:Destination.C
  english               builtin:package_english.C
  httpios               builtin:package_httpios.C
  callsetup             builtin:CallSetup.C
  session_xwork         builtin:Session_XWork.C
  chinese               builtin:package_chinese.C
  consult               builtin:Consult.C
  tclmodule             builtin:TclModule.C
  simple                flash:simple.tcl
  consultresp           builtin:ConsultResp.C
  tclcore               builtin:tclcore_package.C
  pkg3                  flash:pkg3.tcl
  vxmlmodule            builtin:VxmlModule.C
  spanish               builtin:package_spanish.C
  digitcollect          builtin:DigitCollect.C
```

Troubleshooting Tips

If the package does not successfully load onto the gateway, Table 3-6 lists some possible causes and the actions that you can take.

Table 3-6 Package Does Not Load onto Gateway

Possible Causes	Suggested Actions
Cisco gateway cannot access the external server to download the associated VoiceXML document or Tcl script.	Ping the corresponding server to make sure that the gateway has connectivity.
Tcl script or VoiceXML document contains an error that prevents it from being loaded.	Use the **debug voip ivr** command and retry the steps in "Loading a Service onto the Gateway" section on page 25. The **debug voip ivr** command provides information about why the application could not be loaded. For example, this command displays an error message if the script contains bad syntax.

Configuring Service Parameters

To configure service parameters, use the following steps. In this section, we assumed that you are configuring parameters registered under the service's local parameterspace. For example, if you are configuring the fax_detect service, you are configuring parameters registered under the fax_detect namespace. To see a list of parameters registered under the local parameterspace for the service, enter **param ?** in service parameter configuration mode.

To configure parameters using a different parameterspace, see the "Using Parameterspaces" section on page 33.

Prerequisites

The **param register** Tcl command in a service or package registers a parameter and provides a description and default values which allow the parameter to be configured using the CLI. The **param register** command is executed when the service or package is loaded or defined, along with commands such as **package provide**, which register the capability of the configured module and its associated scripts. You must configure and load the Tcl scripts for your service or package and load the package in order to configure its parameters. See the *Tcl IVR API Version 2.0 Programming Guide* for more information.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** *service-name location*
5. **param** *parameter-name value*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

Example: Router> enable

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

Example: Router# configure terminal

Step 3 Enter application configuration mode to configure applications and services:

application

Example: Router(config)# application

Step 4 Enter service parameter configuration mode:

service *service-name location*

Note When configuring a service that is defined, do not specify a location.

Example: Router(config-app)# service debitcard

Step 5 Configure the parameter's value:

param *parameter-name value*

```
Example:
Router(config-app-param)#param ?
  Parameters registered under debitcard namespace:
  name                 type  default value   description
  uid-len              I     10              the number of digits in UID
Router(config-app-param)#param uid-len 4
```

Configuring Package Parameters

To configure package parameters, use the following steps. In this section, we assume that you are configuring parameters registered under the package's local parameterspace. For example, if you are configuring the debit_card package, you are configuring parameters registered under the debit_card namespace.

Prerequisites

The **param register** Tcl command in a service or package registers a parameter and provides a description and default values which allow the parameter to be configured using the CLI. Use the **param register** command when the service or package is loaded or defined, along with commands such as **package provide**, which register the capability of the configured module and its associated scripts. See the *Tcl IVR API Version 2.0 Programming Guide* for more information.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**

4. **package** *package-name*
5. **param** *parameter-name value*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
application

Example: Router(config)# application
```

Step 4 Enter service parameter configuration mode:

package *package-name*

Note When configuring a package that is defined, do not specify a location.

```
Example: Router(config-app)# package httpios
```

Step 5 Configure the parameter's value:

param *parameter-name value*

```
Example:
Router(config-app-param)#param ?
  Parameters registered under callsetup namespace:
  name                 type  default value   description
  mode                 S     rotary          functional mode of this package
Router(config-app-param)#param mode rotary
```

Using Parameterspaces

When you configure parameters for a service or a package using the **param** command, if you do not specify a parameterspace, the parameters are assumed to belong to the local parameterspace of the service or package being configured. For example, if you are configuring the fax_detect service, you are configuring parameters registered under the fax_detect namespace. These parameters and values were registered and their original values defined when the service or package was loaded.

If you want to configure parameters defined under another parameterspace, you can use the **paramspace** *parameter-namespace parameter-value* command. For example, if you are configuring the "bator" service (you are in bator service configuration mode), but want to configure the mode parameter that is registered under a namespace of a callsetup package that this bator service is using, you would use the **paramspace callsetup mode rotary** command.

Note If you are configuring a parameter under a dial-peer or parameter group configuration submode, you **must** use the **paramspace** command, because the local parameterspace is undefined there.

Use the following steps to configure a parameter using a different (other than local) parameterspace.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** [**alternate** | **default**] *service-name*
5. paramspace *parameter-namespace parameter-name parameter-value*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
application

Example: Router(config)# application
```

Step 4 Enter service parameter configuration mode:

```
service service-name
```

Note When configuring a service that is defined, do not specify a location.

```
Example: Router(config-app)# service bator
```

Step 5 Configure the parameter's value:

```
paramspace parameter-namespace parameter-name parameter-value

Example:
Router(config-app-param)# paramspace ?
  tcl20base
  destination
  appcommon
  httpios
  indian
```

```
  callsetup
Router(config-app-param)#paramspace callsetup ?
  Parameters registered under callsetup namespace:
  name                 type  default value   description
  mode                 S     rotary          functional mode of this package
  reroutemode          S     rotary          call reroute handling option
  WORD  Parameter name


Router(config-app-param)#paramspace callsetup mode rotary
```

Note You could substitute the service configuration mode in Step 4 by package, dial-peer, or parameter group configuration submode. Configuring parameters using the **paramspace** command would then be similar to the above example.

Mapping Parameterspaces

Package parameter configuration mode provides the ability to map the parameterspace of another service or package to the local parameterspace. For example, if you are configuring a language application which uses the English package, the parameters registered in the English package are available for the application's use.

If you want parameters from another package (for example, the French package), made available to the application, you can configure the English package to use the parameters defined in the French package (under French parameterspace). This is called parameterspace mapping. This allows the English package to use all of the parameters defined in the French package's parameterspace without having to individually add the French parameters to the English package. This command substitutes a package's original parameterspace with a new one.

Use the following steps to map one package's parameterspace to another package's parameterspace.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **package** *package-name*
5. **use paramspace** *parameter-namespace*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

`enable`

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

`configure terminal`

`Example: Router# configure terminal`

Step 3 Enter application configuration mode to configure applications and services:

`application`

```
Example: Router(config)# application
```

Step 4 Enter service parameter configuration mode:

`package` *`package-name location`*

Note When configuring a package that is defined, do not specify a location.

```
Example: Router(config-app)# package english
```

Step 5 Configure the parameterspace to map to:

`use paramspace` *`parameter-namespace`*

```
Example: Router(config-app-param)# use paramspace french
```

Verifying Parameterspace Configuration

To verify parameterspace configuration, use the **show running-config** command**.**

```
application
service fax_detect flash:app_fax_detect.2.1.2.2.tcl
 paramspace callsetup mode redirect-rotary
 paramspace appl1 warning-time 60
 param mode redirect-at-alert
!
```

Troubleshooting Tips

If parameterspace configuration is not successful, verify the following:

- The application whose parameterspace you want to use is loaded.
- The application whose parameterspace you want to use contains the parameters that you want to configure.

Defining Parameter Groups

Use the **group-params** command to define groups of parameters. By defining a parameter group at the global application level, this group can then be used by as many dial peers, services, or packages as necessary.

For example, if a router has 10 dial peers configured to use the same service, but some of the dial peers need to use different values for the parameters in the service, you can configure parameter groups for each set of dial peers. Rather than configure all parameters under each dial peer individually, you create two parameter groups with a separate set of parameters.

To define parameter groups, use the following commands. These create a parameter group with parameters from two different parameterspaces.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. application
4. **group-params** *group-name*
5. **paramspace** *parameter-namespace parameter-name parameter-value*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
application

Example: Router(config)# application
```

Step 4 Enter group parameter configuration mode:

```
group-params group-name

Example: Router(config-app)# group-params grp1
```

Step 5 Define parameters using the **paramspace** command:

Note You **must** use the **paramspace** command to configure parameters in a group, because the local (default) parameter space is not defined here.

```
paramspace parameter-namespace parameter-name parameter-value

Example:
Router(config-app-param)#paramspace ?
  tcl20base
  cantonese
  media
  destination
  …
Router(config-app-param)#paramspace fax-detect1 retry-count 9
```

Verifying Parameter Group Definition

To verify parameter groups were defined, use the **show running-config** command.

```
application
```

```
service fax_detect flash:app_fax_detect.2.1.2.2.tcl
 paramspace callsetup mode redirect-rotary
 paramspace appl1 warning-time 60
 param mode redirect-at-alert
!
service fax_detect1 flash:app_fax_detect.2.1.2.2.tcl
param retry-count 5
!
service fax_detect2 flash:app_fax_detect.2.1.2.2.tcl
param pin-len 5
!
group-params grp
paramspace fax_detect2 pin-len 9
paramspace fax_detect1 retry-count 9
!
```

Troubleshooting Tips

If parameter group configuration is not successful, verify the following:

- The applications whose parameterspaces you want to group together are loaded.
- The applications whose parameterspaces you want to use contain the parameters that you want to configure.

Using Parameter Groups

To use the parameter group you defined in "Defining Parameter Groups" section on page 36 for a service, perform the following steps.

Note This example describes configuring a previously-defined group on a dial peer. However, a group could be configured at any other configuration level, such as service or package. The configuration process is similar in those cases.

If at a given configuration level (service, package or dial peer), the same parameter is configured individually and in a group, an individual configuration takes a precedence over a configuration in a group, meaning that the parameter would get the value defined as a part of individual parameter configuration.

In the examples shown in the configuration steps, the group is configured and applied to a service on different dial peers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **{pots | voip}**
4. service *service-name*
5. group-name *group-name*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Enter dial-peer configuration mode:

dial-peer number **pots**

```
Example: Router(config)# dial-peer 5 pots
```

Step 4 Configure a service to use under this dial peer:

service *service-name*

```
Example: Router(config-dial-peer)# service serv1
```

Step 5 Configure a parameter group:

group-name *group-name*

```
Example: Router(config-dial-peer)# group-name group1
```

Verifying Parameter Group Configuration

To verify the dial peer is using the parameter group, use the **show running-config** command.

```
.
.
.
!
group-params group1
    paramspace serv1 param1 valueA
    paramspace serv1 param2 valueB

group-params group2
    paramspace serv1 param1 valueC
    paramspace serv1 param2 valueD

dial-peer voice 1 pots
   service serv1
   group-name group1

dial-peer voice 5 pots
   service serv1
   group-name group1

dial-peer voice 6 pots
   service serv1
   group-name group2

dial-peer voice 10 pots
   service serv1
   group-name group2
!
```

.
.
.

Troubleshooting Tips

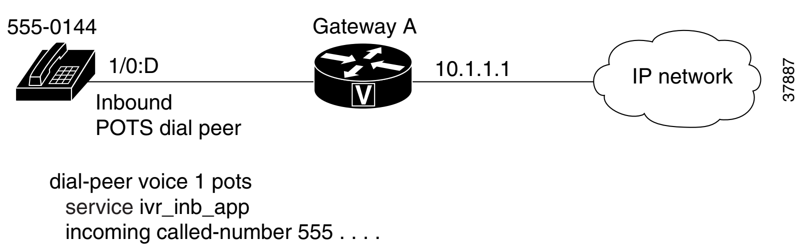
If dial peer parameter group configuration is not successful, verify the following:

- The applications are loaded on the dial peer.
- The parameter group is defined on the dial peer where the application is loaded.

Configuring an Inbound Application

If a voice call requires handling by an initial application first, for example, to collect the destination telephone number or to perform authentication and authorization, then an application must be configured in the inbound POTS dial peer. This type of application is called an "inbound application" because it is configured in the inbound dial peer. Some voice applications require only an inbound POTS dial peer, as shown in Figure 3-4, if no initial processing is necessary and if the call is not being sent over the IP network. Other applications require both an inbound POTS dial peer and an outbound VoIP dial peer, as described in the "Configuring an Outbound Application" section on page 52.

Figure 3-4 Inbound Application



Role of Dial Peers in Configuring Voice Applications

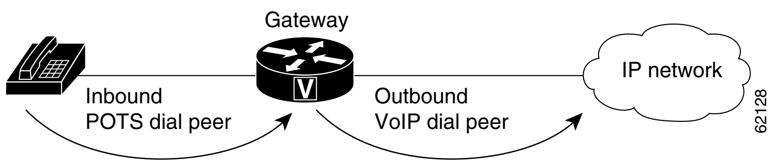
Dial peers are central to the configuration of Cisco voice gateways. They act as the focal point where associated elements come together, including:

- Name of the voice application to invoke
- VoiceXML documents, Tcl IVR scripts, and audio files used by the application
- Destination telephone number and DNIS maps that link callers to the voice application

Cisco voice gateways use dial peers to identify call origination and destination endpoints and to define the characteristics applied to each call leg in a call connection. A call leg is a logical connection between two routers or between a router and a telephony device.

Dial peers are required to link incoming calls to voice applications. These dial peers can be used to run voice applications from inbound or outbound call legs. Figure 3-5 shows a basic overview of two dial peers on the originating gateway that can be used for inbound or outbound applications.

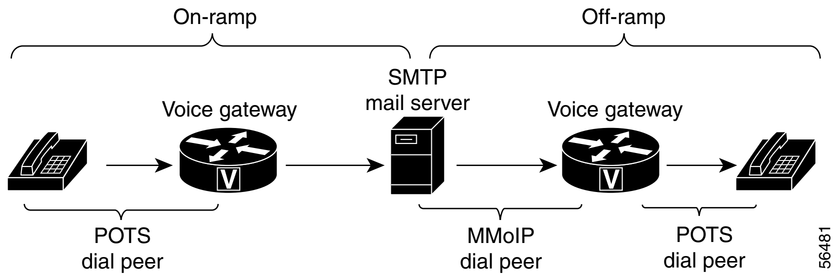
Figure 3-5 Voice Dial Peers



Three types of dial peers may be used by the Cisco voice gateway:

- Plain old telephone service (POTS)—Describe the characteristics of a traditional telephony network connection. POTS dial peers map a dialed string to a specific voice port on the local router, normally the voice port connecting the router to the local PSTN, PBX, or telephone. Only a POTS dial peer must be configured for an inbound voice application. Inbound applications do not require transferring the call to another telephone number or application.
- Voice over IP (VoIP)—Describe the characteristics of an IP network connection. VoIP dial peers map a dialed string to a remote network device, such as the destination router that is connected to the remote telephony device. A VoIP dial peer must be configured if the call is transferred to another telephone number or an application. For example, a Tcl IVR application on the inbound POTS dial peer may handle specific calls that come into the voice gateway, then pass them to a VoiceXML application configured on the outbound VoIP dial peer.
- Multimedia Mail over IP (MMoIP)—Describe the line characteristics generally associated with a packet network connection. With Voice Store and Forward, for example, this is the IP network connection between the on-ramp or off-ramp gateway and the SMTP server, as shown in Figure 3-6.

Figure 3-6 Voice Store and Forward Dial Peers



Note The Voice Store and Forward feature is available in Cisco IOS Release 12.2(11)T and later.

For more information on dial peers, see the *Dial Peer Configuration on Voice Gateway Routers* document, Cisco IOS Voice Configuration Library, Release 12.4T.

How Voice Applications are Matched to Called Numbers

When a call comes into the Cisco gateway, the gateway attempts to match the called number with a VoiceXML or Tcl application. The called number is linked to an application through a dial peer. For inbound calls, the dial peer is matched based on one of the following:

- Incoming called-number—A string representing the called number or dialed number identification service (DNIS). It is configured in a dial peer by using the **incoming called-number** command.

- Port—The voice port through which calls to this dial peer are placed. It is configured in POTS dial peers by using the **port** command. This is used only for calls inbound from the PSTN.

After the inbound dial peer is matched, if a DNIS map has been configured in the dial peer, the default application checks the DNIS map entries for a VoiceXML application to run.

For outbound calls, the dial peer is matched based on one of the following:

- Destination-pattern—A string representing the called number or DNIS. It is configured in a dial peer by using the **destination-pattern** command.
- DNIS map—A table containing multiple called numbers, each individually linked to the URL location of a VoiceXML document. It is configured in a dial peer by using the **dnis-map** command.

If the dialed string matches the destination pattern, the call is routed according to the voice port in POTS dial peers, or the session target in VoIP dial peers.

Using a destination pattern or incoming called number can be somewhat limiting because only one telephone number-to-voice application link can be configured per dial peer, although the dial peer string can contain wildcards. The dial peer, although configured to match on a wide range of dialed numbers, can only point to one voice application.

If several dial peers match a particular destination pattern, this is called a hunt group. The system attempts to place a call to the dial peer with the highest preference. If the call cannot be completed because of a system outage, for example, and the gatekeeper or gateway cannot be contacted, the hunt group feature performs the following:

- Retries the call to the next highest preference dial peer
- Continues until no more matching dial peers are found
- If there are dial peers with the same priority, the order is determined randomly

For detailed information about the dial peer commands described in this section, see the *Cisco IOS Voice Command Reference*, Release 12.3.

Direct Inward Dialing (DID) Behavior

Normally, when a voice call comes into the router, the gateway presents a dial tone to the caller and collects digits until it can identify an outbound dial peer. This process is called two-stage dialing. An application must be configured on the router's inbound dial peer to collect the digits (for example, the Tcl application "session"). After the dialed digits are collected, this application attempts to match them to a destination pattern or DNIS map configured in an outbound dial peer, and transfers the call to the application in the outbound dial peer (for example, a VoiceXML application).

With Direct Inward Dial (DID), the router does not present a dial tone to the caller and does not collect digits; the setup message contains all the digits necessary to route the call. For more information on DID, see the *Dial Peer Configuration on Voice Gateway Routers* document, Cisco IOS Voice Configuration Library, Release 12.3.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **pots**
4. **service** *service-name*
5. **incoming called-number** *string*

6. Configure a DNIS map.

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

`enable`

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

`configure terminal`

`Example: Router# configure terminal`

Step 3 Enter dial-peer configuration mode for a POTS dial peer:

`dial-peer voice` *`number`* **`pots`**

- *number*—Number tag that identifies this dial peer. Range is from 1 to 2,147,483,647.

`Example: Router(config)# dial-peer voice 110 pots`

Step 4 Associate an application with this inbound POTS dial peer:

`service` *`service-name`*

- *service-name*—Name of the voice application. This is the name of the application that was defined when the application was configured using the application configuration submodes.

`Example: Router(config-dial-peer)# service vapp1`

Step 5 Specify the called number that links voice calls to this dial peer:

`incoming called-number` *`string`*

- *string*—Sequence of digits representing the full or a partial telephone number (for example, the extension) used to reach the voice gateway. See the "How Voice Applications are Matched to Called Numbers" section on page 41 for more information.

`Example: Router(config-dial-peer)# incoming called-number 5550139`

Incoming calls that match this called number are linked to this dial peer and to the VoiceXML or Tcl application that is configured in Step 4.

Step 6 (Optional) To configure a DNIS map in this dial peer, see the "Configuring a DNIS Map for VoiceXML Applications" section on page 61.

Note The DNIS map is not used to select the inbound dial peer. If an inbound application uses a DNIS map, the inbound dial peer is selected based on the **incoming called-number** or **port**. The gateway then matches the called number to a VoiceXML document based on the DNIS map, provided that a VoiceXML application is configured in Step 4.

Troubleshooting Tips

If the voice application is not initiated, and instead you hear a dial tone, Table 3-7 lists some possible causes and the actions that you can take. If any of the following causes occur, the call is handed to the default application, which plays a dial tone to the user.

Table 3-7 Dial Tone Instead of Prerecorded Prompt: Troubleshooting

Possible Causes	Suggested Actions
No dial peer matches the call	Verify whether the correct dial peer is being matched for the call leg. See the "Troubleshooting Dial Peer Matching" section on page 49.
Dial peer is not configured with the application	Verify that the correct application is configured in the dial peer. See the "Troubleshooting Dial Peer Configuration" section on page 50.
Gateway cannot find the specified application	Verify that the application is configured on the gateway. See the "Verifying Loading of Service" section on page 26.

Verifying an Inbound Application Configuration

To verify an inbound application configuration, perform the following steps.

SUMMARY STEPS

1. **show running-config**
2. **show dial-peer voice** *tag*
3. Follow the steps in the "Verifying Loading of Service" section on page 26.
4. Follow the steps in the "Verifying the Gateway Configuration by Using a Sample Application" section on page 45.

DETAILED STEPS

Step 1 Use the **show running-config** command to display the dial-peer configuration for your inbound application. The following example shows that the VoiceXML application named vxml_inb_app handles inbound PSTN calls to 7-digit telephone numbers beginning with 555.

```
!
dial-peer voice 100 pots
 service vxml_inb_app
 incoming called-number 555....
 port 0:D
!
```

Step 2 Use the **show dial-peer voice** command to display detailed configuration information about the dial peer, including whether it is operational. The following example shows that dial peer 100 is linked to the application named vxml_inb_app.

```
Router# show dial-peer voice 100

VoiceEncapPeer100
        information type = voice,
        description = '',
        tag = 100, destination-pattern = '',
        answer-address = '', preference=0,
        numbering Type = 'unknown'
        group = 100, Admin state is up, Operation state is up,
        incoming called-number = '555....', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
```

```
huntstop = disabled,
in bound application associated: 'vxml_inb_app'
out bound application associated: ''
dnis-map =
permission :both
incoming COR list:maximum capability
outgoing COR list:minimum requirement
type = pots, prefix = '',
forward-digits default
session-target = '', voice-port = '0:D',
direct-inward-dial = disabled,
digit_strip = enabled,
register E.164 number with GK = TRUE

Connect Time = 0, Charged Units = 0,
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 0, Refused Calls = 0,
Last Disconnect Cause is "",
Last Disconnect Text is "",
Last Setup Time = 0.
```

Step 3 Follow the steps in the "Verifying Loading of Service" section on page 26 to verify that the voice application is loaded and running on the gateway.

Step 4 Follow the steps in the "Verifying the Gateway Configuration by Using a Sample Application" section on page 45 to verify that you can make a call into the gateway and the voice application is invoked.

Verifying the Gateway Configuration by Using a Sample Application

Cisco provides a sample VoiceXML document that you can use to verify that your gateway is configured properly to run voice applications. This simple application lets you test your dial peer configuration to confirm that the gateway can receive and place calls, and demonstrates the behavior of some basic VoiceXML elements.

To download the sample document and verify your configuration, complete the following steps:

Step 1 Log in to the Cisco.com website and go to the following location:

http://www.cisco.com/pcgi-bin/dev_support/access_level/product_support

You must have a Cisco.com login to access the above site. Qualified users can establish an account on Cisco.com by following the directions at http://tools.cisco.com/RPF/register/register.do. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you.

Step 2 Select VoiceXML Gateway from the VOICE TECHNOLOGY/IOS pull-down menu

Step 3 Select the DTMF ONLY Call Application link under the VoiceXML Sample Applications section.

Step 4 Download the simpleCall.zip package, which includes:

- Sample VoiceXML document (simpleCall.vxml)
- Nine prerecorded audio files:
  - busy.au
  - bye.au

- duration.au
- enter_dest.au
- no_input.au
- nomatch.au
- seconds.au
- technicalProblem.au
- welcome_test.au

Step 5 Unzip the files to a TFTP or FTP server.

Step 6 Copy the sample document and nine audio files into Flash memory on your gateway:

```
copy tftp flash
```

Step 7 Load the document into the gateway's memory and assign it the application name *callme*:

```
application
service callme flash:simpleCall.vxml
```

Step 8 Configure an inbound dial peer to trigger the application, as described in the "Configuring an Inbound Application" section on page 40, for example:

```
dial-peer voice 1 pots
 service callme
 incoming called-number 5550121 // Access number to dial into gateway
```

Step 9 Configure an outbound dial peer to place the call to the destination, for example:

```
dial-peer voice 2 voip
 destination-pattern .......
 session target ipv4:1.14.93.201
 codec g711ulaw
 no vad
```

The **session target** command must be configured with the IP address of the destination router.

Step 10 Place a call to the gateway's access number. In this example, you would dial 555-0121.

After the number is dialed, the welcome prompt (welcome_test.au) is played by the application and the caller is prompted for a 7-digit destination number (enter_dest.au). If the caller does not enter any digits, another prompt (no_input.au) is played requesting the destination number. The application collects the DTMF digits and places a call over IP to the destination, as specified by the **session target** command.

If the called party is busy or does not answer within 15 sec, an error prompt is played (busy.au).

Tip If the call is not successful, for example if the audio files do not play, or you encounter any other problem, see the "Troubleshooting Tips" section on page 59.

VoiceXML Document for Verifying Configuration on Gateway: Example

The following output shows the contents of the sample document simpleCall.vxml:

```
<?xml version="1.0"?>
<vxml version="2.0">

<!--
Cisco Voicexml Sample Code
File Name : simpleCall.vxml
Date      : Nov  18th 2002
```

```
Copyright (c) 2002 by Cisco Systems, Inc.
All rights reserved.

SAMPLE APPLICATION AND INFORMATION IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND BY
CISCO, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT
 NOT
LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY FITNESS FOR A PARTICULAR PURPOSE,
NONINFRINGEMENT, SATISFACTORY QUALITY OR ARISING
FROM A COURSE OF
DEALING, LAW, USAGE, OR TRADE PRACTICE. CISCO TAKES NO RESPONSIBILITY REGARDING ITS USAGE
IN AN APPLICATION. THE APPLICATION IS PROVID
ED AS AN EXAMPLE
ONLY, THEREFORE CISCO DOES NOT MAKE ANY REPRESENTATIONS REGARDING ITS RELIABILITY,
SERVICEABILITY, OR FUNCTION. IN NO EVENT DOES CISCO
WARRANT THAT THE
SOFTWARE IS ERROR FREE OR THAT CUSTOMER WILL BE ABLE TO OPERATE THE SOFTWARE WITHOUT
PROBLEMS OR INTERRUPTIONS. NOR DOES CISCO WARRANT
THAT THE SOFTWARE
OR ANY EQUIPMENT ON WHICH THE SOFTWARE IS USED WILL BE FREE OF VULNERABILITY TO INTRUSION
OR ATTACK. THIS SAMPLE APPLICATION IS NOT SUP
PORTED BY CISCO IN
ANY MANNER. CISCO DOES NOT ASSUME ANY LIABILITY ARISING FROM THE USE OF THE APPLICATION.
FURTHERMORE, IN NO EVENT SHALL CISCO OR ITS SUP
PLIERS BE LIABLE FOR
ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, LOST PROFITS, OR LOST DATA, OR ANY OTHER INDIRECT
DAMAGES EVEN IF CISCO OR ITS SUPPLIERS HAVE B
EEN INFORMED OF THE
POSSIBILITY THEREOF.

(10/15/2002)Modification : Fixed the playout of duration of call
-->

<catch event="error.badfetch">
  <prompt>
     <audio src="audio/technicalProblem.au"></audio>
  </prompt>
  <log>  Catch Handler  ::  Bad Fetch </log>
</catch>

<catch event="telephone.disconnect.transfer">
  <log>  Catch Handler  ::  Blind Transfer </log>
</catch>

<catch event="telephone.disconnect.hangup">
  <log> Catch Handler  ::  User disconnected </log>
</catch>

<var name="phone_num"/>
<var name="mydur"/>

<form id="main">

        <noinput>
                <log>  Catch Handler  ::  User did not enter input </log>
        <prompt>
            <audio src="audio/no_input.au"></audio>
        </prompt>
            <reprompt/>
        </noinput>

        <nomatch>
                <log>  Catch Handler  ::  User input does not match DTMF grammar</log>
        <prompt>
```

```
            <audio src="audio/nomatch.au"></audio>
        </prompt>
            <reprompt/>
        </nomatch>

        <block>
        <prompt bargein="true">
            <audio src="audio/welcome_test.au"></audio>
        </prompt>
        </block>

<!-- Prompt the user to enter the destination number and collect the 7 digits destination
number -->
<!-- Only DTMP inputs are accepted -->

        <field name="get_phone_num" type="number">
                <grammar type="application/grammar+regex">.......</grammar>
            <prompt bargein="true">
            <audio src="audio/enter_dest.au"></audio>
            </prompt>

        <filled>
            <assign name="phone_num" expr="get_phone_num"/>
                    <log>  FIELD ITEM  ::  User input collected is <value
expr="phone_num"/></log>
        </filled>
        </field>

     <transfer name="mycall" destexpr="'tel: ' + phone_num" connecttimeout="30s"
cisco-longpound ="true" bridge="true">

        <filled>
            <assign name="mydur" expr="mycall$.duration"/>

            <if cond = "mycall == 'busy'">
                           <prompt>
                             <audio src="audio/busy.au"></audio>
                           </prompt>
                           <log> TRANSFER ITEM   ::   Destination is busy</log>
            <elseif cond = "mycall == 'noanswer'"/>
                           <prompt>
                             <audio src="audio/noanswer.au"></audio>
                           </prompt>
                           <log> TRANSFER ITEM   ::   called party is not answering </log>
            <elseif cond = "mycall == 'near_end_disconnect'"/>
                           <log> TRANSFER ITEM   ::   Calling party disconnected </log>
            <elseif cond = "mycall == 'far_end_disconnect'"/>
                           <log> TRANSFER ITEM   ::   Called party disconnected </log>
                         <elseif cond = "mycall == 'unknown'"/>
                           <log>RANSFER ITEM   ::   Call transfer status is UNKNOWN</log>
                         <else/>
                           <prompt><audio src="audio/busy.au"></audio></prompt>
            </if>

            <log>TRANSFER ITEM   ::   The value in mycall is <value expr="mycall"/></log>
            <log>TRANSFER ITEM   ::   Duration of call is <value expr="mydur"/></log>
        </filled>
        </transfer>

          <block>
              <prompt>
                 <audio src="audio/bye.au"></audio>
              </prompt>
          </block>
```

```
  </form>
</vxml>
```

Troubleshooting Dial Peer Matching

Verification

Be sure that you perform the following verification steps:

- Final verification step in the "Configuring an Inbound Application" section on page 40
- "Verifying an Inbound Application Configuration" section on page 44
- "Verifying the Gateway Configuration by Using a Sample Application" section on page 45

Additional Troubleshooting Actions

To troubleshoot dial peer matching, perform these steps.

SUMMARY STEPS

1. **debug voip application**
2. **show dialplan number** *dial-string*

DETAILED STEPS

Step 1 Use the **debug voip application** command to verify that the gateway matches the correct dial peer for the application, for example:

```
Router# debug voip application

*Jan  4 20:51:57.084:InitiateCallSetup:Incoming[77] AlertTime 0
Destinations(1) [ 5550100  ]
*Jan  4 20:51:57.084:DNInitiate:Destination[5550100]
*Jan  4 20:51:57.084:DNInitiate:5550100 Did not match any peers
```

In the previous example, the gateway could not find a dial peer to match to the called number 555-0100.

Step 2 Use the **show dialplan number** command to verify which dial peer, if any, is being matched to the call, for example:

```
Router# show dialplan number 3800

Macro Exp.: 3800

VoiceEncapPeer3800
        information type = voice,
        description = '',
        tag = 3800, destination-pattern = '3800',
        answer-address = '', preference=0,
        CLID Restriction = None
        CLID Network Number = ''
        CLID Second Number sent
        source carrier-id = '', target carrier-id = '',
        source trunk-group-label = '',  target trunk-group-label = '',
        numbering Type = 'unknown'
        group = 3800, Admin state is up, Operation state is up,
        incoming called-number = '', connections/maximum = 0/unlimited,
```

```
        DTMF Relay = disabled,
        huntstop = disabled,
        in bound application associated: 'vxml_app'
        out bound application associated: ''
        dnis-map =
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        Translation profile (Incoming):
        Translation profile (Outgoing):
        incoming call blocking:
        translation-profile = ''
        disconnect-cause = 'no-service'
        type = pots, prefix = '',
        forward-digits all
        session-target = '', voice-port = '1/0:D',
        direct-inward-dial = disabled,
        digit_strip = disabled,
        register E.164 number with GK = TRUE
        fax rate = system,   payload size =  20 bytes

        Time elapsed since last clearing of voice call statistics never
        Connect Time = 0, Charged Units = 0,
        Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
        Accepted Calls = 0, Refused Calls = 0,
        Last Disconnect Cause is "",
        Last Disconnect Text is "",
        Last Setup Time = 0.
Matched: 3800   Digits: 4
Target:
```

The above output shows that called number 3800 maps to dial peer 3800. Verify that the application is configured in that dial peer.

Note For pointers to configuration examples, debug and voice command references, and troubleshooting documentation, see the "Related Documents" section on page 5.

Troubleshooting Dial Peer Configuration

Verification

Be sure that you perform the following verification steps:

- Final verification step in the "Configuring an Inbound Application" section on page 40
- "Verifying an Inbound Application Configuration" section on page 44
- "Verifying the Gateway Configuration by Using a Sample Application" section on page 45

Additional Troubleshooting Actions

To troubleshoot dial peer configuration, perform these steps.

SUMMARY STEPS

1. **show dial-peer voice** *tag*

2. **debug voip ccapi inout**

DETAILED STEPS

Step 1 Use the **show dial-peer voice** command to verify that the application is configured in the dial peer, for example:

```
Router# show dial-peer voice 555

VoiceEncapPeer555
        information type = voice,
        description = '',
        tag = 555, destination-pattern = '',
        answer-address = '', preference=0,
        CLID Restriction = None
        CLID Network Number = ''
        CLID Second Number sent
        source carrier-id = '', target carrier-id = '',
        source trunk-group-label = '',  target trunk-group-label = '',
        numbering Type = 'unknown'
        group = 555, Admin state is up, Operation state is up,
        incoming called-number = '5550121', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
        huntstop = disabled,
        in bound application associated: 'record'
        out bound application associated: ''
        dnis-map =
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        Translation profile (Incoming):
        Translation profile (Outgoing):
        incoming call blocking:
        translation-profile = ''
        disconnect-cause = 'no-service'
        voice-port = ''
        type = pots, prefix = '',
        forward-digits default
        session-target = '', up,
        direct-inward-dial = disabled,
        digit_strip = enabled,
        register E.164 number with GK = TRUE
        fax rate = system,   payload size =  20 bytes

        Time elapsed since last clearing of voice call statistics never
        Connect Time = 0, Charged Units = 0,
        Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
        Accepted Calls = 0, Refused Calls = 0,
        Last Disconnect Cause is "",
        Last Disconnect Text is "",
        Last Setup Time = 0.
```

Step 2 Use the **debug voip ccapi inout** command to verify whether the gateway is invoking the correct application. In the following example, the call is handed off to the default application because there is no application configured in the dial peer:

```
Router# debug voip ccapi inout

*Jan  1 02:54:14.591:cc_process_call_setup_ind (event=0x622EA710)
*Jan  1 02:54:14.591:>>>>CCAPI handed cid 71 with tag 1111 to app "DEFAULT"
```

In the following example, the call is handed off to the default application because the gateway could not find the application, getdigit, that was specified in the dial peer:

```
Router# debug voip ccapi inout

*Jan  1 02:58:14.987:cc_process_call_setup_ind (event=0x622EB1C0)
*Jan  1 02:58:14.987:%CALL_CONTROL-6-APP_NOT_FOUND:Application getdigit in dial-peer 1111
not found.
Handing callid 73 to default app.
>
*Jan  1 02:58:14.987:>>>>CCAPI handed cid 73 with tag 1111 to app "DEFAULT"
```

Additional Help

For pointers to configuration examples, debug and voice command references, and troubleshooting documentation, see the "Related Documents" section on page 5.

Configuring an Outbound Application

This section explains how to configure an outbound application in a dial peer.

Prerequisites

Before configuring an outbound application, you must first:

- Configure the name and location of the outbound application (see the "Loading a Service onto the Gateway" section on page 25).
- Configure an inbound application if initial processing is required, for example to collect digits from the caller such as account number or PIN for authentication and authorization (see the "Configuring an Inbound Application" section on page 40).

Note Only calls currently being handled by a VoiceXML or Tcl 2.0 application can be handed off to an outbound application. If a call is being handled by a Tcl 1.0 application or by the default application, the outbound application configured in the dial peer is ignored.

Outbound Voice Applications

An outbound application is a VoiceXML or Tcl application that is associated with one or more outbound dial peers on the voice gateway. An incoming call is linked with an inbound dial peer then transferred to the outbound application based on the called number.

An outbound VoIP dial peer is required when some preliminary processing of the call is necessary before the voice application is run, or when the call is being sent across the IP network. A preliminary application is configured in the inbound POTS dial peer, and the outbound application is configured in the outbound VoIP dial peer. Tcl applications such as session or clid_authen_collect, which are contained in Cisco IOS software, are commonly used in the inbound dial peer. These Tcl scripts collect dialed digits from the caller, then hand the call to the outbound application.

Before a call can be handed to an outbound application, the call must currently be handled by a VoiceXML or Tcl 2.0 application. Calls that are being handled by a Tcl 1.0 application or by the default application (DEFAULT) cannot be handed off to an outbound application.

For information on the behavior between Tcl and VoiceXML applications, see the "Call Handling Between Tcl and VoiceXML Applications" section on page 9.

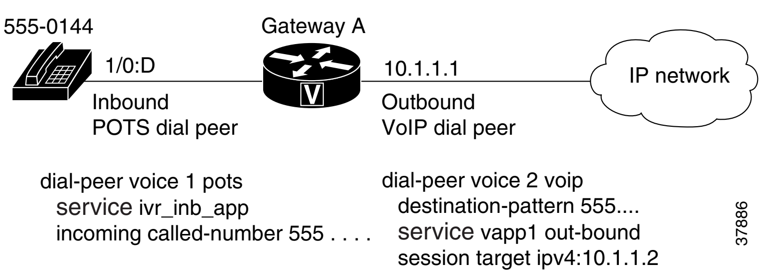
Call Scenario for an Outbound Application

The following is a typical call scenario for an outbound application:

1. An incoming call arrives at the voice gateway, which searches for a matching incoming called-number configured in a dial peer.
2. Finding the matching POTS dial peer (the inbound dial peer), the call is handed to the application (Tcl IVR or VoiceXML) configured in that dial peer.
3. The application configured in the inbound dial peer performs some function. For example, it may prompt the caller to enter a number (for example, 555-0121) and collects these digits.
4. The call is matched to an outbound VoIP dial peer by matching the dialed number (for example, 555-0121) to a destination pattern or to a DNIS map configured in the dial peer.
5. If a destination pattern is matched, the call is transferred to the application that is configured in the dial peer with the **service** command. If a DNIS map is matched, the call is transferred to the outbound VoiceXML application that is listed in the matching DNIS entry.
6. The application's document or script is executed, prompting the caller and providing information in return, or it transfers the call, depending on its design.

Figure 3-7 shows the dial peer configuration for a simple outbound application that does not transfer the call to a remote gateway.

Figure 3-7 Outbound Application without Transfer



SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **voip**
4. **service** *service-name* **out-bound**
5. **destination-pattern** *string*
6. **dnis-map** *map-name*
7. **session target ipv4**:*ip-address*
8. **session protocol** {**cisco** | **sipv2**}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

`enable`

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

`configure terminal`

`Example: Router# configure terminal`

Step 3 Enter dial-peer configuration mode for a VoIP dial peer:

`dial-peer voice` *`number`* **`voip`**

- *number*—Number tag that identifies this dial peer. Range is from 1 to 2,147,483,647.

`Example: Router(config)# dial-peer voice 2 voip`

Step 4 Link calls to the outbound VoiceXML application:

`service` *`service-name`* **`out-bound`**

- *service-name*—Name of the Tcl or VoiceXML application. This is the name of the application that was defined when the application was configured by using the application configuration submodes.
- **out-bound**—Indicates that outbound calls are handed off to the named application.

`Example: Router (config-dial-peer)# service vapp1 out-bound`

Note When using a DNIS map in an outbound dial peer, a VoiceXML application must be configured by using the **service** command with the **out-bound** keyword. Otherwise, the call is not handed off to the application that is specified in the URL of the DNIS map.

Step 5 Specify the called number that is matched to this dial peer:

`destination-pattern` *`string`*

- *string*—Sequence of digits representing the full or a partial telephone number (for example, the extension) used to reach the destination. See the "How Voice Applications are Matched to Called Numbers" section on page 41 for information.

`Example: Router(config-dial-peer)# destination-pattern 5550134`

Step 6 (Optional) Link a DNIS map to this dial peer:

`dnis-map` *`map-name`*

- *map-name*—Name of the DNIS map.

`Example: Router(config-dial-peer)# dnis-map dmap1`

To define a DNIS map, see the "Configuring a DNIS Map for VoiceXML Applications" section on page 61.

Note Destination patterns and DNIS maps are not mutually exclusive in the outbound dial peer; either one or both can be configured. When placing an outbound call, the called number can match either the destination-pattern or a DNIS map entry in the outbound dial peer.

Step 7 Specify the IP address of a terminating router:

`session target ipv4:`*`ip-address`*

- *ip-address*—IP address of the terminating router.

`Example: Router(config-dial-peer)# session target ipv4:10.10.1.1`

Note A session target value must be provided in the outbound VoIP dial peer, even if the application does not transfer the call to a terminating router. Any IP address is accepted for the session target; it does not have to be a valid address. If the outbound application transfers the call to another router, the IP address must be the valid address of that terminating router.

VoiceXML applications that send calls to a terminating router can use the <transfer> tag in the VoiceXML document. For more information, see the "Call Handling Between Tcl and VoiceXML Applications" section on page 9 and the *Cisco VoiceXML Programmer's Guide*.

Step 8 (Optional) Specify the session protocol if you want the dial peer to use SIP instead of H.323:

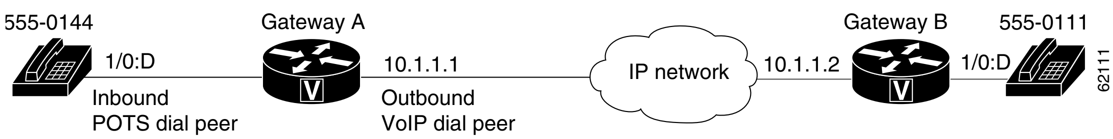
```
session protocol {cisco | sipv2}
```

The default session protocol is H.323. Configure **sipv2** to enable SIP.

Configuring an Outbound VoIP Dial Peer for Call Transfers

If a call that is being handled by an outbound application must be transferred across the IP network to the terminating gateway, an additional VoIP dial peer must be configured on the originating gateway. Figure 3-8 shows a simple network using an outbound application that transfers calls across the IP network to an application that is running on the terminating gateway.

Figure 3-8 Outbound Application With Transfer



The following example shows the dial peer configuration for the outbound application in Figure 3-8. The IP address specified with the **session target** command on the originating gateway must be the IP address of the terminating gateway, as shown for dial peer 4.

Gateway A (Originating)	Gateway B (Terminating)
dial-peer voice 1 pots service clid_authen_collect incoming called-number 555.... port 1/0:D ! dial-peer voice 3 voip dnis-map dmap1 service welcome out-bound session target ipv4:10.1.1.2 ! dial-peer voice 4 voip destination-pattern 555.... session target ipv4:10.1.1.2 dtmf-relay cisco-rtp h245-signal codec g711ulaw	dial-peer voice 1 voip service vxml_app1 incoming called-number 555.... session target ipv4:10.1.1.1 dtmf-relay cisco-rtp h245-signal codec g711ulaw ! dial-peer voice 2 pots destination pattern 555.... port 1/0:D !

Note For VoiceXML applications: At any time during a call transfer, including call setup, a user can terminate a call by pressing the # key for longer than 1 second, or by pressing the # key twice within two seconds. Pressing the # key for more than 1 second, or pressing ##, is treated as a long pound and disconnects the call. This feature is implemented by setting the cisco-longpound attribute to "true" in the <transfer> element in the VoiceXML document. For more information, see the *Cisco VoiceXML Programmer's Guide.*

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **voip**
4. **destination-pattern** *string*
5. **session target ipv4**:*ip-address*
6. **session protocol** {**cisco** | **sipv2**}
7. **dtmf-relay** {**cisco-rtp** | **h245-alphanumeric** | **h245-signal** | **rtp-nte**}
8. **codec** {**clear channel** | **g711alaw** | **g711ulaw** | **g723ar53** | **g723ar63** | **g723r53** | **g723r63** | **g726r16** | **g726r24** | **g726r32** | **g728** | **g729br8** | **g729r8**} [**bytes** *payload_size*]

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

`Example: Router# configure terminal`

Step 3 Enter dial-peer configuration mode for a VoIP dial peer:

dial-peer voice *number* **voip**

- *number*—Number tag that identifies this dial peer. Range is from 1 to 2,147,483,647.

`Example: Router(config)# dial-peer voice 2 voip`

Step 4 Specify the called number that is used to match this dial peer:

destination-pattern *string*

- *string*—Sequence of digits representing the full or a partial telephone number (for example, the extension) used to reach the destination.

`Example: Router(config-dial-peer)# destination-pattern 5550134`

Step 5 Specify the IP address of the terminating router:

session target ipv4:*ip-address*

- *ip-address*—IP address of the terminating router.

`Example: Router(config-dial-peer)# session target ipv4:10.10.1.1`

Step 6 (Optional) Specify the session protocol if you want the dial peer to use SIP instead of H.323:

session protocol {**cisco** | **sipv2**}

The default session protocol is H.323. Configure **sipv2** to enable SIP.

Step 7 (Optional) Specify the DTMF relay method used in the dial peer:

dtmf-relay {**cisco-rtp** | **h245-alphanumeric** | **h245-signal** | **rtp-nte**}

- **cisco-rtp**—Forwards DTMF tones by using Real-Time Transport Protocol (RTP) with a Cisco proprietary payload type.
- **h245-alphanumeric**—Forwards DTMF tones by using the H.245 "alphanumeric" user input indication method. Supports tones 0–9, *, #, and A–D.
- **h245-signal**—Forwards DTMF tones by using the H.245 "signal" user input indication method. Supports tones 0–9, *, #, and A–D.
- **rtp-nte**—Forwards DTMF tones by using Real-Time Transport Protocol (RTP) with the named telephone event (NTE) payload type. This is the required format for calls using SIP.

```
Example: Router(config-dial-peer)# dtmf-relay cisco-rtp h245-signal
```

Note If digit collection is needed on an inbound IP call leg, DTMF Relay is required.

Step 8 (Optional) Specify the codec type in the dial peer:

codec {**clear channel** | **g711alaw** | **g711ulaw** | **g723ar53** | **g723ar63** | **g723r53** | **g723r63** | **g726r16** | **g726r24** | **g726r32** | **g728** | **g729br8** | **g729r8**} [**bytes** *payload_size*]

- clear-channel—Clear channel at 64,000 bits per second (bps).
- g711alaw—G.711 a-law at 64,000 bps
- g711ulaw—G.711 u-law at 64,000 bps
- g723ar53—G.723.1 Annex A at 5,300 bps
- g723ar63—G.723.1 Annex A at 6,300 bps
- g723r53—G.723.1 at 5,300 bps
- g723r63—G.723.1 at 6,300 bps
- g726r16—G.726 at 16,000 bps
- g726r24—G.726 at 24,000 bps
- g726r32—G.726 at 32,000 bps
- g728—G.728 at 16,000 bps
- g729br8—G.729 Annex B at 8,000 bps
- g729r8—G.729 at 8,000 bps. This is the default codec.

```
Example: Router(config-dial-peer)# codec g723r53
```

Note The codec values that are supported by this command vary depending on the platform, Cisco IOS release, and call signaling protocol. For specific details, see the "Codec Support for Audio Recording" section on page 5.

For audio recording and playout over an IP call leg, the codec negotiated between the originating and terminating IP end points must match the codec specified in the VoiceXML document. If the codec used for the VoIP call is different than the codec defined for the audio file in the VoiceXML document, the recording and playback fails and an error is generated.

Note For Cisco 3600 series: A specific codec can be configured in the dial peer as long as it is supported by the **codec complexity** command configured on the voice card. The **codec complexity** command is set to either **high** or **medium**; the setting determines which codecs are supported. For more information, see the "Modifying Codec Complexity on the Cisco 3600 Series" section on page 13.

Verifying the Outbound Application Configuration

To verify an outbound application configuration, perform the following steps.

SUMMARY STEPS

1. **show running-config**
2. **show dial-peer voice** *number*
3. Follow the steps in the "Verifying Loading of Service" section on page 26.

DETAILED STEPS

Step 1 Use the **show running-config** command to display the dial-peer configuration for your outbound application. The following example shows that the DNIS map named dmap1 is used to link called numbers to the URL of specific VoiceXML documents. If a URL is not provided in the DNIS entry, the called number is linked to the vapp1 application.

```
!
dial-peer voice 101 voip
 dnis-map dmap1
 service vapp1 out-bound
 session target ipv4:10.10.1.1
```

Step 2 Use the **show dial-peer voice** command to verify that the application is configured as an outbound application in the dial peer, and that the dial peer is operational. The following example shows that dial peer 101 is linked to the outbound application named vapp1 and uses the DNIS map named dmap1.

```
Router# show dial-peer voice 101

VoiceOverIpPeer101
        information type = voice,
        description = '',
        tag = 101, destination-pattern = '',
        answer-address = '', preference=0,
        numbering Type = 'unknown'
        group = 101, Admin state is up, Operation state is up,
        incoming called-number = '', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
        modem passthrough = system,
        huntstop = disabled,
        in bound application associated: 'DEFAULT'
        out bound application associated: 'vapp1'
        dnis-map = 'dmap1'
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        type = voip, session-target = 'ipv4:10.10.1.1',
```

```
technology prefix:
settle-call = disabled
ip media DSCP = default, ip signaling DSCP = default, UDP checksum = di,
session-protocol = cisco, session-transport = system, req-qos = best-ef
acc-qos = best-effort,
RTP dynamic payload type values: NTE = 101
Cisco: NSE=100, fax=96, fax-ack=97, dtmf=121, fax-relay=122
       CAS=123, ClearChan=125, PCM switch over u-law=126,A-law=127
fax rate = voice,   payload size =  20 bytes
fax protocol = system
fax NSF = 0xAD0051 (default)
codec = g729r8,   payload size =  20 bytes,
Expect factor = 0, Icpif = 20,
Playout Mode is set to default,
Initial 60 ms, Max 300 ms
Playout-delay Minimum mode is set to default, value 40 ms
Expect factor = 0,
Max Redirects = 1, Icpif = 20,signaling-type = ext-signal,
CLID Restrict = disabled
VAD = enabled, Poor QOV Trap = disabled,
voice class perm tag = ''
Connect Time = 0, Charged Units = 0,
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 0, Refused Calls = 0,
Last Disconnect Cause is "",
Last Disconnect Text is "",
Last Setup Time = 0.
```

Note If the application is not configured in the dial peer by using the **out-bound** keyword with the **service** command, the gateway places the call to the outbound call leg as specified by the session target, rather than handing the call off to the VoiceXML application.

Step 3 Follow the steps in the "Verifying Loading of Service" section on page 26 to verify that the voice application is loaded and running on the gateway.

Troubleshooting Tips

If the call is not transferred to the outbound application, Table 3-8 lists some possible causes and the actions that you can take.

Table 3-8 Call Not Transferred to Outbound Application: Troubleshooting

Possible Causes	Suggested Actions
No dial peer matches the call	Verify whether the correct dial peer is being matched for the call leg. See the "Troubleshooting Dial Peer Matching" section on page 49.
Application is not configured as an outbound application in the outbound dial peer	Verify that the application is configured as an outbound application in the dial peer. See the "Verifying the Outbound Application Configuration" section on page 58.

Table 3-8 Call Not Transferred to Outbound Application: Troubleshooting (continued)

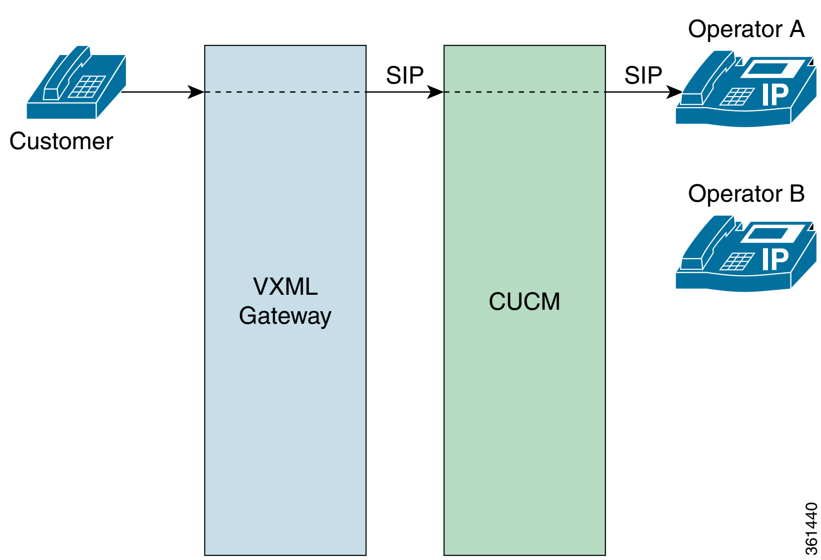
Possible Causes	Suggested Actions
Gateway cannot find the specified application	Verify that the application is configured on the gateway. See the "Verifying Loading of Service" section on page 26.
Inbound dial peer does not have the correct application configured	Verify that the application configured in the inbound dial peer is a VoiceXML or Tcl 2.0 application. Transfer to or from a Tcl 1.0 application, or the default application, is not supported.

Configuring VoiceXML with the SIP Phone

When you deploy the VoiceXML application with the SIP phone and transfer a call from operator A to operator B, the Cisco Unified Communication Manager (CUCM) does not send the SIP REFER message during the transfer, The CUCM sends an invite with the IP address of the remote endpoint to the customer and operator B through the VoiceXML gateway. This results in a call loop, and the quality of the audio deteriorates.

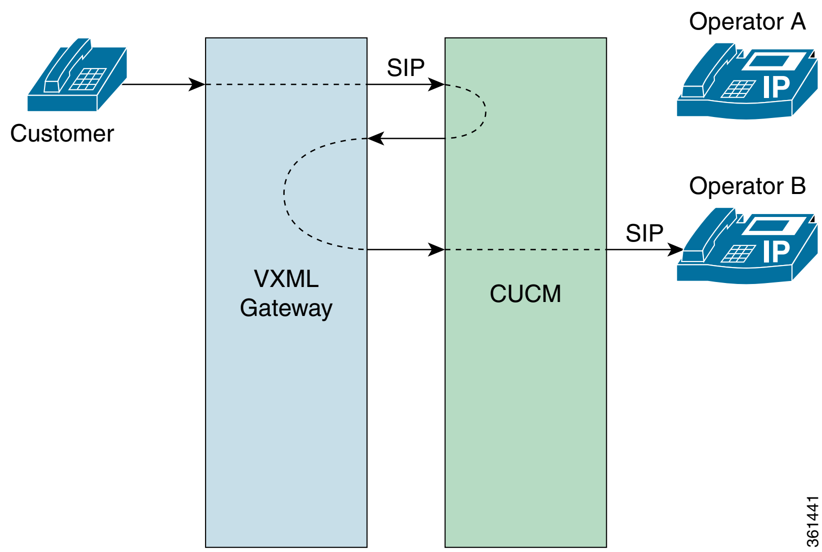
Call loop occurs when an operator transfers a call through the VoiceXML application to another operator, as shown in Figure 3-9.

Figure 3-9 Transfering a Call Through the VoiceXML Application



To prevent call loops from occurring, install a two-legged consultation call-transfer setup without an Exclusive OR (XOR), as shown in Figure 3-10.

Figure 3-10 Two-Legged Consultation Transfer



To enable the two-legged consulation transfer, use the **xfer-out= true** keyword in the VoiceXML script. Also, by default, the **supplementary-service sip refer** command should be enabled. To disable this feature, do not use the **xfer-out= true** keyword in the VoiceXML script or use the **no** s**upplementary-service sip refer** command.

For information on configuring the call-transfer method by using VoiceXML or Tcl properties, see the *Cisco VoiceXML Programmer's Guide* or the *Tcl IVR API Version 2.0 Programmer's Guide*, respectively.

Configuring a DNIS Map for VoiceXML Applications

This section explains how to create and apply a DNIS map for use with VoiceXML applications.

DNIS Maps

An easy way of linking a called number to the desired VoiceXML document is by creating a DNIS map and configuring it in the dial peer for a VoiceXML application. This method has a number of advantages over using a destination pattern or incoming called number.

A DNIS map can contain multiple telephone numbers, each individually linked to the URL location of a VoiceXML document, and associated with a single dial peer. While a destination pattern, using wildcards, can match on a range of telephone numbers, it can link to only one VoiceXML document. In addition, a DNIS map can be a separate text file that is stored in an external location (for example, a TFTP server) and easily updated.

Note DNIS maps are designed for VoiceXML applications. The URLs used in DNIS maps are not supported for non-VoiceXML applications, such as Tcl applications.

Using Cisco IOS Software or Text Files for DNIS Maps

There are two ways to create DNIS maps for VoiceXML applications. Both methods start with the **voice dnis-map** command to create and name a DNIS map, then use one of the following options:

- Cisco IOS software—You add DNIS entries to the DNIS map and store all the information on the voice gateway.

  If you do not enter the optional *url* attribute for the **voice dnis-map** command, the gateway enters DNIS-map configuration mode. You then use the **dnis** command to enter DNIS entries, one at a time.

- An external text file—You create the DNIS entries in a text file and store the file on a server connected to the gateway, for example on a TFTP server.

  If you create a text file with the DNIS information (see the example below), you provide the *url* for this file when using the **voice dnis-map** command. Using this method, a network administrator can create and maintain a single master file of all DNIS map entries. This file can be used by every voice gateway that requires it, sparing the configuration of each DNIS entry on each gateway. The text file can be easily updated and then reloaded by using the **voice dnis-map load** command. Following is an example of the contents of a DNIS map text file:

```
!This is an example of the contents of a DNIS map text file.
!Comments are preceded by a "!"
!
dnis 5550101 url tftp://global/tickets/vapp1.vxml
dnis 5550102 url rtsp://global/stocks/vapp2.vxml
.
dnis 5550199 url http://global/sports/vapp99.vxml
```

Note External DNIS map text files must be stored on TFTP servers; they cannot be stored on HTTP or RTSP servers, although the individual entries in a DNIS map can include URLs for HTTP and RTSP servers.

DNIS Map Limits

DNIS maps configured in Cisco IOS software use 100 bytes of memory for each entry plus the memory allocated for the URL. You may configure as many DNIS map entries as allowed by the available configuration memory of the gateway. If you create a DNIS map through Cisco IOS software that is too large for the available configuration memory, you will be unable to save the configuration. DNIS maps that are larger than the configuration memory of the gateway must be maintained in an external text file.

It is generally recommended that you limit DNIS maps to less than 10,000 entries, depending on the platform and system architecture. DNIS maps with more than a few hundred entries should normally be maintained in an external text file.

URLs in DNIS Maps

URLs in DNIS maps are used only for VoiceXML applications. If a dial peer is configured to use a VoiceXML application, as defined by the **service** command, that application loads the VoiceXML document from the URL that is linked to the called number in the DNIS map.

Non-VoiceXML applications, such as Tcl applications, ignore the URLs in DNIS maps and instead load the application that is configured in the dial peer using the **service** command. If a DNIS map is configured in an inbound dial peer, but that dial peer is not linked to a VoiceXML application, the gateway ignores the URL in the DNIS map. If an outbound dial peer is not linked to an outbound VoiceXML application, as defined by using the **outbound** keyword in the **service** command, the gateway ignores the URL in the DNIS map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice dnis-map** *map-name* [*url*]
4. **dnis** *number* **url** *url*
5. **exit**
6. **dial-peer voice** *number* **voip**
7. **dnis-map** *map-name*
8. **end**
9. **voice dnis-map load** *map-name*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Create and name a DNIS map:

voice dnis-map *map-name* [*url*]

- *map-name*—Name of the DNIS map.
- *url* (optional)—URL of an externally stored text file that is used as the DNIS map.

Note If no URL is entered, the gateway enters DNIS-map configuration mode, as shown in Step 4. If you enter a URL for an externally stored DNIS map file, skip Step 4.

```
Example 1: Router(config)# voice dnis-map dmap1
Example 2: Router(config)# voice dnis-map dmap2 http://dnismaps/dnismap2
```

The following example shows the contents of a DNIS map text file:

```
!This is an example of the contents of a DNIS map text file.
!Comments are preceded by a "!"
!
dnis 5550101 url tftp://global/tickets/vapp1.vxml
dnis 5550102 url rtsp://global/stocks/vapp2.vxml
.
dnis 5550199 url http://global/sports/vapp99.vxml
```

Step 4 (Optional) Add DNIS entries to a DNIS map on the gateway:

dnis *number* **url** *url*

- *number*—User-selected DNIS number.

- *url*—URL of a specific VoiceXML document. If a URL is not entered, the DNIS number is linked to the VoiceXML application that is assigned to the dial peer using the **service** command.

Note Skip this step if in Step 1 you entered the URL to a DNIS map text file.

```
Example: Router(config-dnis-map)# dnis 5550112 url rtsp://global/orders/vapp-abc.vxml
```

Step 5 (Optional) Exit DNIS-map configuration mode and return to global configuration mode:

exit

```
Example: Router(config-dnis-map)# exit
```

Step 6 Enter dial-peer configuration mode for the dial peer where the VoiceXML application is configured:

dial-peer voice *number* **voip**

- *number*—Number tag used to identify this dial peer. Range is from 1 to 2,147,483,647.

```
Example: Router(config)# dial-peer voice 555 voip
```

Step 7 Link the DNIS map to this dial peer:

dnis-map *map-name*

- *map-name*—Name of the DNIS map.

```
Example: Router(config-dial-peer)# dnis-map dmap1
```

Note To use DNIS maps in outbound dial peers, the call application must be configured as an outbound application by using the **out-bound** keyword with the **service** command. Otherwise, the call is not handed off to the VoiceXML application that is specified in the URL of the DNIS map.

Step 8 (Optional) Exit dial-peer configuration mode and enter privileged EXEC mode:

end

```
Example: Router(config)# end
```

Step 9 (Optional) Reload an updated version of a DNIS map file that is located on an external server:

voice dnis-map load *map-name*

- *map-name*—Name of the DNIS map.

```
Example: Router# voice dnis-map load dnismap1
```

Verifying DNIS Map Configuration

To verify DNIS map configuration, perform the following steps.

SUMMARY STEPS

1. **show running-config**
2. **show voice dnis-map** *dnis-map-name*
3. **show voice dnis-map summary**
4. **show dial-peer voice** *number*

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that the DNIS map is defined and assigned to the dial peer, for example:

```
!
voice dnis-map dmap1 tftp://tftp-host/config-files/dmaps1.cfg
!
voice dnis-map dmap2
  dnis 5550111 url http://http-host/vxml/app-for-5550111.vxml
  dnis 5550122 url http://http-host/vxml/app-for-5550122.vxml
  dnis 5550133 url http://http-host/vxml/app-for-5550133.vxml
!
!
dial-peer voice 555 voip
 dnis-map dmap2
 service welcome out-bound
 session target ipv4:10.10.1.1
!
```

Step 2 Use the **show voice dnis-map** command to verify the configuration of the DNIS map. If the DNIS map uses a text file stored on an external server, this command also shows whether the file was successfully loaded, for example:

```
Router# show voice dnis-map dmap1

Dnis-map dmap1
-----------------------------------------
  It has 0 entries
  It is populated from url tftp://tftp-host/config-files/dmaps1.cfg


DNIS                URL
----                ---
```

Step 3 Use the **show voice dnis-map summary** command to see all DNIS maps that are configured on the gateway, for example:

```
Router# show voice dnis-map summary

There are 3 dnis-maps configured

dnis-map            Entries    URL
--------            -------    ---
dmap1               0        tftp://tftp-host/config-files/dmaps1.cfg
dmap2               3
*dmap4              0        http:/sanjose/doclabs/published/dnismaps.txt
```

Note If an asterisk is displayed next to the DNIS map name when using the **summary** keyword, it means that the DNIS map is configured, but not running. Normally this is because the external text file was not successfully loaded

Step 4 Use the **show dial-peer voice** command to verify that the DNIS map is configured in the associated dial peer, for example:

```
Router# show dial-peer voice 555

VoiceOverIpPeer555
        information type = voice,
        description = '',
        tag = 555, destination-pattern = '',
```

```
answer-address = '', preference=0,
numbering Type = 'unknown'
group = 555, Admin state is up, Operation state is up,
incoming called-number = '', connections/maximum = 0/unlimited,
DTMF Relay = disabled,
modem passthrough = system,
huntstop = disabled,
in bound application associated: 'DEFAULT'
out bound application associated: 'welcome'
dnis-map = 'dmap2'
permission :both
incoming COR list:maximum capability
outgoing COR list:minimum requirement

...
```

Modifying HTTP Client Settings

The default HTTP settings are recommended. This section explains how to modify the default HTTP client settings if necessary.

Note For information about the HTTP 1.1 client features that are supported by Cisco IOS software, see the "HTTP Client Support" section on page 11.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **http client cookie**
4. **http client cache memory** {**file** *size* | **pool** *size*}
5. **http client connection idle timeout** *seconds*
6. **http client connection timeout** *seconds*
7. **http client response timeout** *seconds*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Enable the HTTP client to send and receive cookies when using VoiceXML applications:

http client cookie

Note This command is enabled by default. You do not need to enter it unless you have previously disabled cookie support by using the **no http client cookie** command.

Step 4 Change the HTTP client cache size from the default:

`http client cache memory` {**`file`** *`size`* | **`pool`** *`size`*}

- *size*—Maximum file size allowed for caching in kilobytes. Valid range is 1 to 10,000. Any larger file size is not cached. The default is 2 KB.
- *size*—Maximum memory pool size for caching in kilobytes. Valid range is 1 to 100,000. Setting the memory pool size to 0 disables HTTP caching. The default is 100 KB.

A larger cache size permits caching of frequently used files, decreasing the fetching time necessary between the client and server and increasing performance.

```
Example: Router(config)# http client cache memory file 1000
```

Step 5 Change the HTTP client/server idle connection timeout from the default:

`http client connection idle timeout` *`seconds`*

- *seconds*—Seconds that the HTTP client waits before terminating an idle connection. Valid range is 1 to 60 sec. The default is 2 sec.

```
Example: Router(config)# http client connection idle timeout 30
```

Step 6 Change the HTTP client/server connection timeout from the default:

`http client connection timeout` *`seconds`*

- *seconds*—Seconds that the HTTP client waits for a server to establish a connection before giving up. The valid range is 1 to 60 sec. The default is 5 sec.

```
Example: Router(config)# http client connection timeout 30
```

Step 7 Change the HTTP client/server response time from the default:

`http client response timeout` *`seconds`*

- *seconds*—Seconds that the HTTP client waits for a response from the server after making a request. The range is 1 to 300 sec. The default is 10 sec.

```
Example: Router(config)# http client response timeout 60
```

Configuring HTTPS

Note The Cisco IOS **ip http client** commands do not apply to the HTTP client referred to in this section. To configure HTTPS, use the **http client** commands listed in the steps in this section only.

To configure HTTPS you must perform the following tasks:

- Configuring an Exclusive Trustpoint and Cipher Suites, page 67
- Configuring the HTTP Client to Obtain a Certificate, page 68

Configuring an Exclusive Trustpoint and Cipher Suites

To configure a secure certification authority (CA), or trustpoint, and specify cipher suites for the HTTP client, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **http client secure-trustpoint** *name*
4. **http client secure-ciphersuite** {[**3des_cbc_sha**] [**des_cbc_sha**] [**null_md5**] [**rc4_128_md5**] [**rc4_128_sha**]}
5. **exit**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

`enable`

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

`configure terminal`

`Example: Router# configure terminal`

Step 3 Declare the trustpoint that the HTTP client should use:

`http client secure-trustpoint` *`name`*

`Example: Router(config)# http client secure-trustpoint myca`

Step 4 Sets the secure encryption cipher suites for the HTTP client:

`http client secure-ciphersuite` `{[`**`3des_cbc_sha`**`] [`**`des_cbc_sha`**`] [`**`null_md5`**`] [`**`rc4_128_md5`**`] [`**`rc4_128_sha`**`]}`

- If this command is not entered, the HTTP client uses all of the following cipher suites.
- **3des_cbc_sha**—Triple DES (Data Encryption Standard) encryption and the SHA (Secure Hash Algorithm) integrity method.
- **des_cbc_sha**—DES encryption and the SHA integrity method.
- **null_md5**—NULL encryption and the MD5 (Message-Digest algorithm 5) integrity method.
- **rc4_128_md5**—RC4 (or ARCFOUR) encryption and the MD5 integrity method.
- **rc4_128_sha**—RC4 encryption and the SHA integrity method.

`Example: Router(config)# http client secure-ciphersuite 3des_cbc_sha rc4_128_md5`

Step 5 Exit global configuration mode:

`exit`

`Example: Router(config)# exit`

Configuring the HTTP Client to Obtain a Certificate

You configure an HTTP client with manually entered certificate information or you can configure it to obtain a certificate from an external HTTP server.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **crypto pki trustpoint** *name*
4. **enrollment terminal** [**pem**]

   or

   enrollment [**mode**] [**retry period** *minutes*] [**retry count** *number*] **url** *url* [**pem**]
5. **exit**
6. **crypto pki authenticate** *name*
7. **ntp server** *ip-address* | *hostname* [**version** *number*] [**key** *key-id*] [**source** *interface*] [**prefer**]
8. **exit**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Declare the trustpoint that your router should use:

crypto pki trustpoint *name*

```
Example: Router(config)# crypto pki trustpoint myca
```

Step 4 Specify manual cut-and-paste certificate enrollment:

enrollment terminal [**pem**]

```
Example: Router(ca-trustpoint)# enrollment terminal
```

or

Specify the enrollment parameters of a certification authority (CA):

enrollment [**mode**] [**retry period** *minutes*] [**retry count** *number*] **url** *url* [**pem**]

```
Example: Router(ca-trustpoint)# enrollment url http://myserver
```

Step 5 Exit ca-trustpoint configuration mode:

exit

```
Example: Router(ca-trustpoint)# exit
```

Step 6 Authenticate the certification authority by getting the certificate of the CA:

crypto pki authenticate *name*

```
Example: Router(config)# crypto pki authenticate rsa
```

If you entered the **enrollment terminal** command in Step 4, you will be prompted to enter the CA certificate. Cut and paste the certificate at the command line, then press Enter, and type “quit.” The router prompts you to accept the certificate. Enter "yes" to accept the certificate.

If you entered the **enrollment url** command in Step 4, the router obtains the certificate from the trustpoint you configured and prompts you to accept the certificate. Enter “yes” to accept the certificate.

Step 7 Allow the software clock to be synchronized by a Network Time Protocol (NTP) time server:

ntp server *ip-address* | *hostname* [**version** *number*] [**key** *key-id*] [**source** *interface*] [**prefer**]

```
Example: Router(config)# ntp server 10.1.200.10
```

Step 8 Exit global configuration mode:

```
    exit
Example: Router(config)# exit
```

Verifying HTTP Client Settings

To verify HTTP client settings, perform the following steps.

SUMMARY STEPS

1. **show running-config**
2. **show http client cache**
3. **show http client connection**
4. show http client secure status

DETAILED STEPS

Step 1 Use the **show running-config** command to verify the HTTP configuration information. If the defaults are used, HTTP configuration commands are not shown. If changes have been made to the HTTP defaults, the configuration is shown. For example, the following output shows each HTTP parameter configured to a non-default value:

```
http client cache memory pool 200
http client cache memory file 5
http client cache refresh 20
no http client connection persistent
http client connection timeout 20
http client connection idle timeout 60
http client response timeout 20
!
```

Step 2 Use the **show http client cache** command to verify that the HTTP client cache is configured as intended. The command output lists files cached with URLs of both HTTP and HTTPS schemes in separate tables, for example:

```
Router# show http client cache

HTTP Client cached information
==============================
Maximum memory pool allowed for HTTP Client caching = 10000 K-bytes (default)
Maximum file size allowed for caching = 50 K-bytes (default)
Total memory used up for Cache = 4271 Bytes
Message response timeout = 10 secs
Total cached entries = 2
Total non-cached entries = 0

          Cached entries
           ==============
entry 135, 2 entries
Ref   FreshTime   Age          Size        context
---   ---------   ---          ----        -------
0     121393      557          1419        0
url: http://10.1.200.21/vxml/menu_main.vxml

1     121447      13           2119        0
url: https://10.1.200.21/catalog/advance.vxml
```

Step 3 Use the **show http client connection** command to verify that the HTTP client connection is configured as intended, for example:

```
Router# show http client connection

     HTTP Client Connections:
     ========================
Persistent connection   = enabled
Initial socket connection timeout = 20 secs
Connection idle timeout = 60 secs
Total HTTP server connections = 0
```

Step 4 Use the **show http client secure status** command to display the trustpoint and cipher suites that are configured in the HTTP client, for example:

```
Router# show http client secure status

HTTP Client Secure Ciphersuite: rc4-128-md5 rc4-128-sha 3des-cbc-sha des-cbc-sha null-md5
HTTP Client Secure Trustpoint: myca
```

Configuration Examples for Tcl IVR and VoiceXML Applications

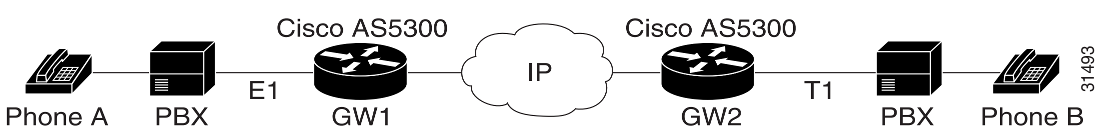
This section provides the following gateway configuration examples of Cisco Tcl and VoiceXML applications. It contains comments in places especially relevant to the configuration of the feature.

- Tcl IVR 2.0 Examples, page 71
- Inbound VoiceXML Application: Example, page 77
- Outbound VoiceXML Application with DNIS Map: Example, page 79

Tcl IVR 2.0 Examples

Figure 3-11 shows the type of topology used in the configuration for the example.

Figure 3-11 Tcl IVR Example Configuration Topology



In this example configuration, GW1 is running IVR for phone A, and GW2 is running IVR for phone B.

This section provides the following configuration examples:

- GW1 IVR Configuration: Example
- GW2 IVR Configuration: Example

GW1 IVR Configuration: Example

```
!
version 12.3
```

```
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname GW1
!
logging buffered 100000 debugging
aaa new-model
aaa authentication login default local group radius
aaa authentication login h323 group radius
aaa authentication login con none
aaa authorization exec h323 group radius
aaa accounting connection h323 start-stop group radius
enable password xxx
!
username lab password 0 lab
!
!
resource-pool disable
!
!
clock timezone PST -8
ip subnet-zero
ip host blue 10.14.124.xxx
ip host green 223.255.254.254
ip host rtspserver3 10.14.1xx.2
ip host rtspserver1 10.14.1xx.2
!
mgcp package-capability trunk-package
mgcp default-package trunk-package
isdn switch-type primary-net5
isdn voice-call-failure 0
!
!
application
    service debit_card tftp://blue/Router/scripts.new/app_debitcard.tcl
    param uid-len 6
    paramspace english language en
    paramspace english index 1
    paramspace english location tftp://blue/hostname/WV/en_new/
    paramspace chinese language ch
    paramspace chinese index 2
    paramspace chinese location tftp://blue/hostname/WV/ch_new/
    !
    service debit_card_rtsp tftp://blue/IVR 2.0/scripts.new/app_debitcard.tcl
    param uid-len 6
    paramspace english language en
    paramspace english index 1
    paramspace english location rtsp://rtspserver1:554/
    paramspace chinese language ch
    paramspace chinese index 2
    paramspace chinese location rtsp://rtspserver1:554/
    !
    !
mta receive maximum-recipients 0
!
!
controller E1 0
 clock source line primary
 pri-group timeslots 1-31
!
controller E1 1
!
controller E1 2
```

```
!
controller E1 3
!
gw-accounting h323
gw-accounting h323 vsa
gw-accounting voip
!
!
interface Ethernet0
 ip address 10.14.128.35 255.255.255.xxx
 no ip directed-broadcast
 h323-gateway voip interface
 h323-gateway voip id gk1 ipaddr 10.14.128.19 1xxx
 h323-gateway voip h323-id gw1@cisco.com
 h323-gateway voip tech-prefix 5#
!
interface Serial0:15
 no ip address
 no ip directed-broadcast
 isdn switch-type primary-net5
 isdn incoming-voice modem
 fair-queue 64 256 0
 no cdp enable
!
!
interface FastEthernet0
 ip address 10.0.0.1 255.255.xxx.0
 no ip directed-broadcast
 duplex full
 speed auto
 no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.14.128.33
ip route 10.14.xxx.0 255.xxx.255.xxx 16.0.0.2
ip route 10.14.xxx.16 255.xxx.255.240 10.14.xxx.33
no ip http server
!
!
radius-server host 10.14.132.2 auth-port 1645 acct-port 1646
radius-server key cisco
radius-server vsa send accounting
radius-server vsa send authentication
!
voice-port 0:D
 cptone DE
!
!
dial-peer voice 200 voip
 incoming called-number 53
 destination-pattern 34.....
 session target ipv4:10.0.0.2
 dtmf-relay h245-alphanumeric
 codec g711ulaw
!
dial-peer voice 102 pots
 service debit_card_rtsp
 incoming called-number 3450072
 shutdown
 destination-pattern 53.....
 port 0:D
!
!
dial-peer voice 202 voip
```

```
 shutdown
 destination-pattern 34.....
 session protocol sipv2
 session target ipv4:16.0.0.2
 dtmf-relay cisco-rtp
 codec g711ulaw
!
dial-peer voice 101 pots
 service debit_card
 incoming called-number 3450070
 destination-pattern 53.....
 port 0:D
!
!
gateway

!
line con 0
 exec-timeout 0 0
 transport input none
line aux 0
line vty 0 4
 password xxx
!
ntp clock-period 17180740
ntp server 10.14.42.23
end
```

GW2 IVR Configuration: Example

```
!
version 12.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname GW2
!
logging buffered 100000 debugging
aaa new-model
aaa authentication login default local group radius
aaa authentication login h323 group radius
aaa authentication login con none
aaa authorization exec h323 group radius
aaa accounting connection h323 start-stop group radius
!
username lab password xxx
username 111119 password xxx
!
!
resource-pool disable
!
!
!
!
!
clock timezone PST -8
ip subnet-zero
ip host radiusserver2 10.14.132.2
ip host radiusserver1 10.14.138.11
ip host baloo 10.14.124.254
```

```
ip host rtspserver2 10.14.136.2
ip host blue 223.255.254.254
ip host rtspserver3 10.14.126.2
!
mgcp package-capability trunk-package
mgcp default-package trunk-package
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
!
application
    service clid_authen_sky
tftp://blue/hostname/sky_scripts/clid_authen_collect_cli_sky.tcl
    !
    service rtsp_demo tftp://blue/hostname/sky_scripts/rtsp_demo.tcl
    !
    service debit_card tftp://blue/IVR 2.0/scripts.new/app_debitcard.tcl
    param debit_card uid-len 6
    paramspace english language en
    paramspace english index 1
    paramspace english location tftp://blue/hostname/WV/en_new/
    paramspace chinese language ch
    paramspace chinese index 2
    paramspace chinese location tftp://blue/hostname/WV/ch_new/
    !
    service clid_authen_rtsp
tftp://blue/IVR2.0/scripts.new/app_clid_authen_collect_cli_rtsp.tcl
    param location rtsp://rtspserver2:554/
    !
    service clid_authen1
tftp://blue/IVR2.0/scripts.new/app_clid_authen_collect_cli_rtsp.tcl
    param location tftp://blue/hostname/WV/en_new/
    param uid-len 6
    param retry-count 4
    !
    !
mta receive maximum-recipients 0
!
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 clock source line secondary 1
!
controller T1 2
!
controller T1 3
!
gw-accounting h323
gw-accounting h323 vsa
gw-accounting voip
!
!
interface Ethernet0
 ip address 10.14.xxx.4 255.255.xxx.240
 no ip directed-broadcast
 h323-gateway voip interface
 h323-gateway voip id gk2 ipaddr 10.14.xxx.18 1719
 h323-gateway voip h323-id gw2@cisco.com
 h323-gateway voip tech-prefix 3#
```

```
!
interface Serial0:23
 no ip address
 no ip directed-broadcast
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 fair-queue 64 256 0
 no cdp enable
!
interface FastEthernet0
 ip address 10.0.0.2 255.xxx.255.0
 no ip directed-broadcast
 duplex full
 speed 10
 no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.14.xxx.5
ip route 10.14.xxx.32 255.255.xxx.240 10.0.0.1
no ip http server
!
!
radius-server host 10.14.132.2 auth-port 1645 acct-port 1646
radius-server key cisco
radius-server vsa send accounting
radius-server vsa send authentication
!
voice-port 0:D
!
dial-peer voice 100 voip
 service debit_card
 incoming called-number 34
 shutdown
 destination-pattern 53.....
 session target ras
 dtmf-relay h245-alphanumeric
 codec g711ulaw
!
dial-peer voice 200 pots
 incoming called-number 30001
 destination-pattern 3450070
 port 0:D
 prefix 50070
!
dial-peer voice 101 voip
 service debit_card
 incoming called-number 34.....
 shutdown
 session protocol sipv2
 session target ipv4:10.0.0.1
 dtmf-relay cisco-rtp
 codec g711ulaw
!
dial-peer voice 102 voip
 incoming called-number 34.....
 destination-pattern 53.....
 session target ipv4:10.0.0.1
 dtmf-relay h245-alphanumeric
 codec g711ulaw
!
gateway
!
line con 0
 exec-timeout 0 0
```

```
 transport input none
line aux 0
line vty 0 4
 password xxx
!
ntp clock-period 17180933
ntp server 10.14.42.23
end
```

Inbound VoiceXML Application: Example

In this example, a VoiceXML application is associated with an inbound dial peer to process a call:

```
!
! This example shows how a VoiceXML application is associated with
! an inbound dial peer to process a call.
!
version 12.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname as5300-10
!
enable secret 5 $1$KRsb$ABCD
enable password lab
!
!
!
resource-pool disable
!
! Configure host IP addresses accessible from
! this gateway
!
ip subnet-zero
ip host vxml-server 10.10.1.1
ip host test 172.17.1.1
ip host project.cisco.com 10.10.1.1
ip host sample 172.17.1.1
ip dhcp smart-relay
!
! Configure ISDN
!
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
mta receive maximum-recipients 0
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
 clock source line secondary 1
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
controller T1 2
```

```
!
controller T1 3
!
!
!
interface Ethernet0
 ip address 10.10.1.2 255.255.0.0
 ip helper-address 172.17.1.1
 no ip route-cache
 no ip mroute-cache
!
interface Serial0:23
 no ip address
 ip mroute-cache
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 isdn disconnect-cause 1
 fair-queue 64 256 0
 no cdp enable
!
interface Serial1:23
 no ip address
 isdn switch-type primary-5ess
 no cdp enable
!
interface FastEthernet0
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
!
ip default-gateway 10.10.0.1
ip classless
ip route 172.17.1.0 255.255.255.0 10.10.0.1
no ip http server
!
access-list 101 permit ip any any
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
call rsvp-sync
!
! The service command specifies where the router can download
! the VoiceXML document "tr6". In this case, the document is accessible via
! an HTTP URL.
!
! The paramspace commands specify that English is the language
! for the tr6 application, and also specify the location of the audio files.
!
!
application
  service voice tr6 http://vxml-server/vxml_app.vxml
   paramspace english language en
   paramspace english index 1
   paramspace english location tftp://tftp-server/prompts/english/
!
! The next two lines show the maximum amount of memory for recording a single voice
! message has been set at 512 kilobytes and the maximum amount of memory for storing all
! recorded voice messages has been set at 5000 kilobytes.
!
ivr record memory session 512
```

```
ivr record memory system 5000
!
voice-port 0:D
!
voice-port 1:D
!
! The following dial peer specifies that incoming PSTN calls that
! match the number 5550154 should be handed to the application
! tr6 (vxml_app.vxml).
!
dial-peer voice 5550154 pots
 service tr6
 incoming called-number 5550154
 port 0:D
!
!
line con 0
 logging synchronous
 transport input none
line aux 0
line vty 0 4
 password lab
 login
!
scheduler interval 1000
end
```

Outbound VoiceXML Application with DNIS Map: Example

In this example a VoiceXML application is associated with an outbound dial peer through a DNIS map:

```
!
! This example shows how a VoiceXML application is associated with
! an outbound dial peer using a DNIS map
!
version 12.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname as5300-10
!
enable secret 5 $1$KRsb$c
enable password demo
!
!
resource-pool disable
!
! Configure host IP addresses accessible from this gateway
!
ip subnet-zero
ip host vxml-server 10.10.99.1
ip host test 172.17.1.1
ip host vxml.cisco.com 10.10.99.1
ip host jacks 172.17.1.1
ip dhcp smart-relay
!
! Configure ISDN
!
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
```

```
mta receive maximum-recipients 0
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
 clock source line secondary 1
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
controller T1 2
!
controller T1 3
!
!
interface Ethernet0
 ip address 10.10.115.90 255.255.0.0
 ip helper-address 172.17.1.1
 no ip route-cache
 no ip mroute-cache
!
interface Serial0:23
 no ip address
 ip mroute-cache
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 isdn disconnect-cause 1
 fair-queue 64 256 0
 no cdp enable
!
interface Serial1:23
 no ip address
 isdn switch-type primary-5ess
 no cdp enable
!
interface FastEthernet0
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
!
ip default-gateway 10.14.0.1
ip classless
ip route 172.17.1.1 255.255.255.0 10.10.0.1
no ip http server
!
access-list 101 permit ip any any
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
call rsvp-sync
!
! The service command specifies where the router can download
! the VoiceXML document "tr3". In this case, the document is accessible via
! an HTTP URL.
!
```

```
! The paramspace commands specify that English is the language for the tr3 application,
! and also specify the location of the audio files.
application
 service tr3 http://vxml-server/vxml_app.vxml
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://prompts/english/
!
!
! Specify a predefined VoiceXML DNIS map file stored on an TFTP server.
! Following is one method of using a DNIS map. It is typically a text file.
!
voice dnis-map dmap1 tftp://tftp-host/config-files/dm1.cfg
!
! Specify a VoiceXML DNIS map directly on the router
! This second method of using DNIS maps requires using Cisco IOS
! to create the DNIS map and populate it with DNIS entries.
!
voice dnis-map dmap2
  dnis 5550111 url http://http-host/vxml/app-for-5550111.vxml
  dnis 5550122 url http://http-host/vxml/app-for-5550122.vxml
!
voice-port 0:D
!
voice-port 1:D
!
!
dial-peer voice 5550154 pots
 service clid_authen_collect
 incoming called-number 5550154
 port 0:D
!
! If a call comes in on port 0:D, it will be handled by the
! clid_authen_collect application, which will authenticate, then
! collect digits. If the user then places a call to 555-0111,
! it will use dial peer 555 and interpret the document
! app-for-555-0111.vxml as specified in DNIS map dmap2.
!
dial-peer voice 555 voip
 dnis-map dmap2
 service tr3 out-bound
 session target ipv4:10.10.1.1

!
line con 0
 logging synchronous
 transport input none
line aux 0
line vty 0 4
 password lab
 login
!
scheduler interval 1000
end
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl IVR and VoiceXML Applications" on page 1.

- To configure voice recording using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward" on page 1.
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties" on page 1.
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML" on page 1.
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features" on page 1.
- To configure session interaction for a Tcl IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- "" on page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release
- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance
- *Cisco IOS Voice Command Reference, Release 12.4T* —Describes commands to configure and maintain Cisco IOS voice applications.

Configuring Audio File Properties for Tcl IVR and VoiceXML Applications

This chapter explains how to configure audio file properties for Tcl IVR and VoiceXML applications.

For more information about this and related Cisco IOS voice features, see the following:

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco IOS Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for Tcl IVR and VoiceXML Applications

This chapter includes information about configuring audio for different Tcl IVR and VoiceXML application features. For a feature history of all Tcl IVR and VoiceXML features, see "Cisco IOS Tcl IVR and VoiceXML Feature List" on page 2.

Contents

- Prerequisites for Audio Files, page 2
- Restrictions for Audio Files, page 2
- Information About Audio File Properties for Tcl and VoiceXML Applications, page 4
- How to Configure Audio File Properties for Applications, page 6
- Configuration Examples for Audio Files, page 16
- Where to Go Next, page 16
- Additional References, page 17

When developing and configuring a voice application, use this chapter and refer to the *Cisco VoiceXML Programmer's Guide* or the *Tcl IVR API Version 2.0 Programmer's Guide*.

Prerequisites for Audio Files

- You must configure basic VoiceXML application functionality as described in "How to Configure Basic Functionality for a Tcl IVR or VoiceXML Application" on page 24.
- The Cisco gateway must have connectivity to the media server where the audio files used by the Tcl or VoiceXML application are stored.
- In Cisco IOS Release 12.2(11)T, to play or record audio files using an RTSP server on a VoIP call leg, use one of the following configuration options on the originating gateway to ensure that packets are not dropped:

Note This prerequisite is not required for Cisco IOS Release 12(2)11T1 and later.

  - Two different physical Ethernet interfaces, one for the RTSP server and one for the VoIP interface to the terminating gateway.
  - If the RTSP server and the terminating gateway share the same physical Ethernet interface, you must configure the **no ip redirects** command for Ethernet interface 0/0 on the originating gateway.

Restrictions for Audio Files

The following sections list restrictions for audio recording and playback features:

- Recording and Playback Restrictions, page 2
- Codec Restrictions, page 3

Recording and Playback Restrictions

- Playback of audio recordings directly from an ESMTP server is not supported.
- For ESMTP recording, only a single mailbox address is supported in the mailto URL.
- Final silence detection, which lets the gateway terminate a recording after a defined length of silence, is not supported for RTSP recording. The final silence detection feature is disabled by default; it must be enabled by using the final silence property in the VoiceXML document.
- The **vad** command must be configured in the VoIP dial peer when final silence detection is needed to terminate a voice recording. When using speech recognition, however, the **no vad** command is required, so final silence is not detected for recording on IP call legs.
- RTSP multicast sessions are not supported by the Cisco IOS RTSP client.
- Full editing features (for example, seek, pause, rewind, append) are not supported.
- Maximum duration of a recording stored in local memory on the Cisco gateway is limited to the amount of available free memory. Memory limits can also be configured on the gateway by using the **ivr record memory session** and **ivr record memory system** commands.
- For RAM recordings submitted using HTTP POST:
  - Only the chunked transfer method is supported.
  - Recordings are sent using enctype of "multipart/form-data."

- The “audio/basic” enctype, which was supported in Cisco IOS Release 12.2(2)XB, is not supported in later releases of Cisco IOS software.
- For audio files streamed to an HTTP server:
  - Only the chunked transfer method is supported.
  - Audio files are sent using enctype of “multipart/form-data.”
- Volume control is supported for audio files played from memory or chunked transfer mode.
- Rate control is supported only for audio files played from memory or chunked transfer mode using the G.711 codec.
- The following restrictions apply to Cisco IOS Release 12.2(2)XB:
  - Audio recordings are stored in .au file format; .wav format is not supported.
  - Supported codecs for recording are G.711 u-law and G.723.1.
  - Supported codecs for audio playback are G.711 u-law, G.723.1, G.726 (Cisco AS5300), and G.729.
  - RTSP recording is not supported.
  - Rate control, volume control, and prompt timing information are not supported.
  - RAM recordings can be submitted to an external HTTP server using the POST method with an enctype of “audio/basic.”

Note A VoiceXML document written to support the “audio/basic” MIME type for <submit> in Cisco IOS Release 12.2(2)XB is not supported by later releases of Cisco IOS software. Later releases of Cisco IOS software support only the “multipart/form-data” type for <submit>.

Codec Restrictions

- Audio prompts played to an incoming VoIP call leg must use the same codec as the codec used for the VoIP call setup. If the codec of the audio file is different from the codec negotiated for the call, the audio prompt playout fails and an error is generated.
- Codec used for playing audio prompts must be used for the duration of the call, even after prompt playout is completed.
- For recording or playback over an IP call leg, the codec negotiated between the originating and terminating ends must match the codec specified in the VoiceXML document. If the codec of the audio file is different than the codec negotiated for the call, the recording or playback fails and an error is generated.
- The **voice-class codec** command is not supported in a dial peer that is configured with a VoiceXML application. Using the **voice-class codec** command results in a codec mismatch error when attempting a VoiceXML recording or prompt playout. You must use the **codec** command instead.
- GSM-EFR and G.728 codecs are not supported on the Cisco AS5350 or Cisco AS5400 for audio recording and playback.
- The GSMFR codec is not supported on the Cisco Integrated Services Routers (ISRs) for media recording.
- For a list of codecs that are supported for recording by platform and Cisco IOS release, see the “Codec Support for Audio Recording” section on page 5.

- Audio players might not recognize audio recordings made by the Cisco gateway using some codecs. The gateway can play back all audio files recorded by the gateway. For more information, see the "Audio File Formats Supported for Recording and Playback" section on page 5.

Information About Audio File Properties for Tcl and VoiceXML Applications

To configure audio files properties for Tcl and VoiceXML applications, you must understand the following concepts:

- Audio File Playout Methods, page 4
- Dynamic Prompts, page 5
- Volume and Rate Controls for Audio Prompts using VoiceXML, page 6

Audio File Playout Methods

Tcl and VoiceXML applications can be configured for incoming POTS or VoIP call legs to play announcements to the user and to request user input (digits). The gateway can also play back audio recordings made by VoiceXML applications. Audio files can be played toward both the PSTN side and the IP side of the call leg.

Tcl and VoiceXML applications can play out audio files by using the following playout methods from different locations:

- Memory
- HTTP, Flash, TFTP, or FTP Streamed
- RTSP Streamed
- TTS Streamed

Memory

The entire audio file is loaded into the gateway's memory and then played out to the appropriate call leg as needed. Memory-based prompts can be loaded from an HTTP server, or from Flash memory, a TFTP server, or an FTP server. Audio files can also be recorded into memory using VoiceXML recording capabilities, then played back from memory or submitted to an external HTTP server to become permanent audio files.

The amount of memory available to store audio prompts on the gateway can be configured by using the **ivr prompt memory** command.

Note Flash memory allows a limited number of entries, typically 32 on most platforms. For the specific Flash memory limits for your platform, refer to the platform-specific reference documentation listed in the "Additional References" section on page 5.

HTTP, Flash, TFTP, or FTP Streamed

The Cisco gateway can stream audio files from an external server to the appropriate call leg as needed. Loading an entire audio prompt into local memory before beginning playout can limit the length of audio prompts and impact memory resources. With streaming, pieces of a prompt are loaded into memory and then, if necessary, deleted after they are played to free up memory. The audio file is played out while it is being loaded into memory, with playback beginning as soon as a piece of the prompt is loaded.

Prompts can be streamed from an HTTP server or from Flash memory, a TFTP server, or an FTP server. With HTTP, each time a prompt is played, the HTTP caching system is checked, and the audio file is reloaded if necessary. The HTTP cached flag in the VoiceXML document specifies whether an audio file that is loaded into memory is safe to use again, and does not have to be deleted.

To enable the gateway to stream audio files during playout, use the **ivr prompt streamed** command. HTTP prompts are streamed by default, but HTTP streaming can be disabled by using the **no ivr prompt streamed http** command.

The amount of memory available to store audio prompts on the gateway can be configured by using the **ivr prompt memory** command. Performance is best when there is enough memory to store the entire audio file. If the **ivr prompt memory** command is set to a value smaller than the size of a streamed file, performance is not as good.

RTSP Streamed

An external Real Time Streaming Protocol (RTSP) server can stream audio to the appropriate call leg as needed. RTSP is an application-level protocol that controls the on-demand delivery of real-time data, such as the delivery of audio streams from an audio server. By implementing an RTSP client on the Cisco VoIP gateway, a voice application running on the gateway can connect calls with audio streams from an external RTSP server. Prompts from RTSP servers are always streamed during playback. RTSP saves memory on the gateway because it is packet-based. Unlike HTTP or TFTP streaming, for example, RTSP streaming does not read any part of the audio file into RAM.

Note When playing a series of short audio prompts, such as with dynamic prompts, nonstreaming might be more efficient; streaming playout can cause noticeable delays and impact voice quality.

TTS Streamed

An external speech synthesizer using MRCP can generate prompts. Requests to synthesize speech from text strings or audio segments are sent to the media server, which responds with a real-time audio stream.

Dynamic Prompts

Dynamic prompts are formed by the underlying system assembling small audio files and playing them out in sequence. This provides simple TTS operations, like playing numbers, dollar amounts, dates, and time. For example, dynamic prompts can inform the caller of how much time is left in their debit account, as in:

> "You have 15 minutes and 32 seconds of call time left in your account."

This prompt is created using eight individual audio files. They are: youhave.au, 15.au, minutes.au, and.au, 30.au, 2.au, seconds.au, and leftinyouraccount.au. These audio files are assembled dynamically by the underlying system and played out as a single prompt.

The language and location of the audio files used for dynamic prompts can be specified in the Tcl script or VoiceXML document, or these parameters can be configured on the Cisco gateway by using the **service** and **package** commands in the application configuration.

Note When playing a series of short audio prompts, such as with dynamic prompts, non-streaming might be more efficient; streaming playout can cause noticeable delays and impact voice quality.

Tcl Language Modules for Dynamic Prompts

Each language uses a Tcl language module. The Tcl language module defines the list of TTS notations that the language supports. Cisco IOS software includes built-in language modules for Chinese, English, and Spanish. You can add support for new languages and new TTS notations by configuring a new Tcl language module on the gateway.

The Cisco IOS infrastructure interfaces with the Tcl language module to translate TTS notations supplied by the voice application into the specified language. Cisco IOS software translates TTS notations into the sequence of audio files according to the language structure. For example, English and French use different sequences for saying the date: the English language structure says the month first and then the day; the French language structure says the day first and then the month.

Note Language modules are not used by external TTS servers; they are used by Cisco IOS software to assemble a list of dynamic prompts.

New TTS notations for the Cisco IOS built-in languages, such as playing dates and times of day, can also be configured. For example, if you configure a new English Tcl language module, it overrides the built-in English Tcl language module during the translation. When completed, any voice application can use the new notations, and the Cisco IOS infrastructure recognizes and plays the audio accordingly.

Note Tcl language modules are not Tcl IVR scripts. They are pure Tcl scripts and any system on the Cisco gateway (Tcl IVR 1.0, 2.0, VoiceXML, MGCP) can use the configured language with little or no change to the Cisco IOS configuration.

For information on writing a new Tcl language module, refer to the *Cisco Pre-Paid Debitcard Multi-Language Programmer's Reference*.

For information on configuring a new language module on the gateway, see the "Specifying a New Language Module for Dynamic Prompts" section on page 7.

Volume and Rate Controls for Audio Prompts using VoiceXML

The volume of audio prompts can be adjusted during playback. Audio prompts that are played out from memory or through chunked transfer mode using the G.711 codec can also be sped up or slowed down. A VoiceXML variable contains the rate and duration of the last prompt that was played.

The rate and volume of prompts is controlled by using Cisco attributes in the VoiceXML document. For detailed information, refer to the *Cisco VoiceXML Programmer's Guide*.

How to Configure Audio File Properties for Applications

- Specifying a New Language Module for Dynamic Prompts, page 7
- Setting Language and Location of Audio Files for Dynamic Prompts, page 8
- Setting Memory Recording Limits, page 11 (optional)

- Verifying Prompt Playout, page 12 (optional)
- Configuring Audio Prompt Streaming, page 13 (optional)
- Modifying Codec Complexity on the Cisco 3600 Series, page 13 (optional)

Specifying a New Language Module for Dynamic Prompts

Cisco includes built-in language modules for Chinese, English, and Spanish. This section explains how to add support for new languages and new text-to-speech (TTS) notations by configuring a new Tcl language module on the gateway.

Note Before configuring a new language module on the gateway, you must first obtain or write a new Tcl language module and install it on a server that is accessible to the gateway. For information, see the *Cisco Pre-Paid Debitcard Multi-Language Programmer's Reference*.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **package** *package-name location*
5. **param** *parameter-name parameter-value*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
    enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
    application

Example: Router(config)# application
```

Step 4 Enter service parameter configuration mode:

```
    package package-name location

Example: Router(config-app)# package french
tftp://server-1/tftpboot/scripts/fr_translate.tcl
```

Step 5 Configure the parameter's value:

```
    param parameter-name parameter-value

Example: Router(config-app-param)# param language fr
    param prefix fr
```

Verifying Configured Languages

To verify configured languages, perform the following steps.

SUMMARY STEPS

1. **show call application voice** *language*

DETAILED STEPS

Step 1 Use the **show call language voice** command to display information about a specific language.

This example shows parts of a Russian (*ru*) Tcl language module.

```
Router# show call language voice ru
Script Name : russian
       URL  : builtin:package_russian.C
       Type : Package
       State: Registered
       Life : Builtin
       Exec Instances: 10

     Parameters registered under russian namespace:
     name                 type  default value   description
     location             S     flash:          The URI of the audio files for this
language
     prefix               S     ru              The prefix of this language
     language             S     ru              The language code of this language
     index                S     -1              The index of this language

   Script Code Begin:
   --------------------------------

   Built in C Generic language translation package for dynamic prompt

   --------------------------------
```

Troubleshooting Tips

If language configuration is not successful, verify the following:

- The language package you are using is loaded.
- The parameters you are configuring are contained in the language package.

Setting Language and Location of Audio Files for Dynamic Prompts

The language and location of audio files that are used for dynamic prompts can be configured on the gateway or these parameters can be specified through properties in the VoiceXML document or Tcl script. For information on specifying the language and location of dynamic prompts by using

VoiceXML or Tcl properties, refer to the *Cisco VoiceXML Programmer's Guide* or *Tcl IVR API Version 2.0 Programmer's Guide*, respectively. This section explains how to configure the language and location of dynamic prompts through the gateway.

Note Specifying the language and location of dynamic prompts in a VoiceXML document or Tcl script takes precedence over the Cisco gateway configuration. Any value that is configured on the gateway is ignored if the same attribute is specified using a VoiceXML or Tcl property.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. service *service-name*
5. **paramspace** *language* **language** *prefix*
6. **paramspace** *language* **index** *number*
7. **paramspace** *language* **location** *location*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode:

```
application

Example: Router(config)# application
```

Step 4 Enter the parameter configuration mode for this application:

```
service service-name
```

Step 5 Specify the language prefix and index number of the audio files that an application uses for dynamic prompts:

```
application
 service service-name
  paramspace language language prefix
  paramspace language index number
```

- *service-name*—Name of the Tcl or VoiceXML application.
- *language*—Name of the language package being configured. There are three built-in language packages: Chinese, English, and Spanish. Other languages may be supported by use a of Tcl language script.
- *number*—Number that identifies the language of the audio files used for dynamic prompts. (This number has no significance in VoiceXML applications. Enter any number.)

- *prefix*—Two-character code for the language:
  - Chinese: ch
  - English: en (default)
  - Spanish: sp
  - all three: aa

```
Example:
Router(config)# application
Router(config-app)# service vapp1
Router(config-app-param)# paramspace english language en
Router(config-app-param)# paramspace english index 1
```

Note When configuring the language using the **index** command, keep in mind that the software is hardcoded with digit 1 to represent the primary language and digit 2 to represent the secondary language.

Step 6 Specify the location of the audio files that an application uses for dynamic prompts:

```
application
  service service-name
    paramspace language location location
```

- *service-name*—Name of the Tcl or VoiceXML application.
- *language*—Name of the language package.
- *location*—URL of the directory that contains the language audio files used by the application, without filenames. Flash memory (flash) or a directory on a server (TFTP, HTTP, or RTSP) are all valid.

```
Example:
Router(config)# application
Router(config-app)# service vapp1
Router(config-app-param)# paramspace english location flash
```

This command specifies the location of the language audio files that are used for the dynamic prompts specified in Step 1. Use this command multiple times for a single application to allow up to four subdirectories for audio files.

The following example shows the creation of two subdirectories:

```
Router(config)# application
Router(config-app)# service vapp1
Router(config-app-param)# paramspace english location http://aufiles/en_dir/
Router(config-app-param)# paramspace spanish location http://aufiles/sp_dir/
```

Verifying Language and Location of Audio Files for Dynamic Prompts

Use the **show running-config** command to see if the audio file language and location are properly configured with the **application** commands, for example:

```
Router# show running-config
!
```

```
application
 service vapp1 flash:demo0.vxml
  paramspace english language en
  paramspace english location flash
!
```

Setting Memory Recording Limits

The Cisco IOS VoiceXML feature supports the VoiceXML 1.0 <record> element for recording speech to a choice of four destinations, including local memory on the Cisco gateway. You can set limits on the amount of memory allocated for audio files recorded to local memory. This section explains how to modify the maximum memory limit for a single recording session (call) or for all recording sessions combined.

Note This procedure only configures memory limits for recordings to local memory on the Cisco gateway. Recording limits are not configurable on the gateway for HTTP, RTSP, or SMTP recordings.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ivr record memory session** *kilobytes*
4. **ivr record memory system** *kilobytes*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

`Example: Router# configure terminal`

Step 3 Set the maximum amount of memory that can be used for recording in a single VoiceXML interpreter session:

ivr record memory session *kilobytes*

- *kilobytes*—Memory size in kilobytes. Range is from 0 to 256,000. The default is 256.

`Example: Router(config)# ivr record memory session 1000`

This example sets the maximum recording memory used by a single session to 1 MB of memory. For G.711 recordings, which use 8 KB/sec, this allows 125 sec of recording to be stored for each session.

Step 4 Set the maximum amount of memory that can be used to store all audio files:

ivr record memory system *kilobytes*

- *kilobytes*—Memory size in kilobytes. Range is from 0 to 256,000. If 0 is configured, the recording function is disabled on the gateway. The default memory size is platform-specific:
  - Cisco 3640 and Cisco AS5300: 10,000 KB
  - Cisco 3660, Cisco AS5350, and Cisco AS5400: 20,000 KB

```
Example: Router(config)# ivr record memory system 15000
```

Troubleshooting Tips

Table 4-1 lists possible causes for audio recording and playout faiing and suggested actions.

Table 4-1 Audio Recording or Playout Fails

Possible Causes	Suggested Actions
VCWare version is not supported (Cisco AS5300 only).	Use the **show vfc version** command to verify that the router is using VCWare version 10.25 or later.
Codec of the audio file on an incoming IP call leg is different than the codec negotiated for the call.	Compare the codec specified in the VoiceXML document with the codec configured in the dial peer on the gateway. If there is a codec mismatch, the audio playout fails and an error is generated.
Audio files recorded with different codecs are included in the same <prompt> elements in the VoiceXML document.	Verify that audio files recorded with different codecs use different <prompt> elements in the VoiceXML document.
RTSP server and VoIP interface share the same Ethernet interface.	See the "Prerequisites for Audio Files" section on page 2.
If using a third-party media tool, the media tool does not support the format of recordings made by the Cisco gateway.	Playback the recording from the gateway or see the "Correction Utility for Audio File Headers" section on page 7.

Verifying Prompt Playout

SUMMARY STEPS

1. **show vfc** *slot* **version vcware**
2. Verify that audio prompts played to an incoming IP call leg use the same codec as the codec used for the VoIP call setup.
3. Verify that audio files recorded with different codecs use different <prompt> elements in the VoiceXML document.

DETAILED STEPS

Step 1 If playing prompts from the Cisco AS5300, verify that the router is using VCWare version 10.25 or later, by using the **show vfc version** command, for example:

```
Router# show vfc 2 version vcware

Voice Feature Card in Slot 2:
     VCware Version     :  10.25
     ROM Monitor Version:  1.2
     DSPware Version    :  1.0
     Technology         :  C542
```

Step 2 Verify that audio prompts played to an incoming IP call leg use the same codec as the codec used for the VoIP call setup. If the codec of the audio file is different than the codec negotiated for the call, the audio prompt playout fails and an error is generated.

Step 3 Verify that audio files recorded with different codecs use different <prompt> elements in the VoiceXML document.

Configuring Audio Prompt Streaming

Audio prompts can be streamed during playback to help free up memory resources. With streaming, smaller pieces of a prompt are loaded into memory and then deleted as they are used, instead of loading the entire prompt into memory before beginning playout. For more information, see the "Audio File Playout Methods" section on page 4.

The following command specifies whether audio prompts from selected media types are streamed during playback:

ivr prompt streamed {**all** | **flash** | **http** | **none** | **tftp**}

- **all**—Audio prompts from all URL types (HTTP, TFTP, Flash memory) are streamed during playback.
- **flash**—Audio prompts from flash memory are streamed during playback.
- **http**—Audio prompts from an HTTP URL are streamed during playback. This is the default value.
- **none**—No audio prompts (HTTP, TFTP, Flash memory) from any media type are streamed during playback.
- **tftp**—Audio prompts from an TFTP URL are streamed during playback.

```
Example: Router(config)# ivr prompt streamed flash
```

Configuring RTSP Live Streaming

RTSP live streaming is supported in Cisco IOS Release 15.0(1)M and later releases. To stream prompts from an RTSP server, use the **rtsp client timeout connect** command to set the number of seconds allowed for the router to establish a TCP connection to an RTSP server. With Cisco IOS Release 15.0(1)M and later releases, use the *cisco-maxtime* attribute of the <prompt> element to control the playout time for RTSP live streaming. If *cisco-maxtime* is zero or has no value set, the RTSP stream is played indefinitely.

Modifying Codec Complexity on the Cisco 3600 Series

Codec complexity affects the number of calls that can take place on the digital signal processor (DSP) interfaces. The greater the codec complexity, the fewer the calls that can be handled. Codec complexity is either medium or high. The default is medium. All medium-complexity codecs can also run in high-complexity mode, but fewer (usually half as many) channels are available per DSP. The value configured for codec complexity determines the choice of codecs that are available in the dial peers.

Codec complexity also determines whether the Cisco 3600 series supports a separate media stream for speech recognition, enabling the gateway to simultaneously perform speech synthesis or play audio files using a different codec. A separate RTP stream for speech recognition is supported on the Cisco 3660 only when the codec complexity is set to high. It is not supported for medium complexity codecs.

This section explains how to modify the codec complexity on the Cisco 3600 series.

For details on the number of calls that can be handled simultaneously using each of the codec standards, refer to the following resources:

- *Cisco IOS Release 12.2 Cross-Platform Release Notes*
- *Digital T1/E1 Packet Trunk Network Module* data sheet
- *Cisco IOS Voice Command Reference, Release 12.3 T*, entries for the **codec** and **codec complexity** commands.

Note On Cisco 3600 series routers with digital T1/E1 packet voice trunk network modules (NM-HDV), codec complexity cannot be configured if DS0 groups are configured. If no DS0 groups are configured, you can skip Steps 6 through 8 in the following procedure.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice-port** *slot/port:ds0-group-no*
4. **shutdown**
5. **exit**
6. **controller** {**t1** | **e1**} *slot/port*
7. **no ds0-group** *ds0-group-no* **timeslots** *timeslot-list* **type** *type*
8. **exit**
9. **voice-card** *slot*
10. **codec complexity** {**high** | **medium**}
11. **exit**
12. Repeat Step 6, then continue with step 13.
13. **ds0-group** *ds0-group-no* **timeslots** *timeslot-list* **type** *type*
14. **exit**
15. Repeat Step 3, then continue with step 16.
16. **no shutdown**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Enter voice-port configuration mode:

```
voice-port slot/port:ds0-group-no
```

- *slot*—Backplane slot number where the voice module is installed. Valid entries are platform dependant, in the range of 0 to 6. Refer to the *Software Configuration Guide for Cisco 2600 Series, Cisco 3600 Series, and Cisco 3700 Series Routers* for more information.
- *port*—Voice interface card location. Valid entries are 0 or 1.
- *ds0-group-no*— Identifies the DS0 group. Range is from 0 to 23 for T1, 0 to 30 for E1.

`Example: Router(config)# voice-port 0/1:23`

Step 4 Deactivate the voice port:

`shutdown`

`Example: Router(config-voiceport)# shutdown`

Step 5 Exit voice port configuration mode and return to global configuration mode:

`exit`

`Example: Router(config-voiceport)# exit`

Step 6 Enter controller configuration mode for the selected T1 or E1 controller:

`controller` `{`**`t1`** `|` **`e1`**`}` *`slot/port`*

- *slot*—Backplane slot number of the interface. Valid entries are platform-dependant, in the range of 0 to 6. Refer to the *Software Configuration Guide for Cisco 2600 Series, Cisco 3600 Series, and Cisco 3700 Series Routers* for more information.
- *port*—Backplane port number of the interface. Valid entries are 0 or 1.

`Example: Router(config)# controller t1 0/1`

Step 7 Remove the related DS0 groups:

`no ds0-group` *`ds0-group-no`*

- *ds0-group-no*— Identifies the DS0 group. Range is from 0 to 23 for T1, 0 to 30 for E1.

`Example: Router(config-controller)# no ds0-group 0 timeslots 1-24 type e&m-wink-start`

Step 8 Exit controller configuration mode and return to global configuration mode:

`exit`

`Example: Router(config-controller)# exit`

Step 9 Enter voice card configuration mode for the card or cards in the slot specified:

`voice-card` *`slot`*

- *slot*—Slot number of the voice card. Valid entries are platform dependant, in the range of 0 to 6. Refer to the *Software Configuration Guide for Cisco 2600 Series, Cisco 3600 Series, and Cisco 3700 Series Routers* for more information.

`Example: Router(config)# voice-card slot`

Note If any DSP voice channels are in the busy state, codec complexity cannot be changed. You can use the **show voice dsp** command to check the DSP voice channel activity.

Step 10 Specify codec complexity based on the codec standard:

`codec complexity` `{`**`high`** `|` **`medium`**`}`

Note If two WAN interface cards are installed, this command configures both cards at once.

- **high**—(Optional) Supports up to six voice or fax calls per DSP module (PVDM-12), using the codecs: G.723, G.728, G.729, G.729 Annex B, fax relay, or any of the medium complexity codecs.
- **medium**—Supports up to 12 voice or fax calls per DSP module (PVDM-12), using the codecs: G.711, G.726, G.729 Annex A, G.729 Annex A with Annex B, and fax relay. The default is **medium**.

This setting restricts the codecs available in dial peer configuration. All voice cards in a gateway must use the same codec complexity setting.

`Example: Router(config-voice-card)# codec complexity high`

Step 11 Exit voice card configuration mode and return to global configuration mode:

`exit`

`Example: Router(config-voice-card)# exit`

Step 12 Repeat Step 6, then continue with Step 13.

Step 13 Add the related DS0 groups:

`ds0-group` *`ds0-group-no`* **`timeslots`** *`timeslot-list`* **`type`** *`type`*

- *ds0-group-no*— Identifies the DS0 group. Range is from 0 to 23 for T1, 0 to 30 for E1.
- *timeslot-list*—Single time slot number, single range of timeslot numbers, or multiple ranges of timeslot numbers separated by commas. Range is from 1 to 24 for T1, 1 to 31 for E1 (time slot 16 is reserved for signaling).
- *type*—Signaling type of the telephony connection.

`Example: Router(config-controller)# ds0-group 0 timeslots 1-24 type e&m-wink-start`

Step 14 Exit controller configuration mode and return to global configuration mode:

`exit`

`Example: Router(config-controller)# exit`

Step 15 Repeat Step 3, then continue with Step 16.

Step 16 Activate the voice port:

`no shutdown`

`Example: Router(config-voiceport)# no shutdown`

Configuration Examples for Audio Files

Configuration examples of Cisco Tcl and VoiceXML applications are in the "Configuration Examples for Tcl IVR and VoiceXML Applications" section on page 71.

Where to Go Next

- To configure voice recording using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward" on page 1.
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties" on page 1.
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML" on page 1.
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features" on page 1.
- To configure session interaction for a Tcl IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- “” on page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release
- “Overview of Cisco IOS Tcl IVR and VoiceXML Applications” on page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance

Configuring VoiceXML Voice Store and Forward

The VoiceXML Voice Store and Forward feature allows streaming-based voice recording and playback features for various media including local memory, HTTP, ESMTP, and RTSP for 14 different Cisco codecs and two standard audio file formats, .au and .wav. This chapter explains how to configure the VoiceXML Store and Forward feature.

For more information about this and related Cisco IOS voice features, see the following:

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for VoiceXML Voice Store and Forward

Release	Modification
12.2(11)T	This feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.

Contents

- Prerequisites for VoiceXML Voice Store and Forward, page 2
- Restrictions for VoiceXML Voice Store and Forward, page 2
- Information About VoiceXML Store and Forward, page 3
- How to Configure VoiceXML Voice Store and Forward, page 8
- Where to Go Next, page 32
- Additional References, page 32

When developing and configuring a voice application, refer to this chapter and to the *Cisco VoiceXML Programmer's Guide* or the *Tcl IVR API Version 2.0 Programmer's Guide*.

Prerequisites for VoiceXML Voice Store and Forward

- You must configure basic VoiceXML application functionality as described in "Configuring Basic Functionality for Tcl IVR and VoiceXML Applications" on page 1.
- You must write a VoiceXML 2.0 document that implements voice store and forward features. To write your own script, refer to the *Cisco VoiceXML Programmer's Guide*.

Restrictions for VoiceXML Voice Store and Forward

The following sections list restrictions for the VoiceXML Voice Store and Forward feature:

Recording and Playback Restrictions

- Playback of audio recordings directly from an ESMTP server is not supported.
- For ESMTP recording, only a single mailbox address is supported in the mailto URL.
- Final silence detection, which lets the gateway terminate a recording after a defined length of silence, is not supported for RTSP recording. The final silence detection feature is disabled by default; it must be enabled by using the final silence property in the VoiceXML document.
- The **vad** command must be configured in the VoIP dial peer when final silence detection is needed to terminate a voice recording. When using speech recognition, however, the **no vad** command is required, so final silence is not detected for recording on IP call legs.
- Maximum duration of a recording stored in local memory on the Cisco gateway is limited to the amount of available free memory. Memory limits can also be configured on the gateway by using the **ivr record memory session** and **ivr record memory system** commands.
- For RAM recordings submitted using HTTP POST:
  - Only the chunked transfer method is supported.
  - Recordings are sent using enctype of "multipart/form-data."
  - The "audio/basic" enctype, which was supported in Cisco IOS Release 12.2(2)XB, is not supported in later releases of Cisco IOS software.
- For audio files streamed to an HTTP server:
  - Only the chunked transfer method is supported.
  - Audio files are sent using enctype of "multipart/form-data."

Audio File Data-Length Restrictions

Because the recording duration is unknown at the beginning of an HTTP or ESMTP streaming session, the .au and .wav file headers generated by the Cisco gateway are encoded with a data length of 0 when recording to an HTTP or ESMTP server; the gateway correctly decodes 0-length files during playback.

The standard .wav format, however, does not allow for a 0-length in the header, so third-party audio players may not be able to play back .wav files encoded by the gateway. Although the .au format allows files with a 0-length, some audio players may not support these files either.

To enable an audio player to play back recordings made by the Cisco gateway to an HTTP or ESMTP server, you must use a correction utility to modify fields in the audio file headers, as described in the "Correction Utility for Audio File Headers" section on page 7.

The following additional restrictions apply:

- Recordings made to an RTSP server, or to local memory and submitted to HTTP, use the correct data length in the headers and do not require a correction utility.
- The Cool Edit audio player recognizes 0-length .au and .wav files recorded by the gateway using G.711 u-law so no changes to those files are required. Use the correction utility to enable the Cool Edit audio player to play back .au and .wav files that use G.711 a-law.
- Windows Media Player and Vovida RTSP player require the correction utility to playback .au and .wav files using G.711 u-law, or .wav files using G.711 a-law.
- Audio files recorded by third-party audio players might not completely follow the .au or .wav file format standards. Cisco reserves the right not to support non-standard .au or .wav file formats.
- Concatenating two audio files without first correcting the data length fields and the data offsets can result in files with invalid data lengths. Cisco reserves the right not to support invalid data-length audio files.

Information About VoiceXML Store and Forward

To configure audio files properties for Tcl and VoiceXML applications, you must understand the following concepts:

- VoiceXML Voice Store and Forward, page 3
- VoiceXML Audio Recording Scenario, page 4
- Audio File Formats Supported for Recording and Playback, page 5
- VoiceXML Recording Locations, page 5
- Codec Support for Audio Recording, page 5
- Codec Mappings in Audio Recordings, page 7
- Correction Utility for Audio File Headers, page 7

VoiceXML Voice Store and Forward

The VoiceXML Voice Store and Forward feature expands Cisco IOS VoiceXML to include the input and processing of form field entries using recorded audio clips, rather than numeric input only. Audio clips can be captured and then submitted to an external web server using HTTP or RTSP, or to a messaging server using Extended Simple Mail Transfer Protocol (ESMTP) for additional processing.

This recording feature can be used to collect caller names or addresses for call screening, product registration, or similar e-commerce applications, and for simple voice messaging, or for any voice browser application where alphanumeric input using DTMF is cumbersome or impractical.

The VoiceXML Voice Store and Forward feature supports speech recording and playback with a choice of four different media locations, including:

- Local memory—Voice recordings are stored in local memory on the Cisco gateway, and can be played back or submitted to an HTTP server using the POST method. Intended for temporarily storing short-length speech clips, such as caller name or address, or a short voice message.

- HTTP—Voice recording is streamed directly to an external HTTP server using the URL specified by the user. Recording can be played back in streaming or non-streaming mode.
- RTSP—Voice recording is directly streamed to and from an external RTSP server using the URL specified by the user. Intended for storing indefinite-length audio recordings.
- ESMTP—Voice recording is directly streamed to the ESMTP server as e-mail audio attachments. This option supports the Mailto: URL.

The Voice Store and Forward feature enables dynamic voice messaging by switching a busy or no-answer voice call to a VoiceXML application. The voice gateway can operate in two modes:

- On-ramp mode—Incoming calls are handled by a VoiceXML document that lets callers record voice messages if the called party is busy or there is no answer. The on-ramp gateway stores the voice recordings as audio clips to the selected media location; an external HTTP or RTSP server, internal memory if space is available, or by directly streaming the voice message as an e-mail attachment to an external ESMTP server.

  To configure the gateway to record and play back voice messages, see the "Configuring the On-Ramp Gateway for VoiceXML Voice Store and Forward" section on page 8.
- Off-ramp mode—An external mail server sends an e-mail notification to the off-ramp gateway. The off-ramp gateway extracts the dialed number in the e-mail header and places an outbound call to the corresponding PSTN or IP destination. When the call is answered, the gateway executes the configured VoiceXML application. The VoiceXML application retrieves the audio clip from the external media server and plays the message to the PSTN or IP destination. The gateway does not support the streaming of audio clips directly from the ESMTP server.

  To configure the gateway to receive e-mail triggers and make outbound calls to deliver voice messages, see the "Configuring the Off-Ramp Gateway to Place a Call" section on page 16.

VoiceXML Audio Recording Scenario

The following is an example call scenario for a VoiceXML application using recording capabilities:

1. The caller dials a number and is connected through the PSTN or the IP network to a Cisco voice gateway that is configured as a VoiceXML-enabled gateway.
2. The Cisco voice gateway associates the dialed number with the appropriate VoiceXML document, residing on a web server.
3. The voice gateway runs the VoiceXML document and responds to the caller's input by playing the appropriate audio content.
4. The gateway executes the document, which prompts the user to record a voice message.
5. The message is recorded by the gateway and stored in local memory with the selected audio encoding.
6. After the recording is completed, the user can review the message or submit it by either pressing a specified key or hanging up the call.
7. When the user submits the voice mail message, the gateway's VoiceXML browser submits the voice message in .au or .wav file format to a specified URL using the HTTP POST method.
8. After receiving the message, the web server can store the message in the appropriate mailbox.
9. After successfully storing the voice message, the application instructs the media stream process to delete the local copy of the voice message.

Audio File Formats Supported for Recording and Playback

Cisco VoiceXML gateways support two standard audio formats for recording and playback: .au (audio/basic) and .wav (audio/wav). The format used to record an audio file is specified by the VoiceXML document at the time of the recording. If it is not defined in the VoiceXML document, the default format type is audio/basic. The gateway uses standard codec numbers for the codec format in the audio header whenever possible, including G.711 u-law, G.711 a-law, and G.726 (32k ADPCM); other codecs use a proprietary format mapping and may therefore not be recognized by third-party audio players. For a listing of the codec numbers used by the Cisco gateway, see the "Codec Mappings in Audio Recordings" section on page 7.

Recordings made by the gateway are open-ended and real-time streaming, so the .au and .wav file headers generated by the Cisco gateway are encoded with a data length of 0. All audio files recorded by the Cisco gateway can be played back by the gateway, however, because the standard .wav format does not allow for a 0 length in the header, some third-party audio players may not be able to play back .wav files encoded by the gateway. The .au file format allows a 0 length, but some audio players may also not support these files.

For example, the Cool Edit audio player can play 0-length .au and .wav files that are recorded by the gateway using the G.711 u-law codec. Other audio players and other codecs require the audio file header to be modified before playback. For example, the Windows Media Player audio player can play G.711 u-law and G.711 a-law audio files if the 0-length field is modified. The Cool Edit audio player can play G.711 a-law files if the header is corrected.

To enable an audio player to play back recordings made by the Cisco gateway, you must use a correction utility to modify specific fields in the audio file headers. For information about how to modify the file headers, see the "Correction Utility for Audio File Headers" section on page 7.

VoiceXML Recording Locations

The VoiceXML Voice Store and Forward feature supports audio recording and playback using local memory on the Cisco gateway or a choice of external media server locations:

- Local memory—Recording and playback is supported for storing and retrieving audio files. Audio recordings can also be submitted to HTTP servers for permanent storage.
- ESMTP—Recording is supported by directly streaming audio to ESMTP server as e-mail attachment. Playout directly from the ESMTP server is not supported.
- HTTP—Recording is supported by directly streaming audio to HTTP server using the chunked transfer-encoding method; playback is supported using streaming and non-streaming methods.
- RTSP—Recording and playout is supported by directly streaming audio to and from an RTSP server.
- TFTP—Playout supported by retrieving audio file from a TFTP server, using streaming or non-streaming methods. Recording audio to a TFTP server is not supported.

The URL of the recording destination is specified in the VoiceXML document by using the Cisco property cisco-dest. For more information, refer to the *Cisco VoiceXML Programmer's Guide*.

Codec Support for Audio Recording

Table 5-1 shows the codecs that are supported for audio recording and playback, by platform and minimum required Cisco IOS release.

Note All codecs listed are supported for H.323 and SIP unless otherwise noted.

The GSMFR codec is not supported on the Cisco Integrated Services Routers (ISRs) for media recording.

Table 5-1 Codec Support for Audio Recording by Platform and Minimum Cisco IOS Release

		Cisco IOS Release			
Codec	**Cisco IOS Value**	**Cisco 3600 Series**	**Cisco AS5300**	**Cisco AS5350**	**Cisco AS5400**
G.711 a-law	g711alaw	12.2(11)T	12.2(11)T	12.2(11)T	12.2(11)T
G.711 u-law	g711ulaw	12.2(11)T	12.2(2)XB	12.2(2)XB	12.2(2)XB
G.723.1 Annex-A (5.3 kbps)	g723ar53	12.2(11)T	12.2(11)T	12.2(11)T	12.2(11)T
G.723.1 Annex-A (6.3 kbps)	g723ar63	12.2(11)T	12.2(11)T	12.2(11)T	12.2(11)T
G.723.1 (5.3 kbps)	g723r53	12.2(11)T	12.2(2)XB[1]	12.2(2)XB[1]	12.2(2)XB[1]
G.723.1 (6.3 kbps)	g723r63	12.2(11)T	12.2(2)XB	12.2(2)XB[1]	12.2(2)XB[1]
G.726 (16 kbps)[2]	g726r16	12.2(11)T	12.2(11)T[3]	12.2(11)T[3]	12.2(11)T[3]
G.726 (24 kbps)[2]	g726r24	12.2(11)T	12.2(11)T	12.2(11)T[3]	12.2(11)T[3]
G.726 (32 kbps)[2]	g726r32	12.2(11)T	12.2(11)T	12.2(11)T[3]	12.2(11)T[3]
G.728 (16 kbps)	g728	12.2(11)T	12.2(11)T	Not supported	Not supported
G.729 (8 kbps)[2]	g729r8 (high complexity)	12.2(11)T	12.2(11)T	12.2(11)T	12.2(11)T
G.729 Annex-A (8 kbps)	g729r8 (medium complexity)	12.2(11)T	Not supported	Not supported	Not supported
G.729 Annex-B (8 kbps)[2]	g729br8 (high complexity)	12.2(11)T	12.2(11)T	12.2(11)T	12.2(11)T
G.729A Annex-B (8 kbps)	g729br8 (medium complexity)	12.2(11)T	Not supported	Not supported	Not supported

1. This codec is supported only for H.323 on this platform in Cisco IOS Release 12.2(2)XB; it is supported for SIP in Cisco IOS Release 12.2(11)T.
2. This codec is supported only for audio playback in Cisco IOS Release 12.2(2)XB.
3. This codec is not supported for SIP on this platform.

Note

- If the codec for an audio recording is not specified in the VoiceXML document, the default codec used for the recording is G.711 u-law.
- For recording and playback over an IP call leg, the negotiated codec must match the codec specified in the VoiceXML document. If the codec specified for the audio file is different than the codec negotiated for the call, the recording or playback fails and an error is generated.
- To determine specific codec support for your platform and Cisco IOS release, use the **codec** command in dial-peer configuration mode.

Codec Mappings in Audio Recordings

Table 5-2 lists the codec number that is mapped to each codec in the audio file header, for .au format and .wav format files.

Table 5-2 Codec Numbers Used in Audio File Headers

.AU Files		.WAV Files	
Number	**Codec**	**Number**	**Codec**
1	PCMULAW	1	MS_PCM
23	32K_ADPCM	2	MS_ADPCM
27	PCMALAW	6	G711_ALAW
39	CISCO_G729	7	G711_MULAW
42	CISCO_G729_b	100	32K_ADPCM
47	CISCO_G723_1r53	5339	CISCO_G729
48	CISCO_G723_1r63	5342	CISCO_G729_b
49	CISCO_G723_1ar53	5347	CISCO_G723_1r53
50	CISCO_G723_1ar63	5348	CISCO_G723_1r63
51	CISCO_G726_r16	5349	CISCO_G723_1ar53
52	CISCO_G726_r24	5350	CISCO_G723_1ar63
53	CISCO_G726_r32	5351	CISCO_G726_r16
55	CISCO_G728	5352	CISCO_G726_r24
		5353	CISCO_G726_r32
		5355	CISCO_G728

Correction Utility for Audio File Headers

This section describes the header fields that a correction utility must modify to enable an audio player to play back recordings made by the Cisco gateway to an HTTP or ESMTP server.

.AU File Format Correction

The data_size field in the .au header requires correcting. It is located at offset 8 from the beginning of the .au file. The correction utility must find the total size of the source .au file, subtract the .au header size at offset 4 from this file size, and insert the result into the data_size field at the offset 8 position.

The Cisco gateway puts a value of 24 in offset 4 and a value of 0 in offset 8, but this is not guaranteed. The correction utility should follow the correction logic outlined above.

Note The .au header fields are in big-endian format; byte-swapping is required for a correction utility running on little-endian platforms.

.WAV File Format Corrections

Three fields in the .wav header require correcting:

- total_size_minus_8 field—Located at offset 4 from the beginning of the .wav file. The correction utility must find the total size of the source .wav file, subtract 8 from this file size, and insert the result into this field at the offset 4 position.
- chunk_data_length field—The correction utility must find the total size of the source .wav file, and subtract one of these values from the file size:
  - 90 bytes for 32k ADPCM (G.726) codec
  - 56 bytes for all other codecs

  then insert the result into the field at one of these offset positions:
  - Offset 86 for 32k ADPCM (G.726) codec
  - Offset 52 for all other codecs
- total_sample_blocks field—total number of sample blocks inserted at this position:
  - Offset 74 for 32k ADPCM (G.726) codec
  - Offset 40 for all other codecs

The Cisco gateway puts a value of 0 in offset 4 and a value of 0 in offset 40, but this is not guaranteed. The correction utility should follow the correction logic outlined above.

Note The .wav header fields are in little-endian format; byte-swapping is required for a correction utility running on big-endian platforms.

How to Configure VoiceXML Voice Store and Forward

This section contains the following procedures:

- Configuring the On-Ramp Gateway for VoiceXML Voice Store and Forward, page 8
- Configuring the Off-Ramp Gateway to Place a Call, page 16

Configuring the On-Ramp Gateway for VoiceXML Voice Store and Forward

This section contains the following procedures for configuring the on-ramp gateway to record voice messages to an ESMTP server:

- Configuring the Interface Type for Sending Voice Mail, page 9 (optional)
- Configuring the Sending MTA, page 9 (required)
- Configuring the POTS Dial Peer, page 11 (required)
- Verifying the On-Ramp Gateway Configuration, page 12 (optional)

On-Ramp Gateway for VoiceXML Voice Store and Forward

When acting as an on-ramp voice gateway, the Cisco VoiceXML gateway records voice messages from end users, converts them into e-mail attachments, and forwards the MIME e-mail message with the voice mail attachments to an ESMTP server for storage. The voice calls and the delivery of the e-mails with voice mail attachments are controlled by VoiceXML documents running on the Cisco gateway.

Voice mail stored on the ESMTP server can be delivered either as an e-mail message with attachment when the recipient downloads messages or it can be forwarded as an e-mail notification to the off-ramp gateway. In the latter case, the ESMTP server delivers the voice mail to the off-ramp Cisco gateway, which processes the e-mail and initiates an outbound call to the destination. For more information, see the "Configuring the Off-Ramp Gateway to Place a Call" section on page 16.

The Cisco gateway supports the following mail formats:

- Mailto URL in it simplest form, which is the Internet mail address, as described in RFC-2368.
- Content transfer encoding is Base64.
- MIME content type is multipart/voice-message with the audio/* sub-type. Within the multipart/voice-message level, only one audio/* sub-type is enclosed.
- Audio formats are audio/basic and audio/wav.

The content type and audio format is set through a VoiceXML property. The sending MTA defines the delivery parameters for the e-mail message to which the recorded audio file is attached. The sending MTA can be specified through properties in the VoiceXML document or it can be configured on the Cisco gateway. A POTS dial peer on the gateway associates the dialed number with the appropriate VoiceXML application.

Configuring the Interface Type for Sending Voice Mail

Note On platforms with only voice cards, the default interface type is **fax-mail**, so you do not have to configure this command if your gateway has only voice cards. On platforms with a combination of modem and voice cards, the default is **modem**. and you must configure this command.

Specify the voice module in the gateway as the interface through which to send voice messages by using this command in global configuration mode:

```
    fax interface-type fax-mail
Example: Router(config)# fax interface-type fax-mail
```

Note After using the **fax interface-type fax-mail** command to change the interface type, you must reload the gateway for the new configuration to be effective.

Configuring the Sending MTA

The sending MTA defines the elements of the e-mail message to which the recorded audio file is attached. These elements can be configured globally on the gateway or they can be specified through Cisco properties in the VoiceXML document. For information on specifying the MTA agent by using VoiceXML properties, refer to the *Cisco VoiceXML Programmer's Guide*.

Note Specifying the sending MTA by using VoiceXML properties takes precedence over the gateway configuration. Any value that is configured on the gateway is ignored if the same attribute is specified in the VoiceXML document.

To configure the sending MTA on the on-ramp gateway, enter the following commands in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mta send mail-from hostname** *string*
4. **mta send mail-from** {**username** *string* | **username s**}
5. **mta send server** {*host-name* | *ip-address*}
6. **mta send subject** *string*
7. **mta send postmaster** *e-mail-address*
8. **mta send origin-prefix** *string*
9. **mta send return-receipt-to hostname** *string*
10. **mta send return-receipt-to username** {*string* | **s**}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
```

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
```

```
Example: Router# configure terminal
```

Step 3 Specify the host name portion of the origination e-mail address:

```
mta send mail-from hostname string
```

- *string*—Text string that specifies the SMTP host name or IP address. To specify an IP address, enclose it in brackets as follows: [xxx.xxx.xxx.xxx].

```
Example: Router(config)# mta send mail-from hostname onramp-gateway.com
```

Step 4 Specify the user name portion of the origination e-mail address:

```
mta send mail-from {username string | username $s$}
```

- **username** *string*—Text string that specifies the sender's username.
- **username s**—Macro that specifies that the username comes from the calling number.

```
Example: Router(config)# mta send mail-from username mailuser
```

Note The **mta send mail-from hostname** and **mta send mail-from username** commands configure the RFC 822 From: field and the RFC 821 Mail From field of the e-mail message. These two commands form a complete e-mail address, like mailuser@onramp-gateway.com. The To: address is configured by using the cisco-dest property in the VoiceXML document.

Step 5 Specify the destination server:

mta send server {*host-name* | *ip-address*}

- *host-name*—Host name of the destination mail server.
- *IP-address*—IP address of the destination mail server.

```
Example: Router(config)# mta send server mail-server
```

Step 6 Define the text that appears in the Subject field of the e-mail message:

mta send subject *string*

- *string*—Text string that appears in the subject header of an e-mail message.

```
Example: Router(config)# mta send subject e-mail subject goes here
```

Step 7 Define the origination address to use if the **mta send mail-from** address is not configured:

mta send postmaster *e-mail-address*

- *e-mail-address*—Destination address (the mail server postmaster account). This is also the address where all undeliverable e-mail is sent.

```
Example: Router(config)# mta send postmaster test@mail-server
```

Step 8 (Optional) Define additional identifying information to be prepended to the e-mail header.

mta send origin-prefix *string*

- *string*—Text string that adds comments to the e-mail prefix header. If this string contains more than one word, the string value should be contained within quotation marks (“x”).

```
Example: Router(config)# mta send origin-prefix “here is origin-prefix”
```

Step 9 (Optional) Specify the host name address where MDNs are sent:

mta send return-receipt-to hostname *string*

- **hostname** *string*—Text string that specifies the SMTP host name where MDNs will be sent.

```
Example: Router(config)# mta send return-receipt-to hostname onramp-gateway.com
```

Step 10 (Optional) Specify the username address where MDNs are sent:

mta send return-receipt-to username {*string* | **s**}

- *string*—Text string that specifies the sender’s username where MDNs will be sent.
- **s**—Wildcard that specifies that the calling number (ANI) generates the disposition-notification to the e-mail address.

```
Example: Router(config)# mta send return-receipt-to username $s$
```

Note If the **mta send return-receipt-to** address is not configured, the on-ramp gateway inserts the postmaster address in this field as a default.

Configuring the POTS Dial Peer

To configure the POTS dial peer, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **pots**
4. **service** *name*
5. **incoming called-number** *string*

6. **direct-inward-dial**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

`enable`

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

`configure terminal`

`Example: Router# configure terminal`

Step 3 Enter dial-peer configuration mode for a POTS dial peer:

`dial-peer voice` *`number`* **`pots`**

- *number*—Number tag used to identify this dial peer. Range is from 1 to 2147483647.

`Example: Router(config)# dial-peer voice 900 pots`

Step 4 Associate a specific call application with this dial peer:

`service` *`service-name`*

- *service-name*—Name of the VoiceXML application. This is the name of the application that was defined when the application was configured by using the application configuration submodes.

`Example: Router(config-dial-peer)# service vxml_smtp`

Step 5 Specify the called number that links voice calls to this dial peer:

`incoming called-number` *`string`*

- *string*—Sequence of digits representing the full or a partial telephone number (for example, the extension) used to reach the voice gateway. See the "How Voice Applications are Matched to Called Numbers" section on page 41 for more information.

If DID is enabled, the incoming called number (DNIS) is used to match the destination pattern of outgoing MMoIP dial peers.

`Example: Router(config-dial-peer)# incoming called-number 5551211`

Step 6 (Optional) Specify Direct Inward Dialing (DID) for this dial peer:

`direct-inward-dial`

`Example: Router(config-dial-peer)# direct-inward-dial`

Verifying the On-Ramp Gateway Configuration

To verify the on-ramp gateway configuration, perform the following steps:

SUMMARY STEPS

1. **show running-config**
2. **show dial-peer voice** *number*
3. **show call application voice summary**
4. **debug mmoip send email**

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that:

a. The interface type is set to **fax interface-type fax-mail**, for example:

```
!
fax interface-type fax-mail
```

> **Note** If your gateway has only voice cards, **fax-mail** is the default setting so the command does not show up in the configuration output.

b. The sending MTA is configured with the **mta send** commands, for example:

```
mta send server mapp-smtp
mta send subject subject line here
mta send origin-prefix This is the origin-prefix
mta send postmaster joe@mapp-smtp
mta send mail-from hostname Cisco-5300.cisco.com
mta send mail-from username $s$
mta send return-receipt-to hostname Cisco-5300.cisco.com
!
```

c. The POTS dial peer is configured to accept inbound calls to the called number in the **incoming called-number** command and links calls to the VoiceXML document in the **service** command, for example:

```
!
dial-peer voice 5550 pots
service smtp_rec
incoming called-number 5550100
!
```

Step 2 Use the **show dial-peer voice** command to display detailed configuration information about the dial peer, including whether it is operational and the assigned application. The following example output shows that POTS dial peer 5550 is linked to the application smtp_rec.

```
Router# show dial-peer voice 5550

VoiceEncapPeer5550
        information type = voice,
        description = `',
        tag = 5550, destination-pattern = `',
        answer-address = `', preference=0,
        CLID Restriction = None
        CLID Network Number = `'
        CLID Second Number sent
        source carrier-id = `', target carrier-id = `',
        source trunk-group-label = `',  target trunk-group-label = `',
        numbering Type = `unknown'
        group = 5550, Admin state is up, Operation state is up,
        incoming called-number = `5550100', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
        huntstop = disabled,
        in bound application associated: 'smtp_rec'
        out bound application associated: ''
        dnis-map =
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        Translation profile (Incoming):
        Translation profile (Outgoing):
```

```
incoming call blocking:
translation-profile = `'
disconnect-cause = `no-service'
voice-port = `'
type = pots, prefix = `',
forward-digits default
session-target = `', up,
direct-inward-dial = disabled,
digit_strip = enabled,
register E.164 number with GK = TRUE
fax rate = system,   payload size =  20 bytes

Time elapsed since last clearing of voice call statistics never
Connect Time = 0, Charged Units = 0,
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 0, Refused Calls = 0,
Last Disconnect Cause is "",
Last Disconnect Text is "",
Last Setup Time = 0.
```

Step 3 Use the **show call application voice summary** command to verify that the VoiceXML application is loaded and running on the gateway, for example:

```
Router# show call application voice summary

name                 description

session              Basic app to do DID, or supply dialtone.
fax_hop_on           Script to talk to a fax redialer
clid_authen          Authenticate with (ani, dnis)
clid_authen_collect  Authenticate with (ani, dnis), collect if that fails
clid_authen_npw      Authenticate with (ani, NULL)
clid_authen_col_npw  Authenticate with (ani, NULL), collect if that fails
clid_col_npw_3       Authenticate with (ani, NULL), and 3 tries collecting
clid_col_npw_npw     Authenticate with (ani, NULL) and 3 tries without pw
DEFAULT              Default system session application
lib_off_app          Libretto Offramp
callme               flash:call.vxml
*smtp_rec            tftp://10.10.5.3/scripts/smtp_rec.vxml

TCL Script Version 2.0 supported.
TCL Script Version 1.1 supported.
Voice Browser Version 2.0 for VoiceXML 1.0 & 2.0 supported.
```

Tip If an asterisk is displayed next to the application name when using the **summary** keyword, it means that the application is configured, but not running. Normally this is because the application was not successfully loaded. For troubleshooting information, see the "Verifying Loading of Service" section on page 26.

Step 4 Use the **debug mmoip send email** command and send an e-mail to a specified e-mail address to test connectivity between the on-ramp gateway and the e-mail server.

Troubleshooting Recording Configuration

SUMMARY STEPS

1. Play the recorded audio file using an audio tool such as Cool Edit.
2. **debug vxml puts**
3. **debug voip application**
4. **show vsp session**

DETAILED STEPS

Step 1 Play the recorded audio file using an audio tool such as Cool Edit.

Step 2 Use the **debug vxml puts** command to verify the duration and size of the recorded audio file.

You can run a simple VoiceXML document that uses the <cisco-puts> tag to print the duration and size of the audio recording, for example:

```
<form id="record_to_http">

     <record name="myrec" beep="true" maxtime="15s" finalsilence="10s" dtmfterm="true"
     type="audio/basic;codec=g711ulaw">
     <prompt><audio src="record.au"/></prompt>
     </record>

     <block>
     <cisco-puts>DURATION:<cisco-putvar namelist="myrec$.duration"/></cisco -puts>
     <cisco-puts>SIZE:<cisco-putvar namelist="myrec$.size"/></cisco-puts> ….
```

Step 3 Use the **debug voip application** command to verify the start and stop times of the audio recording, for example:

```
Router# show dial-peer voice 555

*Mar 1 14:53:26.610: $ mr_record_start: stream 0x63811808 recording is now
startedmc=0x63819DC8
*Mar 1 14:53:36.610: ms_stop_record: mc=0x63819DC8, cid=0x38,
cause=MS_STOP_MAX_TIME, RECORDING, RAM
*Mar 1 14:53:36.610: mr_vsp_flush_stream: mc=0x63819DC8, cid=0x38
*Mar 1 14:53:36.610: ms_stop_record_done: cid=0x38 stopped at 14:53:33.376, dur=10000
(ms),
DISASSOCIATING
*Mar 1 14:53:36.610: msu_recrd_ms_record_complete: callID=0x38(56),
context=0x62E13E9C,
cause=MS_STOP_MAX_TIME, stream_id=1
*Mar 1 14:53:36.610: msu_recrd_ms_record_complete: Recorded MC:0x63819DC8
URL ram:myrecord_6_0_56
*Mar 1 14:53:36.610: msu_call_app: app_cbf=0x610B09E0
```

Step 4 Use the **show vsp session** command to view the packet statistics for the recording session, for example:

```
Router# show vsp session

VSP_STATS: Session Statistics -
        sessions total=4; max_active=1, current=1
        session_duration last=0; max=21000, min=21000 ms
        pre_stream_wait last=0; max=20, min=20 ms
        stream_duration last=0; max=20980, min=20980 ms
        post_stream_wait last=0; max=0, min=0 ms
        stream_size last=240; max=167804, min=240 bytes
```

```
        streaming_rate last=0; max=7000, min=7000 bytes/sec
        total_packet_count last=0; max=745, min=745 packets
        drop_packet_count last=0; max=0, min=0 packets
        particle_packet_count last=0; max=745, min=745 packets

VSP: current reference time=63832
        se_st = session start time; st_st = stream start time
        rx_ct = rx codec type; tx_ct = tx codec type
        rx_pt = rx payload type; tx_pt = tx payload type
VSP:<4>[0x649FBE78](0x649FBE78) MS->HTTP se_st=277460, st_st=277472; rx_ct=g711ulaw,
tx_ct=g711ulaw
```

Configuring the Off-Ramp Gateway to Place a Call

To configure the off-ramp gateway, perform the tasks in the following sections:

- Downloading the Tcl Script for Off-Ramp Mail Application, page 19 (required)
- Loading the Mail Application onto the Gateway, page 19 (required)
- Configuring the Interface Type for Receiving Voice Mail, page 21 (required)
- Configuring the Receiving MTA for Voice Store and Forward, page 21 (required)
- Configuring the POTS Dial Peer, page 22 (required)
- Configuring the MMoIP Dial Peer, page 23 (required)
- Verifying the Off-Ramp Gateway Configuration, page 24 (optional)

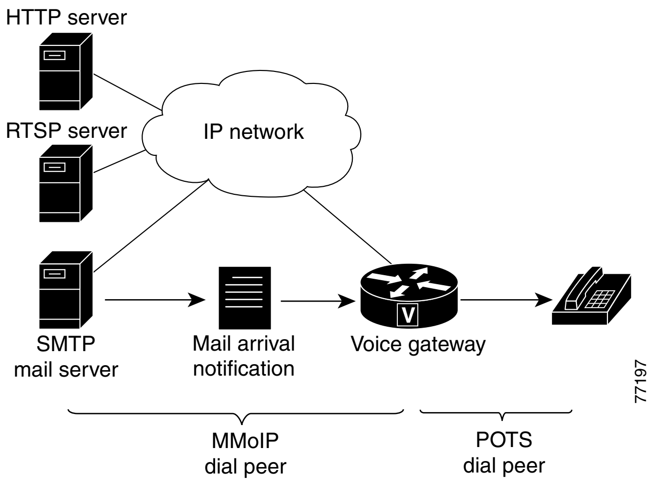
Off-Ramp Gateway Placing a Call

The Cisco VoiceXML gateway can act as an off-ramp gateway to dial the PSTN or IP network after it receives an e-mail notification of a voice message. The off-ramp gateway:

- Accepts e-mail notifications of voice messages from ESTMP servers.
- Hands e-mail messages over to the off-ramp mail application script, *offramp-mapp.tcl,* which extracts the destination telephone number.
- Places an outgoing call to the destination PSTN or IP voice user.
- Loads the specified VoiceXML application when the destination answers.
- Retrieves audio clips and plays out voice messages to the destination.

The off-ramp gateway uses receiving MTAs to define the parameters of the SMTP server. An MMOIP dial peer specifies the Tcl mail application that hands calls off to the outbound POTS dial peer. A call is placed to the destination telephone number and the specified VoiceXML application is executed when the call is answered.

Figure 5-1 Voice Store and Forward Off-Ramp Call



Note Converting voice mail attachments to audio clips and saving them to an external RTSP or HTTP server is the responsibility of external application servers; it is not within the scope of the Cisco VoiceXML gateway.

Off-Ramp Mail Trigger Call Scenario

The following steps describe how the off-ramp gateway delivers voice messages:

Step 1 The gateway receives an e-mail notification from the external MTA. The notification e-mail does not contain a real voice message; it merely acts as a trigger to launch the off-ramp mail application. The voice mail attachments are ignored by the gateway.

Step 2 If the receipt To: field in the message header begins with *VXML=*, the e-mail message is treated as a voice notification. The following example shows how the receipt To: field in the e-mail message header begins with VXML=, followed by the called number:

```
To: VXML=+16505554321@cisco.com
From: "John Smith" <15105553456@cisco.com>
Date: Mon, 26 Sept 2001 10:20:20 -0700 (CDT)
Subject: Voice message from cisco systems
Message-ID: 1234563456789@cisco.com
MIME-Version: 1.0  (Voice 2.0)
X-Mailer: IOS (tm) 5300 Software (C5300-IS-M)
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
```

If the receipt To: field begins with FAX=, it is considered a T.37 fax message and is handed off to the Store and Forward Fax feature.

Step 3 The gateway extracts the destination telephone number from the e-mail header and matches it to an MMOIP dial peer.

Step 4 The mail application that is configured in this MMOIP dial peer is executed. This is the mail application that uses the app_voicemail_offramp.tcl script provided by Cisco.

Step 5 The mail application hands off the call to the POTS dial-peer that matches the dialed number, and an outbound call is made to the PSTN using this destination telephone number.

Step 6 If the call is answered, the gateway executes the VoiceXML document that is configured in the POTS dial peer. If the called number is busy or there is no answer, the VoiceXML script is not executed.

If DSN is configured and the called number is busy, or there is no answer or some other error occurs, the sender receives a "mail is not deliverable" error message. When there is no answer or the called number is busy, the gateway sends the DSN error status after the SMTP transaction completes (which means that the gateway has received the full e-mail from the mail server). If the gateway can not match an outbound dial peer, the DSN error message is delivered immediately.

MDNs

A message disposition notification (MDN) indicates that an e-mail message has been opened. A sender requests that an MDN be returned when the receiver opens an e-mail message. The following header is included in the e-mail header of the message:

```
Disposition-Notification-To:
```

The MDN is initiated by the sending e-mail client, and the return receipt is generated by the receiving e-mail client. Most PC-based e-mail software applications, such as Eudora, Netscape Messenger, and Microsoft Outlook, generate MDNs. The MDN is sent by the on-ramp gateway to the user defined in the **mta send return-receipt-to** command.

RFC 2298 requires that the receiver be allowed to prevent the automatic generation of an MDN. Because of this requirement, it is difficult to determine whether or not the user has actually received the e-mail message. For example, the recipient can always choose not to respond to MDN requests, or the recipient software might not understand or accept MDN requests.

DSNs

A delivery status notification (DSN) is a message or response that is automatically generated and sent to the originator of an e-mail message by the SMTP server, notifying the sender of the status of the e-mail message. The on-ramp DSN request is included as part of the e-mail message sent by the on-ramp gateway if the appropriate DSN property is specified in the VoiceXML document. The on-ramp DSN response is generated by the SMTP server when the e-mail message is accepted. The DSN is sent to the user defined in the **mta send mail-from** command.

The off-ramp DSN is requested by the e-mail client. The DSN response is generated by the off-ramp gateway when it receives a request as part of the e-mail message. DSNs can only be generated if the mail client on the SMTP server is capable of responding to a DSN request. Because the off-ramp gateway generates the DSNs, you must configure both the **mta send mail-from** and **mta send return-receipt to** commands to use the DSN feature, for example:

```
mail from: <user@mail-server.company.com>
rcpt to: <555-1212@company.com> NOTIFY=SUCCESS,FAILURE,DELAY
```

Three different states can be reported back to the sender in the DSN:

- Delayed—Message delivery was delayed.
- Failure—SMTP server was unable to deliver the message to the recipient.
- Success—Message was successfully delivered to the recipient mailbox.

Note For information on specifying DSN properties in a VoiceXML document, refer to the *Cisco VoiceXML Programmer's Guide*.

Downloading the Tcl Script for Off-Ramp Mail Application

Before configuring the Cisco gateway to act as an off-ramp gateway for placing outbound calls, you must download and install the required Tcl script for the off-ramp mail application. This Tcl script, *app_voicemail_offramp.tcl*, hands off calls to the configured VoiceXML application. For a description of how this mail trigger works, see the "Off-Ramp Mail Trigger Call Scenario" section on page 17.

Download the required Tcl script by completing the following steps:

Step 1 Log in to the Cisco web site and go to http://www.cisco.com/cgi-bin/tablebuild.pl/tclware.

Step 2 Select the Tcl zip file that contains the mail application: app-voicemail-offramp.2.0.0.0.zip (or later).

Step 3 Select the Cisco.com server nearest your physical location and download the file.

Step 4 Unzip the contents of the file.

The zip file that you download includes:

- Mail application Tcl script (app_voicemail_offramp.tcl)
- ReadMe file
- Call flow file

Step 5 Move the application script file (app_voicemail_offramp.tcl) to a location that can be accessed by your gateway using a URL format.

Tcl scripts and VoiceXML documents can be stored in any of the following locations: TFTP, FTP, or HTTP servers, in Flash memory of the voice gateway, or on the removable disks of the Cisco 3600 series. The audio files that they reference can be stored in any of these locations, and on RTSP servers. The URL is a standard URL that points to the location of the script. Examples include:

- flash:myscript.tcl—The script called myscript.tcl is being loaded from Flash memory on the router.
- slot0:myscript.tcl—The script called myscript.tcl is being loaded from a device in slot 0 on the router.
- tftp://bigserver/myscripts/ticketime.tcl—The script called ticketime.tcl is being loaded from a server called bigserver in a directory within the tftpboot directory called myscripts.

Loading the Mail Application onto the Gateway

To load the off-ramp mail application and corresponding VoiceXML application into the gateway's memory, enter the following commands:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. application

4. **service** *application-name location*
5. **param mail-script** *application-name*
6. **service** *application-name location*
7. **param dsn-script** *application-name*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
    enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode:

```
    application
Example: Router(config)# application
```

Step 4 Load the off-ramp mail script and assign it an application name:

```
application
  service application-name location
```

- *application-name*—Name of the off-ramp mail application.
- *location*—Directory and filename of the Tcl script in URL format. For example, flash memory (flash:filename), a TFTP (tftp://../filename) or an HTTP server (http://../filename) are valid locations. This application must use the app_voicemail_offramp.tcl script downloaded from Cisco.

```
Example:
Router(config)# application
Router(config-app)# service off tftp://serv1/app_voicemail_offramp.tcl
```

Step 5 Load the VoiceXML application that handles calls from the off-ramp mail application:

```
application
  service application-name location
```

- *application-name*—Name of the VoiceXML application to which the off-ramp mail application hands off calls when the destination answers.
- *location*—Directory and filename of the VoiceXML document in URL format. For example, flash memory (flash:filename), a TFTP (tftp://../filename) or an HTTP server (http://../filename) are valid locations.

```
Example: Router(config)#
Router(config)# application
Router(config-app)# service mapp1 tftp://serv1/mapp-1.vxml
```

Step 6 Link the off-ramp mail application to the VoiceXML application:

```
application
service mail-application-name
  param mail-script application-name
```

- *mail-application-name*—Name of the off-ramp mail application that launches the app_voicemail_offramp.tcl script when the gateway receives an e-mail trigger. This is the application named in Step 4.

- *application-name*—Name of the VoiceXML application to which the off-ramp mail application hands off the call when the destination answers. This is the application named in Step 5.

```
Example:
Router(config)#application
Router(config-app)#service offramp
Router(config-app-param)#param mail-script mapp1
```

Step 7 (Optional) Link the off-ramp mail application to the VoiceXML application that handles calls for off-ramp DSN and MDN e-mail messages:

```
application
  service mail-application-name
    param dsn-script application-name
```

- *mail-application-name*—Name of the off-ramp mail application that launches the app_voicemail_offramp.tcl script when the gateway receives an e-mail trigger.
- *application-name*—Name of the VoiceXML application to which the off-ramp mail application hands off the call when the destination answers.

```
Example:
Router(config)#application
Router(config-app)#service offramp
Router(config-app-param)#param dsn-script map_dsn
```

Configuring the Interface Type for Receiving Voice Mail

Note On platforms with only voice cards, the default interface type is **fax-mail**, so you do not have to configure this command if your gateway has only voice cards. On platforms with a combination of modem and voice cards, the default is **modem**. and you must configure this command.

Specify the voice module in the gateway as the interface through which to receive voice messages by using this command in global configuration mode:

```
    fax interface-type fax-mail
Example: Router(config)# fax interface-type fax-mail
```

Note After using the **fax interface-type fax-mail** command to change the interface type, you must reload the gateway for the new configuration to take affect.

Configuring the Receiving MTA for Voice Store and Forward

To configure the receiving MTA on the off-ramp gateway, enter the following commands in global configuration mode:

1. **enable**
2. **configure terminal**
3. **mta receive aliases** *string*
4. **mta receive maximum-recipients** *number*
5. **mta receive generate-mdn**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Define a host name to use as an alias for the off-ramp gateway:

mta receive aliases *string*

- *string*—Host name or IP address to use as an alias for the SMTP server.

You can define up to ten different aliases. The Cisco gateway MTA only accepts incoming mail if the destination host name of the incoming mail matches one of the aliases configured by the **mta receive aliases** command. This command does not automatically allow reception for a domain IP address—it must be explicitly added. If you add an IP address, you must enclose the address in brackets as follows: [xxx.xxx.xxx.xxx].

```
Example 1: Router(config)# mta receive aliases cisco.com
Example 2: Router(config)# mta receive aliases [10.10.1.1]
```

Step 4 Define the number of simultaneous SMTP recipients handled by this gateway:

mta receive maximum-recipients *number*

- *number*—Maximum number of recipients for all SMTP connections. Range is from 0 to 1024. The default is 0, which means that the off-ramp function is disabled.

This limits the number of resources (modems) allocated for e-mail transmissions.

```
Example: Router(config)# mta receive maximum-recipients 10
```

Step 5 (Optional) Configure the off-ramp gateway to generate an MDN message when requested to do so:

mta receive generate-mdn

You may want to enable this feature depending on the type of mailer in use.

```
Example: Router(config)# mta receive generate-mdn
```

Configuring the POTS Dial Peer

To configure a POTS dial peer that the off-ramp gateway uses to place calls, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **pots**
4. **destination-pattern** *string*
5. **port** *controller-number*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

`Example: Router# configure terminal`

Step 3 Enter dial-peer configuration mode for a POTS dial peer:

dial-peer voice *number* **pots**

- *number*—Number tag used to identify this dial peer. Range is from 1 to 2147483647.

`Example: Router(config)# dial-peer voice 555 pots`

Step 4 Identify the destination telephone number to which the mail application places calls:

destination-pattern *string*

- *string*—Sequence of digits representing the full or a partial telephone number (for example, the extension) used to reach the destination.

`Example: Router(config-dial-peer)# destination-pattern 5551939`

Step 5 Associate a specific voice port with this dial peer:

port *controller-number*

This command specifies the T1 controller port through which outgoing voice calls are routed.

- *controller-number*—Number of the T1 or E1 controller.
- **:D**—D channel associated with ISDN PRI.

Note The syntax of the **port** command is platform-specific. For information about the specific syntax for your platform, refer to the *Cisco IOS Voice Command Reference*, Release 12.3.

```
Example 1: (Cisco 3600 series) Router(config-dial-peer)# port 1/0/0
Example 2: (Cisco AS5300, AS5350, and AS5400) Router(config-dial-peer)# port 1/0:D
```

Configuring the MMoIP Dial Peer

To configure the off-ramp gateway MMoIP dial peer, use the following commands beginning in global configuration mode:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **mmoip**
4. **service** *service-name*
5. **incoming called-number** *string*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Enter dial-peer configuration mode for a MMOIP dial peer:

dial-peer voice *number* **mmoip**

- *number*—Number tag used to identify this dial peer. Range is from 1 to 2147483647.

```
Example: Router(config)# dial-peer voice 1000 mmoip
```

Step 4 Associate the mail application with this dial peer:

service *service-name*

- *service-name*—Name of the mail application. Calls matched to this dial peer are handed off to this application. This application must use the app_voicemail_offramp.tcl script provided by Cisco.

```
Example: Router(config-dial-peer)# service mapp1
```

Step 5 Specify the called number that links voice calls to this dial peer:

incoming called-number *string*

- *string*—Sequence of digits representing the full or a partial telephone number (for example, the extension) used to reach the voice gateway. See the "How Voice Applications are Matched to Called Numbers" section on page 41 for more information.

```
Example: Router(config-dial-peer)# incoming called-number 5551939
```

Note The value of the **incoming called-number** command in the MMoIP dial peer must match the value of the **destination-pattern** command in the corresponding off-ramp POTS dial peer.

Verifying the Off-Ramp Gateway Configuration

To verify the off-ramp gateway configuration, perform the following steps:

SUMMARY STEPS

1. **show running-config**
2. **show dial-peer voice** *number*
3. **show call application voice summary**
4. Send an e-mail to the off-ramp gateway and request a return receipt.
5. **debug mta receive all**

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that:

a. The interface type is set to **fax interface-type fax-mail**, for example:

```
!
fax interface-type fax-mail
```

Note If your gateway has only voice cards, **fax-mail** is the default setting so the command does not show up in the configuration output.

b. The receiving MTA is configured with the **mta receive alias** and the **mta receive maximum-recipients** commands, for example:

```
mta receive aliases cisco.com
mta receive maximum-recipients 10
!
```

c. The off-ramp mail application script and the associated VoiceXML document are configured on the gateway with the application configuration submodes, and the two applications are linked with the **param mail-script** command, for example:

```
application
  service offramp tftp://10.10.17.1/scripts/tcl/app_voicemail_offramp.tcl
    param mail-script mapp1
  service tftp://10.10.17.1/scripts/vxml/mapp-1.vxml
!
```

d. The MMoIP dial peer is configured to use the off-ramp mail application with the **service** command, for example:

```
!
dial-peer voice 555 mmoip
 service offramp
 incoming called-number 5551212
```

Note Be sure that the **information-type fax** command is not configured in the MMoIP dial peer. The information type is set to voice by default and therefore does not display in the configuration output.

e. The POTS dial peer is configured to make outbound calls to the destination by using the called number in the **destination-pattern** command, for example:

```
!
dial-peer voice 5551 pots
 destination-pattern 5551212
 port 0:D
!
```

Step 2 Use the **show dial-peer voice** command to display detailed configuration information about the dial peers, including the information type, whether they are operational, and the assigned application. For example, the following output shows that MMoIP dial peer 555 is linked to the application *offramp*.

```
Router# show dial-peer voice 555

MultiMediaOverIpPeer555
        information type = voice,
        description = `',
        tag = 555, destination-pattern = `',
        answer-address = `', preference=0,
        CLID Restriction = None
        CLID Network Number = `'
        CLID Second Number sent
        source carrier-id = `', target carrier-id = `',
        source trunk-group-label = `',  target trunk-group-label = `',
        numbering Type = `unknown'
        group = 555, Admin state is up, Operation state is up,
        incoming called-number = `555....', connections/maximum = 0/unlimited,
```

```
DTMF Relay = disabled,
modem transport = system,
huntstop = disabled,
in bound application associated: 'offramp'
out bound application associated: ''
dnis-map =
permission :both
incoming COR list:maximum capability
outgoing COR list:minimum requirement
Translation profile (Incoming):
Translation profile (Outgoing):
incoming call blocking:
translation-profile = `'
disconnect-cause = `no-service'
type = mmoip, session-target = `',
session-protocol = SMTP,
Image Encoding Type = pass-through,
Image Resolution = pass-through,
Disposition Notification = disabled,
Delivery Status Notification = ,
Time elapsed since last clearing of voice call statistics never
Connect Time = 0, Charged Units = 0,
Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
Accepted Calls = 0, Refused Calls = 0,
Last Disconnect Cause is "",
Last Disconnect Text is "",
Last Setup Time = 0.
```

Step 3 Use the **show call application voice summary** command to verify that the VoiceXML application is loaded and running on the gateway, for example:

```
Router# show call application voice summary

name                 description

session              Basic app to do DID, or supply dialtone.
fax_hop_on           Script to talk to a fax redialer
clid_authen          Authenticate with (ani, dnis)
clid_authen_collect  Authenticate with (ani, dnis), collect if that fails
clid_authen_npw      Authenticate with (ani, NULL)
clid_authen_col_npw  Authenticate with (ani, NULL), collect if that fails
clid_col_npw_3       Authenticate with (ani, NULL), and 3 tries collecting
clid_col_npw_npw     Authenticate with (ani, NULL) and 3 tries without pw
DEFAULT              Default system session application
lib_off_app          Libretto Offramp
callme               flash:call.vxml
smtp_rec             tftp://10.10.5.3/scripts/smtp_rec.vxml
offramp              tftp://10.10.17.1/scripts/tcl//app_voicemail_offramp.tcl
mapp1                tftp://10.10.17.1/scripts/vxml/mapp-1.vxml

TCL Script Version 2.0 supported.
TCL Script Version 1.1 supported.
Voice Browser Version 2.0 for VoiceXML 1.0 & 2.0 supported.
```

Tip If an asterisk is displayed next to the application name when using the **summary** keyword, it means that the application is configured, but not running. Normally this is because the application was not successfully loaded. For troubleshooting information, see the "Verifying Loading of Service" section on page 26.

Step 4 Send an e-mail to the off-ramp gateway and request a return receipt. To be accepted by the gateway, the destination e-mail address must use the appropriate MTA receive alias, as configured by using the **mta receive alias** command.

Step 5 Use the **debug mta receive all** command to view output relating to the activity on the SMTP server (messages exchanged, for example, the handshake) between the e-mail server and the off-ramp gateway.

Configuration Examples for VoiceXML Voice Store and Forward

This section provides the following gateway configuration examples of Cisco Tcl and VoiceXML applications. It contains comments in places especially relevant to the configuration of the feature.

- VoiceXML Voice Store and Forward on Cisco AS5300 Example, page 27
- VoiceXML Voice Store and Forward Off-Ramp on Cisco 3600 Series Example, page 30

VoiceXML Voice Store and Forward on Cisco AS5300 Example

```
!
version 12.3
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
service udp-small-servers
service tcp-small-servers
!
hostname Cisco-5300
!
logging buffered 1000000 debugging
aaa new-model
!
aaa authentication login fax enable
aaa authorization exec fax group radius
aaa authorization network fax group radius
aaa accounting connection fax stop-only group radius
aaa session-id common
!
username access-class nopassword
!
resource-pool disable
!
ip subnet-zero
ip tftp source-interface Ethernet0
ip domain-name cisco.com
ip host mail-server.cisco.com 1.14.116.1
ip host demo 223.255.254.254
ip host rtsp-ws 1.7.153.4
ip host mapp-smtp.cisco.com 1.7.127.3
ip host mapp-rtsp.cisco.com 1.7.127.151
ip name-server 1.14.116.1
!
isdn switch-type primary-5ess
!
voice service pots
 fax protocol t38 ls-redundancy 5 hs-redundancy 0
```

```
!
voice service voip
 fax protocol t38 ls-redundancy 5 hs-redundancy 0
 h323
!
fax interface-type fax-mail
mta send server mapp-smtp
mta send subject subject line here
mta send origin-prefix This is the origin-prefix
mta send postmaster joe@mapp-smtp
mta send mail-from hostname Cisco-5300.cisco.com
mta send mail-from username $s$
mta send return-receipt-to hostname Cisco-5300.cisco.com
mta send return-receipt-to username $s$
mta receive aliases mail-server2
mta receive aliases [1.7.87.4]
mta receive aliases cisco.com
mta receive maximum-recipients 10
mmoip aaa send-id primary gateway
mmoip aaa method fax authentication gateway
mmoip aaa send-authentication enable
mmoip aaa receive-accounting enable
mmoip aaa receive-authentication enable
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing sf
 linecode ami
!
controller T1 2
 framing sf
 linecode ami
!
controller T1 3
 framing sf
 clock source line secondary 3
 linecode ami
!
interface Ethernet0
 ip address 1.7.87.4 255.255.0.0
!
interface Serial0:23
 no ip address
 isdn switch-type primary-5ess
 isdn incoming-voice modem
!
interface FastEthernet0
 no ip address
 shutdown
 duplex auto
 speed auto
!
ip default-gateway 1.7.0.1
ip classless
ip route 223.255.254.0 255.255.255.0 1.7.0.1
no ip http server
ip pim bidir-enable
!
!
```

```
priority-list 1 protocol ip high tcp 1720
dialer-list 1 protocol ip permit
!
!
radius-server host 1.7.127.3 auth-port 1645 acct-port 1646
radius-server retransmit 3
radius-server key password
radius-server vsa send accounting
radius-server vsa send authentication
call rsvp-sync
application
 service rtsp_record tftp://demo/vxml/rtsp_record_vxml.vxml
 !
 service rtsp_play tftp://demo/vxml/rtsp_play.vxml
 !
 service http_submit tftp://demo/vxml/http_submit.vxml
 !
 service testcgi tftp://demo/vxml/testcgi.vxml
 !
 service http_play tftp://demo/vxml/http_play.vxml
 !
 service http_record tftp://demo/vxml/http_record.vxml
 !
 service mapp1 tftp://demo/tcl/mapp-1.tcl
  param mail-script offramp-mapp-test
  param authentication enable
  param authen-list fax
  param authen-method envelope-from
 !
 service offramp-mapp-test tftp://demo/vxml/testplay.vxml
 !
voice-port 0:D
!
mgcp modem passthrough voip mode ca
no mgcp timer receive-rtcp
!
mgcp profile default
!
dial-peer voice 101 pots
 description ***** change the application on this dial peer accordingly *****
 service http_play
 incoming called-number 5....
 port 0:D
!
dial-peer voice 208 voip
 shutdown
 description ***** no shut this dial peer for voip call *****
 destination-pattern 5556681
 session target ipv4:1.7.87.2
 codec g711ulaw
!
dial-peer voice 210 voip
 destination-pattern 52924
 session target ipv4:1.7.87.5
!
dial-peer voice 108 pots
 destination-pattern 5556681
 port 0:D
 prefix 95556681
!
dial-peer voice 545 pots
 incoming called-number 529..
 shutdown
 destination-pattern 529..
```

```
 direct-inward-dial
 port 0:D
 prefix 529
!
dial-peer voice 1008 mmoip
 description ***** email trigger incoming dial peer *****
 service mapp1
 incoming called-number 5556681
!
dial-peer voice 555 voip
 incoming called-number 5556248
 codec g711ulaw
!
dial-peer voice 5556 pots
 description ***** outgoing dial peer to the destination number *****
 destination-pattern 5556681
 port 0:D
 prefix 95556681
!
line con 0
 exec-timeout 0 0
 password itxctest
line aux 0
 password password
 no exec
line vty 0
 exec-timeout 0 0
 password password
 no exec
line vty 1 3
line vty 4
 exec-timeout 0 0
!
end
```

VoiceXML Voice Store and Forward Off-Ramp on Cisco 3600 Series Example

```
!
version 12.2
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Cisco-3640
!
logging rate-limit console 10 except errors
no logging console
enable secret 5 $1$2DFtyu
enable password sample
!
username test password 0 test123
!
!
voice-card 1
 codec complexity high
!
ip subnet-zero
!
ip ftp username test
ip ftp password test123
!
```

```
no ip dhcp-client network-discovery
isdn switch-type primary-5ess
!
!
fax interface-type fax-mail
mta send server 10.10.17.1
mta send subject Test message from Cisco-3640
mta send postmaster test@ultra5.cisco.com
mta send mail-from hostname Cisco-3640
mta send mail-from username test
mta receive aliases [10.10.178.11]
mta receive maximum-recipients 120
!
!
controller T1 1/0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1/1
 framing esf
 clock source internal
 linecode b8zs
!
!
interface FastEthernet0/0
 ip address 1.2.178.11 255.255.0.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial1/0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-5ess
 isdn incoming-voice voice
 isdn T310 4000
 no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 1.2.0.1
no ip http server
!
!
snmp-server manager
call rsvp-sync
!
application
 service mapp1 tftp://10.10.17.1/docs/scripts/app_voicemail_offramp.tcl
  param mail-script docs-mapp-test
 !
 service docs-mapp-test tftp://10.10.17.1/docs/scripts/offramp/mapp_31.vxml
 !
voice-port 1/0:23
!
!
mgcp profile default
!
```

```
dial-peer cor custom
!
!
dial-peer voice 1000 pots
 destination-pattern 5551211
 port 1/0:23
 prefix 5551211
!
dial-peer voice 2000 mmoip
 service mapp1
 incoming called-number 5551211
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 password sample
 login
line vty 5 15
 password sample
 login
!
exception core-file Cisco-3640-core
exception protocol ftp
exception dump 10.10.17.1
no scheduler allocate
!
end
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl IVR and VoiceXML Applications" on page 1.
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties" on page 1.
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML" on page 1.
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features" on page 1.
- To configure session interaction for a Tcl IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- "" on page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release

- “Overview of Cisco IOS Tcl IVR and VoiceXML Applications” on page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance

Configuring ASR and TTS Properties

This chapter describes how to configure automatic speech recognition (ASR) and test-to-speech (TTS) attributes in Cisco IOS software for use by Tcl and VoiceXML applications. These attributes are part of the Speech Recognition and Synthesis for Voice Applications feature.

For more information about this and related Cisco IOS voice features, see the following:

- “” on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for Speech Recognition and Synthesis for Voice Applications

Release	Modification
12.2(11)T	This feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.
12.4(15)T	Support was added for Media Resource Control Protocol version 2 (MRCP v2).

Contents

- Prerequisites for Using ASR and TTS, page 2
- Restrictions for External ASR and TTS Servers, page 3
- Information About Speech Recognition and Synthesis, page 4
- How to Configure External Server Properties, page 5
- Configuration Examples for ASR and TTS, page 18
- Where to Go Next, page 22

- Additional References, page 22

When developing and configuring a voice application, use this chapter and see the *Cisco VoiceXML Programmer's Guide* or the *Tcl IVR API Version 2.0 Programmer's Guide*.

Prerequisites for Using ASR and TTS

Using speech recognition or synthesis with a voice application requires:

- Cisco IOS Release 12.2(11)T or a later release
- Basic Cisco IOS functionality as described in "Configuring Basic Functionality for Tcl IVR and VoiceXML Applications" on page 1.
- Tcl IVR 2.0 script or VoiceXML 2.0 document that implements ASR or TTS. To write your own script, see the *Tcl IVR API Version 2.0 Programmer's Guide* or *Cisco VoiceXML Programmer's Guide*, respectively.
- Compatible version of external media server software, and VCWare (Cisco AS5300), as listed in Table 6-1.

For Cisco IOS Release 12.4(15)T and later releases, you also need:

- Loquendo ASR and TTS MRCP v2 servers
- Loquendo Speech Suite version 7.0.6 or later

or

- Nuance Recognizer 9.0.0
- RealSpeak 4.5.0
- Nuance Speech Server 5.0.1

Table 6-1 Cisco IOS Compatibility Matrix

Software	Supported Version	Cisco IOS Release
Cisco VCWare	10.26a with Cisco DSPWare 4.0.26	12.2(11)T 12.2(11)T1
Nuance ASR/TTS Minimum Recommended Software	• Nuance Recognizer 9.0.0 – At least one Nuance Recognizer 9.0 language • RealSpeak 4.5.0 – At least one RealSpeak 4.5 voice • Nuance Speech Server 5.0.1	12.4(15)T
	• OpenSpeech Recognizer 3.0.12 – At least one OSR 3.0 language • RealSpeak 4.0.12 – At least one RealSpeak 4.0 voice • SpeechWorks MediaServer 3.1.13	12.4(15)T

Table 6-1 Cisco IOS Compatibility Matrix

Software	Supported Version	Cisco IOS Release
	• OpenSpeech Recognizer 3.0.9 – At least one OSR 3.0 language • RealSpeak 4.0.10 – At least one RealSpeak 4.0 voice • SpeechWorks MediaServer 3.1.13	12.4(6)XT
	• Nuance MRCP 1.0.0 SP10 (includes Nuance 8.5 SP050930) • Vocalizer 4.0.6 – At least one Vocalizer 4.0 voice	12.4(6)XT
Loquendo ASR and TTS MRCP v2 servers	• Speech Suite version 7.0.6 or later	12.4(15)T

Nuance Communications Resources

For general information, visit the Nuance Communications, Inc. website:

http://www.nuance.com/

For Nuance software downloads and documentation, visit the Nuance Network Customer Service Portal:

http://network.nuance.com

Loquendo Resources

For specific information on installing and configuring the external media server, see the documentation resources on the Loquendo website.

Restrictions for External ASR and TTS Servers

- For speech recognition and external DTMF recognition, the actual grammar formats that are supported are dependent on the media server being used.
- External ASR and TTS media servers must support:
  - Server side of MRCP
  - HTTP clients
  - W3C XML grammar format as a minimum for ASR
  - W3C speech synthesis markup language specification for TTS
- To use speech recognition on an IP call leg:
  - The **codec** command must be set to G.711 u-law in the VoIP dial peer.
  - The **dtmf-relay** command must be set to rtp-nte if DTMF input is required.
  - The **no vad** command must be configured in the VoIP dial peer. The ASR server performs voice activity detection (VAD) so for accurate results, VAD should not be configured on the gateway.

- A separate G.711 u-law RTP stream (media forking) for speech recognition has the following platform restrictions:
  - It is supported on the Cisco 3660 only when the codec complexity is set to high on the voice card. It is not supported for medium complexity codecs.
  - It is not supported on the Cisco 1700 series.

Information About Speech Recognition and Synthesis

Cisco IOS Release 12.2(11)T and later releases supports automatic speech recognition (ASR) and text-to-speech (TTS) capabilities for VoiceXML and Tcl applications on Cisco voice gateways.

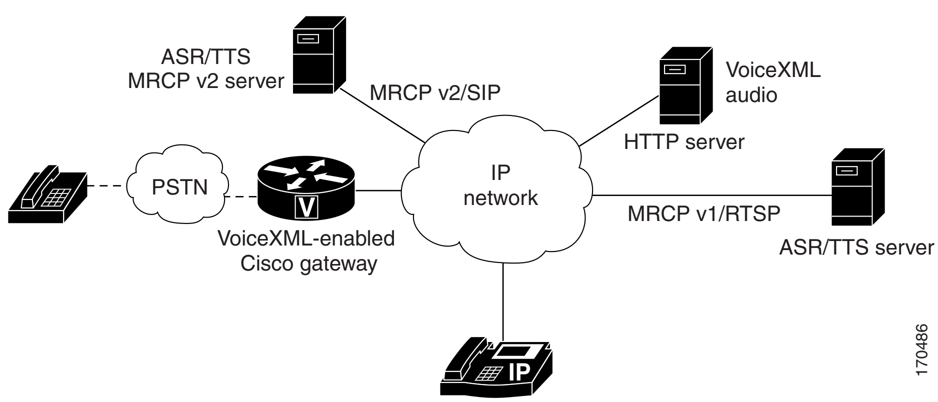
Cisco IOS Release 12.4(15)T adds support for Media Resource Control Protocol version 2 (MRCP v2) which depends on Session Initiation Protocol (SIP) to establish the control and media sessions between the Cisco gateway as a client and the MRCP v2 server.

The Speech Recognition and Synthesis feature provides interfaces to ASR and TTS media servers by using Media Resource Control Protocol (MRCP), an application-level protocol developed by Cisco and Nuance Communications. Client devices that are processing audio or video streams use MRCP to control media resources on external media servers, such as speech synthesizers for TTS and speech recognizers for ASR. The Cisco gateway, running a voice application, and the media servers providing speech recognition and speech synthesis, maintain a client/server relationship through an RTSP connection; the gateway is the RTSP client and the RTSP server is the streaming media server providing speech recognition and speech synthesis.

While doing speech recognition, the gateway creates a separate G.711 u-law RTP stream to the media server, enabling the gateway to simultaneously perform speech synthesis or play audio files using a different codec.

Figure 6-1 shows an example of a media server providing speech recognition and synthesis to a Cisco VoiceXML gateway.

Figure 6-1 Cisco IOS VoiceXML Network with ASR and TTS



Tcl IVR 2.0 and VoiceXML Integration

Tcl IVR 2.0 extensions in Cisco IOS Release 12.2(11)T allow Tcl applications to leverage support for ASR and TTS by invoking and managing VoiceXML-based dialogs within Tcl IVR scripts. This enables the implementation of hybrid applications using Tcl IVR for call control and VoiceXML for dialog management. For more information, see the *Cisco VoiceXML Programmer's Guide*.

How to Configure External Server Properties

- Specifying ASR and TTS Media Server Locations, page 5 (optional)
- Verifying the Media Server Locations, page 8 (optional)
- Troubleshooting ASR and TTS Server Functionality, page 8 (optional)
- Specifying MRCP v2 ASR and TTS Media Server Locations, page 10 (optional)
- Verifying the MRCP v2 Media Server Locations, page 12 (optional)
- Troubleshooting MRCP v2 ASR and TTS Server Functionality, page 14 (optional)
- Setting MRCP Client History Limits, page 17 (optional)
- ASR and TTS: Example, page 18 (optional)

Specifying ASR and TTS Media Server Locations

The location of media servers that are used for speech recognition and synthesis can be specified globally on the gateway, or these attributes can be specified in the individual VoiceXML document by using Cisco properties. These two methods have different results:

- **ivr tts-server** and **ivr asr-server** commands on the gateway—Media server sessions are created for each call to IVR applications, regardless of whether an application needs to talk to the media server.
- com.cisco.tt-server and com.cisco.asr-server <property> extensions—Media server sessions are created for each call to that application. If only a small number of applications require TTS/ASR media sessions, you should use the <property> extensions within those applications to define the external media server URL in the VXML script.

For information on identifying ASR or TTS servers through VoiceXML properties, see the *Cisco VoiceXML Programmer's Guide*.

Note Specifying the URL of media servers in a VoiceXML document takes precedence over the gateway configuration. Any value that is configured on the gateway is ignored if the same attribute is configured with a VoiceXML property.

Note If the asr and tts servers are in the same server, you have to differentiate them using the **ip host** command.
ip host asr-server <ip-address>
ip host tts-server <ip-address>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ivr asr-server** *url*
4. **ivr tts-server** *url*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Specify the ASR server location for VoiceXML documents:

```
ivr asr-server url
```

- *url*—Location of ASR resource on the media server in URL format.

```
Example: Router(config)# ivr asr-server rtsp://demo/recognizer
```

Note Nuance media servers using the default installation require the following syntax for the URL:
rtsp://*host***:***port***/recognizer**
(*host* is the host name of the media server; *port* is optional)

For media servers using MRCP v2, specify the URL as follows:
sip:*server-name@host-name* | *ip-address*

Step 4 Specify the TTS server location for VoiceXML documents:

```
ivr tts-server url
```

- *url*—Location of TTS resource on the media server in URL format.

```
Example: Router(config)# ivr tts-server rtsp://demo/synthesizer
```

Note Nuance media servers using the default installation require the following syntax for the URL:
rtsp://*host***:***port***/synthesizer**
(*host* is the host name of the media server; *port* is optional)

For media servers using MRCP v2, specify the URL as follows:
sip:*server-name@host-name* | *ip-address*

Troubleshooting Tips

If speech recognition or synthesis is not working, Table 6-2 lists some possible causes and the actions that you can take.

For additional debugging information on media servers, see the "Troubleshooting ASR and TTS Server Functionality" section on page 8.

Table 6-2 Speech Recognition or Synthesis Fails

Possible Causes	Suggested Actions
Server is not configured either on the Cisco gateway or in the VoiceXML document.	Verify that the server location is configured by using at least one of these methods: • Globally on the gateway by using the **ivr asr-server** or **ivr tts-server** command. See the "Verifying the Media Server Locations" section on page 8. • With the *com.cisco.asr-server* or *com.cisco.tts-server* property in the VoiceXML document. See the *Cisco VoiceXML Programmer's Guide*.
Gateway cannot access external ASR or TTS server or server is not running.	Ping the external server to make sure that the gateway has connectivity.
RTSP or MRCP errors are occurring between the gateway and the media server.	See the "Troubleshooting ASR and TTS Server Functionality" section on page 8.
MRCP v1 ports and URL defaults do not match.	Make the following changes to the Nuance configuration file. If you are using Nuance SpeechWorks MediaServer (SWMS), the configuration file is osserver.cfg. If you are using Nuance Speech Server (NSS), the configuration file is NSSserver.cfg. **TCP port:** Default: server.mrcp1.transport.port VXIInteger 4900 New value: server.mrcp1.transport.port VXIInteger **554** **TTS URL:** Default: server.mrcp1.resource.2.url VXIString media/speechsynthesizer New value: server.mrcp1.resource.2.url VXIString **/synthesizer** **ASR URL:** Default: server.mrcp1.resource.3.url VXIString media/speechrecognizer New value: server.mrcp1.resource.3.url VXIString **/recognizer**

Verifying the Media Server Locations

SUMMARY STEPS

1. **show running-config**
2. **show mrcp client session active**

DETAILED STEPS

Step 1 Use the **show running-config** command to display the media server configuration on your gateway, for example:

```
!
ivr asr-server rtsp://asr-server/recognizer
ivr tts-server rtsp://tts-server/synthesizer
!
```

Note The location of the external speech recognition or speech synthesis server can instead be configured in the VoiceXML document. The configuration in the VoiceXML document takes precedence over the gateway configuration.

Step 2 Make a call to an application using ASR or TTS and use the **show mrcp client session active** command to verify connectivity between the gateway and the media server, for example:

```
Router> show mrcp client session active

No Of Active MRCP Sessions:1

          Call-ID:0x1A
     Resource Type:Synthesizer   URL:rtsp://tts-server/synthesizer
Method In Progress:SPEAK State:SPEAKING
     Resource Type:Recognizer    URL:rtsp://asr-server/recognizer
Method In Progress:RECOGNIZE State:RECOGNIZING
```

Note The external media server has vendor-specific default and configurable parameters that control its functionality. These parameters are server-dependent and could affect the media server's ability to interoperate with the Cisco gateway. See your server manufacturer's documentation for details.

Troubleshooting ASR and TTS Server Functionality

SUMMARY STEPS

1. **debug vxml**
2. **debug mrcp error**
3. **debug rtsp**

DETAILED STEPS

Step 1 Use the **debug vxml error** and **debug vxml event** commands to verify that the external media server is reachable and its location is configured on the gateway or in the VoiceXML document. In the following example, the application failed because the media server is not configured on the gateway or in the VoiceXML document.:

```
Router# debug vxml error
Router# debug vxml event

*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_vapp_tts:
tftp://demo/sample/banking.vxml at line 17: vapp_tts() fail with vapp error 1
**Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_event_proc:
*Jan  5 18:24:19.507:    <event>: event=error.badfetch status=0
*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_default_event_handler:  use default
event handler
*Jan  5 18:24:19.507: //62/36CA25A68036/VAPP:/vapp_session_exit_event_name: Exit Event
error.badfetch
*Jan  5 18:24:19.507: //62/36CA25A68036/VAPP:/vapp_session_exit_event_name: Exit Event
error.badfetch
*Jan  5 18:24:19.507: //62/36CA25A68036/VAPP:/vapp_session_exit_event_name: Exit Event
Name already set to error.badfetch
*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_vapp_terminate: vapp_status=0 ref_count
0
*Jan  5 18:24:19.507: //62/36CA25A68036/VXML:/vxml_vapp_terminate: vxml session
terminating with code=ERROR

     vapp status=VAPP_SUCCESS  vxml async status=VXML_ERROR_BAD_FETCH
```

In the following example, the application failed because the media server is either unreachable or is not running.

```
*Jan  5 18:36:44.451: //83/ECD9B163804B/VXML:/vxml_media_done: : media play failed to
setup with VAPP error=31,
                 protocol_status_code=0
**Jan  5 18:36:44.451:   <event>: event=error.com.cisco.media.resource.unavailable
status=0
*Jan  5 18:36:44.451: //83/ECD9B163804B/VXML:/vxml_default_event_handler:  use default
event handler
*Jan  5 18:36:44.451: //83/ECD9B163804B/VAPP:/vapp_session_exit_event_name: Exit Event
error.com.cisco.media.resource.unavailable
*Jan  5 18:36:44.451: //83/ECD9B163804B/VAPP:/vapp_session_exit_event_name: Exit Event
error.com.cisco.media.resource.unavailable
*Jan  5 18:36:44.451: //83/ECD9B163804B/VAPP:/vapp_session_exit_event_name: Exit Event
Name already set to error.com.cisco.media.resource.unavailable
*Jan  5 18:36:44.451: //83/ECD9B163804B/VXML:/vxml_vapp_terminate: vapp_status=0 ref_count
0
```

Step 2 Use the **debug mrcp error** command to verify the connection between the gateway and the server. The following example shows the error if the RTSP connection to the server fails:

```
Router# debug mrcp error

*May  9 20:29:09.936:Connecting to 10.1.2.58:554 failed
```

The following error occurs if the response from the server is incorrect:

```
*May  9 20:29:09.936:Response from 10.1.2.58:554 failed
*May  9 20:29:09.936:MRCP/1.0 71 422 COMPLETE
```

The following error occurs if the recognize request comes out of sequence:

```
*May  9 20:29:09.936:act_idle_recognize:ignoring old recognize request
```

Step 3 Use the **debug rtsp error**, **debug rtsp session**, and **debug rtsp socket** commands to verify the RTSP connection with the media server, for example:

The following message displays if the RTSP connection fails:

```
*Sep 25 15:02:32.052: //-1//RTSP:/rtsplib_connect_to_svr: Socket Connect failed:
172.19.140.31:554
```

The following message displays if the RTSP client receives an incorrect response from the server:

```
*Sep 25 15:03:35.062: //-1//RTSP:/rtsp_process_single_svr_resp: Parse Server Response
failed, 172.19.140.31:554
```

The following message displays if the codec configured on the IP side is not G.711:

```
*Sep 25 15:05:15.765: //-1//RTSP:/rtsplib_rtp_associate_done: Association mismatch
```

Specifying MRCP v2 ASR and TTS Media Server Locations

The location of MRCP v2 media servers that are used for speech recognition and synthesis can be specified globally on the gateway, or these attributes can be specified in the individual VoiceXML document by using Cisco properties. These two methods have different results:

- **ivr tts-server** and **ivr asr-server** commands on the gateway—Media server sessions are created for each call to IVR applications, regardless of whether an application needs to talk to the media server.
- com.cisco.tt-server and com.cisco.asr-server <property> extensions—Media server sessions are created for each call to that application. If only a small number of applications require TTS/ASR media sessions, you should use the <property> extensions within those applications to define the external media server URL in the VoiceXML script.

For information on identifying MRCP v2 ASR or TTS servers through VoiceXML properties, see the *Cisco VoiceXML Programmer's Guide*.

Note Specifying the URL of media servers in a VoiceXML document takes precedence over the gateway configuration. Any value that is configured on the gateway is ignored if the same attribute is configured with a VoiceXML property.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class uri** *tag* **sip**
4. **pattern** *uri-pattern*
5. **exit**
6. **voice class uri** *tag* **sip**
7. **pattern** *uri-pattern*
8. **exit**
9. **dial-peer voice** *tag* **voip**

10. **description** *description*
11. **session protocol sipv2**
12. **destination uri** *tag*
13. **dtmf-relay rtp-nte**
14. **codec** {*codec* [**bytes** *payload_size*] | **transparent**}
15. **exit**
16. **Repeat steps 9 to 15 for the TTS server**
17. **ivr asr-server** *url*
18. **ivr tts-server** *url*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

`Example: Router# configure terminal`

Step 3 Create a voice class for matching dial peers to a SIP or telephone (TEL) uniform resource identifier (URI) for the ASR server:

voice class uri *tag* **sip**

`Example: Router(config)# voice class uri ASRserver sip`

Step 4 Match a call based on the entire SIP or TEL URI:

pattern *uri-pattern*

`Example: Router(config-voice-uri-class)# pattern mrcpv2ASRServer@10.5.18.22`

Step 5 Exit voice URI class configuration mode.

exit

`Example: Router(config-voice-uri-class)# exit`

Step 6 Create a voice class for matching dial peers to a SIP or telephone (TEL) URI for the TTS server:

voice class uri *tag* **sip**

`Example: Router(config)# voice class uri TTSserver sip`

Step 7 Match a call based on the entire SIP or TEL URI:

pattern *uri-pattern*

`Example: Router(config-voice-uri-class)# pattern mrcpv2TTSServer@10.5.18.22`

Step 8 Exit voice URI class configuration mode.

exit

`Example: Router(config-voice-uri-class)# exit`

Step 9 Define a particular dial peer for the MRCP v2 ASR server termination, specify the method of voice encapsulation, and enter dial-peer configuration mode:

dial-peer voice *tag* **voip**

`Example: Router(config)# dial-peer voice 2234 voip`

Step 10 Add a description to a dial peer:

description *description*

- *description*—Text string up to 64 alphanumeric characters.

`Example: Router(config-dial-peer)# description SIP ASR Media Call`

Step 11 Specify a session protocol for calls between local and remote routers using the packet network:

session protocol sipv2

`Example: Router(config-dial-peer)# session protocol sipv2`

Step 12 Specify the voice class used to match a dial peer to the destination URI of an outgoing call:

destination uri *tag*

`Example: Router(config-dial-peer)# destination uri myASR`

Step 13 Specify how an H.323 or SIP gateway relays dual-tone multifrequency (DTMF) tones between telephony interfaces and an IP network:

dtmf-relay rtp-nte

- rtp-nte—Forwards DTMF tones by using Real-Time Transport Protocol (RTP) with the Named Telephone Event (NTE) payload type.

`Example: Router(config-dial-peer)# dtmf-relay rtp-nte`

Step 14 Specify the voice codec rate of speech for a dial peer:

codec {*codec* [**bytes** *payload_size*] | **transparent**}

- *codec*—**g711ulaw** or **g711alaw.** Only the g711ulaw and g711alaw codecs are supported for ASR and TTS media service.

`Example: Router(config-dial-peer)# codec g711ulaw`

Step 15 Exit dial-peer configuration mode.

exit

`Example: Router(config-dial-peer)# exit`

Step 16 Repeat steps 9 to 15 for the TTS server.

Step 17 Specify the MRCP v2 ASR server location for VoiceXML documents:

ivr asr-server *url*

- *url*—Location of ASR resource on the media server in URL format. For media servers using MRCP v2, specify the URL as follows:
  sip:*server-name@host-name* | *ip-address*

`Example: Router(config)# ivr asr-server sip:mrcpv2ASRServer@10.5.18.22`

Step 18 Specify the MRCP v2 TTS server location for VoiceXML documents:

ivr tts-server *url*

- *url*—Location of TTS resource on the media server in URL format. For media servers using MRCP v2, specify the URL as follows:
  sip:*server-name@host-name* | *ip-address*

`Example: Router(config)# ivr tts-server sip:mrcpv2TTSServer@1.5.18.22`

Verifying the MRCP v2 Media Server Locations

SUMMARY STEPS

1. **show running-config**
2. **show mrcp client session active detailed**
3. show call active media compact

DETAILED STEPS

Step 1 Use the **show running-config** command to display the media server configuration on your gateway, for example:

```
!
```

```
voice class uri testASR sip

pattern mrcpv2ASRServer@10.5.18.22
!
voice class uri testTTS sip
pattern mrcpv2TTSServer@10.5.18.22
!
!
ivr asr-server sip:mrcpv2ASRServer@10.5.18.224
ivr tts-server sip:mrcpv2TTSServer@10.5.18.224
!
dial-peer voice 2234 voip
description SIP ASR Media Call
session protocol sipv2
destination uri testASR
dtmf-relay rtp-nte
codec g711ulaw
no vad
!
dial-peer voice 2235 voip
description SIP TTS Media Call
session protocol sipv2
destination uri testTTS
dtmf-relay rtp-nte
codec g711ulaw
no vad
!
```

Note The location of the external speech recognition or speech synthesis server can instead be configured in the VoiceXML document. The configuration in the VoiceXML document takes precedence over the gateway configuration.

Step 2 Make a call to an application using ASR or TTS and use the **show mrcp client session active detailed** command to verify connectivity between the gateway and the media server, for example:

```
Router# show mrcp client session active detailed

No Of Active MRCP Sessions: 1
Call-ID: 0x14 same: 0
--------------------------------------------
Resource Type: Synthesizer URL: sip:mrcpv2TTSServer@10.5.18.224
Method In Progress: SPEAK State: S_SYNTH_IDLE
Associated CallID: 0x17
MRCP version: 2.0
Control Protocol: TCP Server IP Address: 10.5.18.224 Port: 51000
Data Protocol: RTP Server IP Address: 10.5.18.224 Port: 10000
Signalling URL: sip:mrcpv2TTSServer@10.5.18.224:5060
Packets Transmitted: 0 (0 bytes)
Packets Received: 177 (28320 bytes)
ReceiveDelay: 100 LostPackets: 0
--------------------------------------------
--------------------------------------------
Resource Type: Recognizer URL: sip:mrcpv2ASRServer@10.5.18.224
Method In Progress: RECOGNITION-START-TIMERS State: S_RECOG_RECOGNIZING
Associated CallID: 0x18
MRCP version: 2.0
Control Protocol: TCP Server IP Address: 10.5.18.224 Port: 51001
Data Protocol: RTP Server IP Address: 10.5.18.224 Port: 10002
Packets Transmitted: 191 (30560 bytes)
Packets Received: 0 (0 bytes)
ReceiveDelay: 100 LostPackets: 0
```

Note The external media server has vendor-specific default and configurable parameters that control its functionality. These parameters are server-dependent and could affect the media server's ability to interoperate with the Cisco gateway. See your server manufacturer's documentation for details.

Step 3 Use the **show call active media compact** command to display call information for media calls in progress.

```
Router# show call active media comp
<callID> A/O FAX T<sec> Codec type Peer Address IP R<ip>:<udp>
Total call-legs: 2
4 ORG T3 g711ulaw VOIP-MEDIA Psip:MRCPv2Server@1.5.18.224:5060 1.5.18.224:10000
5 ORG T3 g711ulaw VOIP-MEDIA Psip:MRCPv2Server@1.5.18.224:5060 1.5.18.224:10002
```

Troubleshooting MRCP v2 ASR and TTS Server Functionality

SUMMARY STEPS

1. **debug voip application vxml**
2. **debug mrcp error**
3. **debug voip application callsetup**
4. **show call active media**
5. **show call history media**

DETAILED STEPS

Step 1 Use the **debug voip application vxml error**, **debug voip application vxml event** and **debug voip application vxml application** commands to verify that the external MRCP v2 media server is reachable and its location is configured on the gateway or in the VoiceXML document. In the following example, the application failed because the media server is not configured on the gateway or in the VoiceXML document.:

```
Router# debug voip application vxml error
Router# debug voip application vxml event
Router# debug voip application vxml application

Jul 12 12:37:42.067: //-1//AFW_:/AFW_VxmlModule_New:
Jul 12 12:37:42.091: //9//AFW_:/vapp_get_type_detail:
Jul 12 12:37:42.091: //9//AFW_:/vapp_get_incoming_gtd_list:
Jul 12 12:37:42.095: //9//AFW_:/vapp_tts:
Jul 12 12:37:42.095: //9//AFW_:/vapp_tts: bargein=1
Jul 12 12:37:42.095: //9//AFW_:/vapp_tts: timeout=0
Jul 12 12:37:42.095: //9/0B92138D8004/VXML:/vxml_vapp_tts:
CALL_ERROR; tftp://dirt/dileung/jeff-grammar.vxml
at line 20: fail with vapp error 2
Jul 12 12:37:42.099: //9//AFW_:/vapp_terminate:
Jul 12 12:37:42.099: //9//AFW_:/vapp_session_exit_event_name: Exit Event
vxml.session.complete
Jul 12 12:37:42.099: //9//AFW_:/AFW_M_VxmlModule_Terminate:
Jul 12 12:37:42.099: //9//AFW_:/vapp_checksessionstate:
Jul 12 12:37:42.099: //9//AFW_:/vapp_checkifdone: Object: 1, Leg: 1
Jul 12 12:37:42.099: //9//AFW_:/vapp_checksessionstate:
Jul 12 12:37:42.099: //9//AFW_:/vapp_checkifdone: Object: 0, Leg: 0
```

```
Jul 12 12:37:42.227: //-1//AFW_:HN000F5B84:/AFW_M_VxmlModule_Free:
Jul 12 12:37:42.227: MOD[VxmlModule_63CD2B68_0_1006468] ( )
```

Step 2 Use the **debug mrcp error** command to display debugging messages for MRCP v1 and v2 operations. In the following example, the server message request timer expired:

```
Router# debug mrcp error

*May 26 06:00:13.066: //31//MRCP:/mrcpv2_process_timers:
ERROR: MRCPv2 Timer Expired; TimerType:[MRCPV2_SVR_MSG_MGD_TIMER(21)]
FSM: Type=[RECOGNIZER(2),FSM_TYPE_REMOTE_SERVER(3)]
(debug mrcp error : Fail to establish TCP connection with server)
Jul 6 14:53:38.199: //30//MRCP:/mrcpv2_handle_socket_read:
Socket Connection Error; Socket=0,errno=-1
(debug mrcp error : Fail to send control message to server)
Jul 6 14:53:38.223: //30//MRCP:/mrcpv2_partial_socket_send:
Socket Send Failed; fd=1:1.5.18.224:51001, errno=-1
```

Step 3 Use the **debug voip application callsetup** command to display all application debug messages. In the following example, there is no match on the VoIP SIP dial peer for the MRCP v2 server:

```
Router# debug voip application callsetup

Jul 12 14:47:38.019: //48//Call:/AFW_CallSetup_AddDest: sip:mrcpv2TTSServer@1.5.18.224
Jul 12 14:47:38.019: //48//Call:/AFW_M_CallSetup_Initiate:
Jul 12 14:47:38.019: //48//Call:/CallSetupInitiate:
Jul 12 14:47:38.019: //48//Call:/CS_Placecall:
Jul 12 14:47:38.019: //-1//Dest:/AFW_Destination_New:
Jul 12 14:47:38.019: //48//Dest:/AFW_Destination_AddDest:
Jul 12 14:47:38.019: //-1//Dest:/AFW_Destination_AddDest: adding destination
"sip:mrcpv2TTSServer@1.5.18.224"
Jul 12 14:47:38.019: //48//Dest:/AFW_M_Destination_Initiate:
Jul 12 14:47:38.019: //48//Dest:/AFW_M_Destination_Initiate: Outgoing guid :
324C5D9B.10EC11DB.8011000B.5FDA0EF4
Jul 12 14:47:38.019: Incoming_guid : 00000000.00000000.00000000.00000000
Jul 12 14:47:38.019: //48//Dest:/DestSetupInitiate:
Jul 12 14:47:38.019: //48//Dest:/DestMatchDialPeer:
Jul 12 14:47:38.019: //48//Dest:/DestMatchDialPeer: src carrier id:test5, tgt carrier id:
Jul 12 14:47:38.019: //48//Dest:/DestSetupInitiate: Did not match any peers
Jul 12 14:47:38.019: //48//Dest:/DestResetCallInfo:
Jul 12 14:47:38.019: //48//Dest:/DestComplete: peer #:0 IW State IW_STATE_INIT, OB State
OB_STATE_INIT
Jul 12 14:47:38.019: //-1//Dest:/DestStatusFromDiscCause: mapped "unassigned number
(1)"(1) to DEST_INVALID_NUMBER(4)
Jul 12 14:47:38.019: //48//Dest:/DestReturn: Destination Returning(ds_004 Status
DEST_INVALID_NUMBER)
Jul 12 14:47:38.019: //48//Call:/CS_Placecall: Call placed to
sip:mrcpv2TTSServer@1.5.18.224
Jul 12 14:47:38.023: //48//Dest:/AFW_M_Destination_EventPreProcess:
Jul 12 14:47:38.023: //48//Call:/AFW_M_CallSetup_Action:
Jul 12 14:47:38.023: //48//Call:/CS_Placecall_DestDone:
Jul 12 14:47:38.023: //48//Call:/CS_Complete: CallSetup Returning(ls_004 Status
CS_INVALID_NUMBER)
```

Step 4 Use the **show call active media** command to display call information for media calls in progress. The following example displays the SIP media call leg that is used to establish the SIP session between the Cisco gateway, which is the client, and the MRCP v2 server:

```
Router# show call active media brief

<ID>: <CallID> <start>hs.<index> +<connect> pid:<peer_id> <dir> <addr> <state>
dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes>
IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms lost:<lost>/<early>/<late>
delay:<last>/<min>/<max>ms <codec>
```

```
media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>

long duration call detected:<y/n> long duration call duration :<sec> timestamp:<time>
MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
last <buf event time>s dur:<Min>/<Max>s
FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
Tele <int> (callID) [channel_id] tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l>
i/o:<l>/<l> dBm
MODEMRELAY info:<rcvd>/<sent>/<resent> xid:<rcvd>/<sent> total:<rcvd>/<sent>/<drops>
speeds(bps): local <rx>/<tx> remote <rx>/<tx>
Proxy <ip>:<audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
bw: <req>/<act> codec: <audio>/<video>
tx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
Telephony call-legs: 0
SIP call-legs: 0
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Media call-legs: 2
Total call-legs: 2
11EE : 57 10092920ms.1 +140 pid:2235 Originate sip:mrcpv2TTSServer@1.5.18.224:5060 active
dur 00:00:06 tx:0/0 rx:316/50560
IP 1.5.18.224:10000 SRTP: off rtt:0ms pl:2000/3520ms lost:0/0/0 delay:100/95/100ms
g711ulaw TextRelay: off

media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
11EE : 58 10092940ms.1 +160 pid:2234 Originate sip:mrcpv2ASRServer@1.5.18.224:5060 active
dur 00:00:06 tx:328/52480 rx:0/0
IP 1.5.18.224:10002 SRTP: off rtt:0ms pl:2000/3380ms lost:0/0/0 delay:100/95/100ms
g711ulaw TextRelay: off

media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long duration call duration:n/a timestamp:n/a
Telephony call-legs: 0
SIP call-legs: 0
H323 call-legs: 0
Call agent controlled call-legs: 0
SCCP call-legs: 0
Multicast call-legs: 0
Media call-legs: 2
Total call-legs: 2
```

Step 5 Use the **show call history media** command to display the call history table for media calls. The following example displays the SIP media call leg that is used to establish the SIP session between the Cisco gateway, which is the client, and the MRCP v2 server:

```
Router# show call history media brief

<ID>: <CallID> <start>hs.<index> +<connect> +<disc> pid:<peer_id> <direction> <addr>
dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes> <disc-cause>(<text>)
IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms lost:<lost>/<early>/<late>
delay:<last>/<min>/<max>ms <codec>

media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>

long duration call detected:<y/n> long duration call duration :<sec> timestamp:<time>
MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
```

```
last <buf event time>s dur:<Min>/<Max>s
FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
<codec> (payload size)
Telephony <int> (callID) [channel_id] tx:<tot>/<voice>/<fax>ms <codec> noise:<lvl>dBm
acom:<lvl>dBm
MODEMRELAY info:<rcvd>/<sent>/<resent> xid:<rcvd>/<sent> total:<rcvd>/<sent>/<drops>
disc:<cause code>
speeds(bps): local <rx>/<tx> remote <rx>/<tx>
Proxy <ip>:<audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
bw: <req>/<act> codec: <audio>/<video>
tx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
Telephony call-legs: 0
SIP call-legs: 0
H323 call-legs: 0
Call agent controlled call-legs: 0
Media call-legs: 2
Total call-legs: 2
11EE : 57 10092920ms.21 +140 +58380 pid:2235 Originate sip:mrcpv2TTSServer@1.5.18.224:5060
dur 00:00:58 tx:0/0 rx:2918/466880 10 (normal call clearing (16))
IP 1.5.18.224:10000 SRTP: off rtt:0ms pl:19000/50540ms lost:0/0/0 delay:100/35/100ms
g711ulaw TextRelay: off

media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long dur callduration :n/a timestamp:n/a
11EE : 58 10092940ms.22 +160 +58360 pid:2234 Originate sip:mrcpv2ASRServer@1.5.18.224:5060
dur 00:00:58 tx:2862/457920 rx:0/0 10 (normal call clearing (16))
IP 1.5.18.224:10002 SRTP: off rtt:0ms pl:19000/49720ms lost:0/0/0 delay:100/35/100ms
g711ulaw TextRelay: off

media inactive detected:n media contrl rcvd:n/a timestamp:n/a
long duration call detected:n long dur callduration :n/a timestamp:n/a
```

Setting MRCP Client History Limits

Media Resource Control Protocol (MRCP) is used for controlling media resources such as speech synthesizers for TTS and speech recognizers for ASR. The Cisco voice gateway can store records of previous MRCP sessions that are no longer active. The maximum number of history records and the length of time that the records are stored can also be configured on the gateway.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **mrcp client history records** *number*
4. **mrcp client history duration** *seconds*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

`enable`

`Example: Router> enable`

Enter your password if prompted.

Step 2 Enter global configuration mode:

`configure terminal`

`Example: Router# configure terminal`

Step 3 Set the maximum number of records for inactive MRCP sessions that the gateway can store:

`mrcp client history records` *`number`*

- *number*—Maximum number of MRCP history records to save. If 0 is configured, no MRCP records are stored on the gateway. The maximum value is platform-specific. The default value is 50.

`Example: Router(config)# mrcp client history records 100`

Step 4 Set the maximum number of seconds that the history records of MRCP sessions are stored on the gateway:

`mrcp client history duration` *`seconds`*

- *seconds*—Maximum number of seconds that MRCP history records are stored. If 0 is configured, no MRCP records are stored on the gateway. Range is from 0 to 99,999,999. The default value is 3600.

`Example: Router(config)# mrcp client history duration 7200`

Note The default values for the **mrcp client history records** and **mrcp client history duration** commands are not platform-specific.

Configuration Examples for ASR and TTS

This section provides gateway configuration examples of VoiceXML applications using ASR and TTS.

- ASR and TTS: Example, page 18
- MRCP v2 ASR and TTS: Example, page 20

ASR and TTS: Example

The following is an example configuration for MRCP v1 ASR and TTS.

```
!
version 12.3
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname lab13
!
enable secret 5 $1$aBC123
enable password sample
!
voice-card 1
!
ip subnet-zero
no ip routing
!
!
no ip domain-lookup
ip host tftp-server1 10.1.2.2
!
isdn switch-type primary-ni
```

```
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
ivr asr-server rtsp://10.1.2.52/recognizer
ivr tts-server rtsp://10.1.2.52/synthesizer
fax interface-type fax-mail
mta receive maximum-recipients 0
!
controller T1 1/0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1/1
 shutdown
 framing sf
 linecode ami
!
!
!
!
interface FastEthernet0/0
 ip address 10.1.2.228 255.255.0.0
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
 no cdp enable
!
interface FastEthernet0/1
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
interface Serial1/0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice voice
 no cdp enable
!
ip classless
ip http server
ip pim bidir-enable
!
!
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
no cdp run
!
!
call rsvp-sync
!
application
 service rdnis tftp://10.1.2.2/demo/sample/test27535.vxml
 !
```

```
 service event tftp://10.1.2.2/demo/sample/test51565-2-1.0.vxml
 !
 service myapp tftp://10.1.2.2/demo/scr/func/A_1.vxml
 !
!
voice-port 1/0:23
!
voice-port 2/1/0
!
voice-port 2/1/1
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 10 pots
 service myapp
 incoming called-number 50202
 port 1/0:23
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
 password sample
 login
!
end
```

MRCP v2 ASR and TTS: Example

The following is an example configuration for MRCP v2 ASR and TTS.

```
!
! Last configuration change at 16:24:12 UTC Sat Jun 28 2006
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname rtr2800
!
boot-start-marker
boot-end-marker
!
card type e1 0 3
logging buffered 10000000 debugging
no logging console
!
no aaa new-model
!
resource policy
!
no network-clock-participate wic 0
network-clock-participate wic 2
network-clock-participate wic 3
```

```
!
!
ip cef
!
!
ip host dirt 223.255.254.253
!
voice class uri testasr sip
pattern MRCPv2ASRServer@10.5.24.1
!
!
voice class uri testtts sip
pattern MRCPv2TTSServer@10.5.24.1
!
ivr asr-server sip:MRCPv2ASRServer@10.5.24.100
ivr tts-server sip:MRCPv2TTSServer@10.5.24.100
!
application
service myservice tftp://dir1/dir2/myservice.vxml
!
!
interface Serial3/1:23
no ip address
encapsulation hdlc
isdn switch-type primary-5ess
isdn incoming-voice voice
no cdp enable
!
!
controller E1 0/0/0
!
controller E1 0/0/1
framing NO-CRC4
clock source internal
!
controller E1 0/3/0
clock source internal
pri-group timeslots 1-31
!
interface GigabitEthernet0/0
ip address 10.5.14.6 255.255.0.0
duplex auto
speed auto
!
interface GigabitEthernet0/1
ip address 172.16.156.23 255.255.0.0
duplex auto
speed auto
!
interface Serial0/3/0:15
no ip address
encapsulation hdlc
no logging event link-status
dialer pool-member 1
isdn switch-type primary-net5
isdn protocol-emulate network
isdn T310 30000
isdn integrate calltype all
!
!
voice-port 0/3/0:15
!
!
dial-peer voice 2323 voip
```

```
session protocol sipv2
destination uri testasr
dtmf-relay rtp-nte
codec g711ulaw
no vad
!
dial-peer voice 2324 voip
session protocol sipv2
destination uri testtts
dtmf-relay rtp-nte
codec g711ulaw
no vad
!
dial-peer voice 3838 pots
service jeff1
incoming called-number 3838
direct-inward-dial
!
dial-peer voice 38381 voip
destination-pattern 3838
session target ipv4:10.5.14.13
!
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl IVR and VoiceXML Applications" on page 1.
- To record voice messages using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward" on page 1.
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML" on page 1.
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features" on page 1.
- To configure session interaction for a Tcl IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- "" on page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release.
- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance.

Configuring Fax Detection for VoiceXML

This chapter explains how to configure the Fax Detection for VoiceXML feature.

For more information about this and related Cisco IOS voice features, see the following:

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for Fax Detection for VoiceXML

Release	Modification
12.2(2)XB	This feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.

Contents

- Prerequisites for Fax Detection for VoiceXML, page 2
- Restrictions for Fax Detection for VoiceXML, page 2
- Information About Fax Detection for VoiceXML, page 2
- How to Configure Fax Detection for VoiceXML, page 3
- Configuration Examples for Fax Detection for VoiceXML, page 6
- Where to Go Next, page 8
- Additional References, page 9

When developing and configuring a voice application, refer to this chapter and to the *Cisco VoiceXML Programmer's Guide* or the *Tcl IVR API Version 2.0 Programmer's Guide*.

Prerequisites for Fax Detection for VoiceXML

- You must configure basic VoiceXML application functionality as described in "Configuring Basic Functionality for Tcl IVR and VoiceXML Applications" on page 1.
- You must write a VoiceXML document that implements fax detection. To write your own script, refer to the *Cisco VoiceXML Programmer's Guide*.

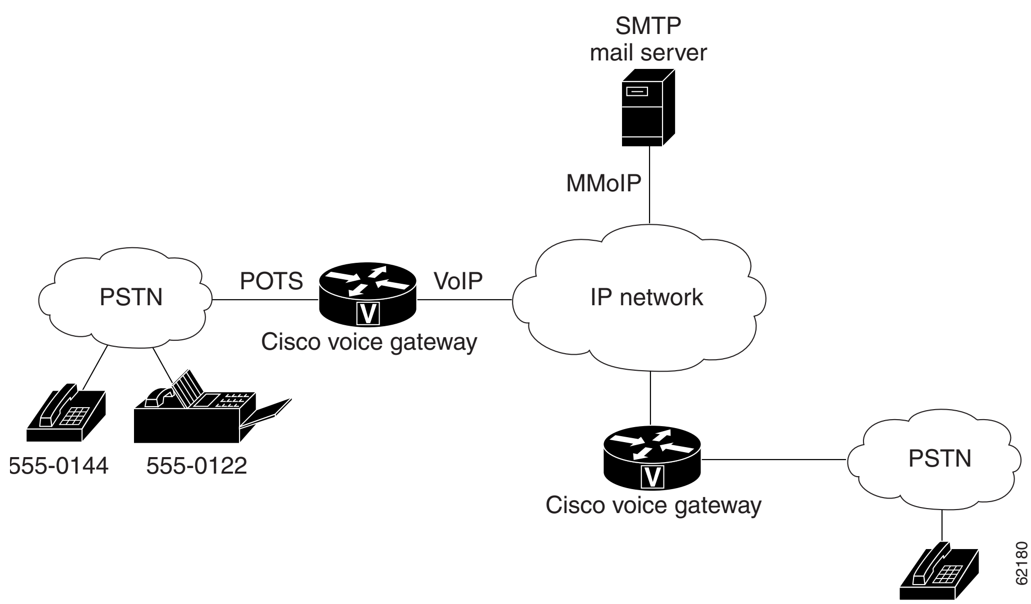
Restrictions for Fax Detection for VoiceXML

- In Cisco IOS Release 12.2(2)XB, fax detection for VoiceXML is supported only on the Cisco AS5300.

Information About Fax Detection for VoiceXML

When a VoiceXML fax detection application is configured on the gateway, callers can dial a single number for both voice and fax calls. The gateway automatically detects when a call is a fax transmission by listening for a CNG (CalliNG) tone, the distinctive fax calling tone. The Cisco IOS VoiceXML gateway, when configured for fax detection, continuously listens to incoming calls to determine which calls are voice or fax. The gateway then routes calls to the appropriate application or media server.

Figure 7-1 Fax Detection on Cisco VoiceXML Gateway



Note In Cisco IOS Release 12.2(2)XB, VoiceXML fax detection is supported only on the Cisco AS5300.

After a call is established, the VoiceXML application can play an audio prompt to the caller while waiting for CNG detection. CNG detection continues for the entire duration of the call, so it is possible that a caller could first be connected to a voice-mail server and leave a voice message, then start to transmit a fax and the application would automatically switch the call to the fax application. After the application detects whether a call is voice or fax, the gateway routes the call based on dial peers. The fax detection application requires at least two dial peers:

- Inbound POTS dial peer, for incoming calls from the PSTN
- Outbound MMoIP dial peer for store-and-forward fax, to send fax transmissions to an e-mail server

For a general description of dial peers, see the "Role of Dial Peers in Configuring Voice Applications" section on page 40.

For information on configuring fax relay or store-and-forward fax, refer to the *Cisco IOS Fax Services over IP Application Guide,* Release 12.3.

How to Configure Fax Detection for VoiceXML

- Configuring Fax Detection for VoiceXML, page 3 (required)
- Verifying VoiceXML Fax Detection Configuration, page 4 (optional)

Configuring Fax Detection for VoiceXML

Step 1 Load your fax detection application onto the gateway. For detailed instructions, see the "Loading a Service onto the Gateway" section on page 25.

```
Example:
application
 service vxml_fax_detect tftpboot://172.16.1.1/scripts/fax_detect.vxml
```

Note You must write your own VoiceXML document that implements fax detection. Cisco does not provide a VoiceXML script that you can download.

Step 2 Configure the inbound POTS dial peer. For detailed instructions, see the "Configuring an Inbound Application" section on page 40.

```
Example:
!
dial-peer voice 5 pots
 service vxml_fax_detect
 incoming called-number 55..
```

When DID is enabled, the incoming called number (DNIS) is used to match the destination pattern of outgoing MMoIP dial peers.

Step 3 Configure the outbound MMoIP dial peer and the sending MTA for T.37 store-and-forward fax. For detailed instructions, refer to the *Cisco IOS Fax Services over IP Application Guide,* Release 12.3.

```
Example:
!
fax interface-type fax-mail
mta send server 172.16.1.25
mta send subject Test Message
mta send origin-prefix Cisco Fax
mta send postmaster postmaster@mail-server.com
mta send mail-from hostname zebra.unified-messages.com
mta send mail-from username $s$
```

```
mta send return-receipt-to hostname zebra.unified-messages.com
mta send return-receipt-to username $s$
!
!
dial-peer voice 555 mmoip
 service fax_on_vfc_onramp_app out-bound
 destination-pattern 55..
 information-type fax
 session target mailto:$e$@mail-server.com
```

The name of the application for store-and-forward fax is **fax_on_vfc_onramp_app**. This application is included in the Cisco IOS software; you do not have to download it or configure it by using the application configuration submodes.

The **session target mailto** command specifies the address of the mail server where faxes are e-mailed. The **e** macro indicates that the VoiceXML application sets the mailto address to either the DNIS, RDNIS, or a string representing a valid e-mail address. The "cisco-mailtoaddress" variable in the transfer tag of the VoiceXML fax detection application replaces the **e** in the mailto address. The VoiceXML application maps the "cisco-mailtoaddress" variable to the e-mail address only if the **e** macro is configured in the MMoIP dial peer. By default, if the "cisco-mailtoaddress" variable is not specified in the transfer tag, the VoiceXML application maps the DNIS to **e**.

If **e** is not specified in the **session target mailto** command, but the *cisco-mailtoaddress* variable is specified in the transfer tag of the fax detection document, then whatever is specified in the MMoIP dial peer takes precedence; the "cisco-mailtoaddress" variable is ignored.

For information about the *cisco-mailtoaddress* transfer tag variable, refer to the *Cisco VoiceXML Programmer's Guide*.

For information about configuring fax detection using Tcl applications, refer to the *Cisco IOS Fax Services over IP Application Guide,* Release 12.3.

Verifying VoiceXML Fax Detection Configuration

SUMMARY STEPS

1. **show running-config**
2. **show call application voice summary**
3. **show call application voice**
4. **show dial-peer voice** *tag*

DETAILED STEPS

Step 1 Use the **show running-config command** to verify the application's configuration parameters.

```
application
 service vxml_fax_detect tftp://10.1.1.1/scripts/vxml_fax_detect.vxml
  !
!
dial-peer voice 55 pots
 service vxml_fax_detect
 incoming called-number 55..
 direct-inward-dial
```

Step 2 Use the **show call application voice summary** command to list all voice applications loaded on the router to confirm that the fax detection document is loaded. In the sample output below, the VoiceXML document for the fax detection application vxml_fax_detect is loaded from a TFTP server.

```
Router# show call application voice summary

name                 description

session              Basic app to do DID, or supply dialtone.
fax_hop_on           Script to talk to a fax redialer
clid_authen          Authenticate with (ani, dnis)
clid_authen_collect  Authenticate with (ani, dnis), collect if that fails
clid_authen_npw      Authenticate with (ani, NULL)
clid_authen_col_npw  Authenticate with (ani, NULL), collect if that fails
clid_col_npw_3       Authenticate with (ani, NULL), and 3 tries collecting
clid_col_npw_npw     Authenticate with (ani, NULL) and 3 tries without pw
DEFAULT              Default system session application
lib_off_app          Libretto Offramp
fax_on_vfc_onramp_ap Fax onramp for VFC
vxml_fax_detect      tftp://server/tftpboot/scripts/vxml/vxml_fax_detect.vxml
offramp_mapp         tftp://server/tftpboot/vrsf/tcl/app_voicemail_offramp.tcl

TCL Script Version 2.0 supported.
TCL Script Version 1.1 supported.
Voice Browser Version 2.0 for VoiceXML 1.0 & 2.0 supported.
```

Step 3 Use the **show call application voice** command to display the line-by-line contents of the VoiceXML document associated with a specific loaded application.

Step 4 Use the **show dial-peer voice** command to verify that the operational status of a dial peer is up.

```
Router# show dial-peer voice 55

VoiceEncapPeer55
        information type = voice,
        description = `',
        tag = 55, destination-pattern = `',
        answer-address = `', preference=0,
        numbering Type = `unknown'
        group = 55, Admin state is up, Operation state is up,
        incoming called-number = `55..', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
        huntstop = disabled,
        in bound application associated: 'vxml_fax_detect'
        out bound application associated: ''
        dnis-map =
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        type = pots, prefix = `',
        forward-digits default
        session-target = `', voice-port = `0:D',
        direct-inward-dial = disabled,
        digit_strip = enabled,
        register E.164 number with GK = TRUE

        Connect Time = 0, Charged Units = 0,
        Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
        Accepted Calls = 0, Refused Calls = 0,
        Last Disconnect Cause is "",
        Last Disconnect Text is "",
        Last Setup Time = 0.
```

Configuration Examples for Fax Detection for VoiceXML

This section provides a gateway configuration example of a VoiceXML application using fax detection.

- Fax Detection for VoiceXML with T.37 Store-and-Forward Fax Example, page 6

Fax Detection for VoiceXML with T.37 Store-and-Forward Fax Example

This example is a basic configuration for fax detection for incoming calls. The application is called vxml_fax_detect on the gateway.

```
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime localtime
no service password-encryption
!
hostname zebra
!
resource-pool disable
!
clock timezone AEST 10
ip subnet-zero
!
isdn switch-type primary-5ess
call rsvp-sync
!
!VXML application configuration for Fax Detection
application
service vxml_fax_detect tftp://10.1.1.1/scripts/vxml_fax_detect.vxml
!
!
cns event-service server
!
fax receive called-subscriber $d$
fax send transmitting-subscriber $s$
fax send left-header $s$
fax send center-header $t$
fax send right-header Page $p$
fax send coverpage enable
fax send coverpage email-controllable
fax send coverpage comment Cisco cover page comment
fax interface-type fax-mail
mta send server 172.16.1.25
mta send subject Test Message
mta send origin-prefix Cisco Fax
mta send postmaster postmaster@mail-server.unified-messages.com
mta send mail-from hostname zebra.unified-messages.com
mta send mail-from username $s$
mta send return-receipt-to username $s$
mta receive aliases sydney.com
mta receive maximum-recipients 120
mta receive generate-mdn
!
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
```

```
controller T1 1
 framing esf
 clock source line secondary 1
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 2
 framing esf
 clock source line secondary 2
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 3
 clock source line secondary 3
!
interface Ethernet0
 ip address 10.2.14.90 255.0.0.0
!
interface Serial0
 no ip address
 no ip mroute-cache
 shutdown
 no fair-queue
 clockrate 2015232
!
interface Serial1
 no ip address
 shutdown
 no fair-queue
 clockrate 2015232
!
interface Serial0:23
 no ip address
 ip mroute-cache
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 isdn T203 10000
 no cdp enable
!
interface Serial1:23
 no ip address
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 no cdp enable
!!
interface FastEthernet0
 ip address 172.16.14.90 255.255.0.0
 duplex auto
 speed auto
 h323-gateway voip interface
 h323-gateway voip h323-id 5300-voip
 h323-gateway voip tech-prefix 2#
!
ip classless
no ip http server
!
voice-port 0:D
!
voice-port 1:D
!
voice-port 2:D
!
!POTS dial-peer configuration for VoiceXML Fax Detection
dial-peer voice 1 pots
```

```
 service fax_detect
 incoming called-number 5...
 direct-inward-dial
!
!Mail dial-peer configuration for T.37 store-and-forward fax
dial-peer voice 3 mmoip
 service fax_on_vfc_onramp_app out-bound
 destination-pattern 5...
 information-type fax
 session target mailto:$e$@mail-server.com
!
!
gateway
!
!
line con 0
 exec-timeout 0 0
 transport input none
line aux 0
line vty 0 4
 login
!
ntp clock-period 17180419
ntp source Ethernet0
ntp server 10.1.1.1
end
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl IVR and VoiceXML Applications" on page 1.
- To record voice messages using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward" on page 1.
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties" on page 1.
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features" on page 1.
- To configure session interaction for a Tcl IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- "" on page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance

Configuring Media Inactive Call Detection (Silent Call Detection)

The Media Inactive Call Detection feature enhances Cisco IOS behavior for disconnecting a call when an inactive condition is detected. Prior to Cisco IOS Release 12.4(4)T, the behavior automatically disconnected inactive calls when notification was not configured. This original behavior is retained in Cisco IOS Release 12.4(4)T, but more control has been added for managing these calls by enhancing notification of inactive calls, depending on the particular application.

The Media Inactive Call Detection feature detects inactive (silent) H.323 or SIP call legs on Cisco IOS-based gateways, and reports this situation to the Tcl IVR 2.0 application (which can disconnect the call). When the Media Inactive Call Detection feature is enabled through application CLI (see the "Configuring CallFeature Parameters for Media Inactive Call Detection" section on page 7), Cisco IOS software does not automatically disconnect detected inactive calls.

History for the Media Inactive Call Detection (Silent Call Detection) Feature

Release	Modification
12.4(4)T	This feature was introduced.

Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

The following sections provide information for the Media Inactive Call Detection feature:

- Overview of the Media Inactivity Detection Feature, page 2
- Prerequisites for Media Inactive Call Detection, page 2
- Restrictions for Media Inactive Call Detection, page 4
- Information About Media Inactive Call Detection, page 4
- Configuring CallFeature Parameters for Media Inactive Call Detection, page 7
- Examples, page 15

Overview of the Media Inactivity Detection Feature

The media inactivity detection timer is defined using two CLI commands. One command configures the RTCP report interval, and another defines the multiplying factor **M** (this also identifies the mode of detection with Cisco IOS Release 12.4(4)T. The controlling mechanism is accomplished through the configuration of application CLI.

Media inactive timer = **M** * ip rtcp report interval

The inactivity detection is supported in two modes based on which timer multiplying factor configuration (**M** factor) is used:

- **timer receive-rtcp**—Beginning in Cisco IOS Release 12.3(4)T, this mode detects inactivity using no DSP stats (either an RTP or RTCP packet received is considered active). No explicit enabling is needed. This timer is the default. When this timer is used, the call is disconnected when a silent call is detected. This behavior is not DSP-based, but is the default behavior when no application CLI is configured.
- **timer media-inactive**—This mode is available starting in Cisco IOS Release 12.4(4)T, where detection is based on DSP statistics (it uses RTP-only mechanism; packets sent or received are considered active). If both directions are absent, it is considered inactive. This timer is enabled or disabled using application CLI, which can also be used to control notification.

Prior to Cisco IOS Release 12.4(4)T, the behavior of Media Inactivity Detection enables you to configure a timer using the **timer receive-rtcp** command. This detects media inactivity using packets received on an IP interface and does not provide the granularity to determine if the packet is comfort noise or a real, acceptable voice packet. This mode of inactivity detection is based on either RTP or RTCP packets received from the network.

Beginning with Cisco IOS Release 12.4(4)T, you can use the CLI to configure the timer (**timer media-inactive**) for media inactivity detection using DSP and the criterion is RTP only. In this case, the media are considered inactive only if both send and receive of RTP packets does not happen. If RTP is present in either send or receive direction, it is considered active. In this mode, DSP filters out any comfort noise packets, and the presence of any comfort noise packet is considered inactivity in either direction.

Prerequisites for Media Inactive Call Detection

Note To enable the Media Inactive Call Detection feature, you should first set the inactive timer. This can be done using the **timer receive-rtcp** command in the CLI or by setting the command in the script itself using the **infotag set media_timer_factor** command.

The Media Inactive Call Detection feature requires Cisco IOS Release 12.3(4)T or later. New functionality is added in Cisco IOS Release 12.4(4)T.

For Cisco IOS Release 12.3(4)T, to enable the Media Inactive Call Detection feature, set the information tag **evt_feature_report** using **media_inactivity** type. For example:

infotag set evt_feature_report media inactivity

Beginning in Cisco IOS Release 12.4(4)T, this information tag is no longer supported. The configuration is accomplished using CLI, as discussed in the "New Command-Line Interface in Cisco IOS Release 12.4(4)T" section on page 3.

New Command-Line Interface in Cisco IOS Release 12.4(4)T

New commands in Cisco IOS Release 12.4(4)T provide better control of inactivity detection using application-level CLI. The media inactivity detection timer has to be configured based on which mode of detection is needed:

- Mode 1: using the **timer receive-rtcp**—Silent detection is configured using the **timer receive-rtcp** command together with the **ip rtcp report interval** command. The only action on this mode is to disconnect the call when the call is detected to be inactive. This applies to Cisco IOS Release 12.3(4)T, and is retained in Cisco IOS Release 12.4(4)T. Application CLI can be used to control this detection.
- Mode 2: using **timer media-inactive**—The inactive timer is a combination of the **timer media-inactive** command and the **ip rtcp report interval** command. The use of the **timer media-inactive** command means uses DSP statistics. This capability is based on the configuration of callfeature parameters using application CLI to enable control. This type of configuration is available beginning in Cisco IOS Release 12.4(4)T.

Table 8-1 lists the commands required to configure inactivity duration for the various detection methods.

Table 8-1 Media Inactivity Detection Commands

Detection Method	Commands for Inactivity Duration
RTP only (nonDSP based)	**timer receive-rtcp** **ip rtcp report interval**
RTP only (DSP based)[1]	**timer media-inactive** **ip rtcp report interval**
RTCP only	**timer receive-rtcp** **ip rtcp report interval**
RTP and RTCP (all)	**timer receive-rtcp** **ip rtcp report interval**

1.RTCP commands are used to specify the inactivity duration even if media inactivity detection is based on RTP only.

For more information about the commands and configuration, refer to *Configuring Basic Functionality for TCL IVR and VoiceXML Applications.* You are prompted by the CLI to specify the mode of detection. The mode of configuration allows you to enable or disable the media inactivity detection and to control actions and disconnect cause codes. The configurations are available at three different levels:

1. Dial Peer level
2. Service (application) level
3. Package level

When parameters are defined at different levels, the most localized level of configuration takes precedence. For example, definition at the dial-peer level overrides definition at the package level.

Configuration of the different modes of detection is described in the "Configuring CallFeature Parameters for Media Inactive Call Detection" section on page 7.

Restrictions for Media Inactive Call Detection

- The Media Inactive Call Detection feature is supported on the Cisco AS5350, Cisco AS5350XM, Cisco AS5400, Cisco AS5400XM, and Cisco AS5850 only.
- The Media Inactive Call Detection feature does not change the existing behavior for the default session application and Tcl IVR 1.0 or the existing Tcl IVR 2.0 script behavior that does not request the new feature.
- This feature does not support MGCP call legs.
- The Media Inactive Call Detection feature works in IP only. This feature does not include PSTN inactive call detection.
- This feature supports RTP and RTCP media inactivity detection and notification only on H.323 and SIP basic calls.

Information About Media Inactive Call Detection

This section provides information about the configuration of the Media Inactive Call Detection feature.

- Functionality of Media Inactive Call Detection in Cisco IOS Release 12.4(4)T, page 4
- Modifications to Information Tags and Internal Error Codes, page 5

Functionality of Media Inactive Call Detection in Cisco IOS Release 12.4(4)T

Legacy Functionality

This functionality is an enhancement to the preexisting Media Inactivity Timer feature, which enables gateways to monitor and disconnect VoIP calls if no RTCP packets are received within a configurable time period.

The media inactivity timer feature requires the configuration of the **ip rtcp report interval** command and the **timer receive-rtcp** command to enable detection of RTP and RTCP packets by the gateway. If these commands are configured, the gateway uses RTP and RTCP report detection to determine whether calls on the gateway are still active or should be disconnected.

If no RTCP or RTP packets are received in the resulting time period, the call is disconnected.

The **ip rtcp report interval** command configures the RTCP reporting interval in milliseconds in the range of 1 to 65535. The **timer receive-rtcp** command configures the multiplier in the range of 4 to 1000. These values can be adjusted depending on network traffic conditions. Under normal conditions, a value of 5000 for the **ip rtcp report interval** and a value of 5 for the **timer receive-rtcp** is typical.

Functionality Added in Cisco IOS Release 12.3(4)T

The Media Inactive Call Detection feature offers the following:

- The **show call active** command indicates whether a call has no RTP or RTCP inactivity.
- The **clear call** command offers options so that an inactive call can be released using the called number or calling number. The **clear call** command has been enhanced to configure the Q.850 release cause code to be used when the call is released.

Functionality Added in Cisco IOS Release 12.4(4)T

Features added in this Cisco IOS release include the following:

- The **clear call voice** command is expanded to allow manual release of a call by providing both the called number and the calling number. The system administrator also has the option to clear all calls where media inactivity has been detected.
- The action function configuration for media inactivity and long duration calls includes ignore/syslog/disconnect. If ignore is configured, it allows the application to log whatever it intends to do. If syslog is configured, the call feature package will do some logging while letting the application do whatever extra logging is needed.
- The call feature package can be used for long duration call detection. The Tcl no longer needs to start a leg timer. CLI commands can be used to define the duration, enabling or disabling of the feature, and action function for the feature. CIC information is logged when a long duration call is detected.

Modifications to Information Tags and Internal Error Codes

This feature includes modifications to two information tags and the addition of one information tag:

- **evt_feature_type**
- **evt_feature_param**
- **media_inactivity_err**

For more detailed information, refer to the *Cisco IOS TCL and VoiceXML Application Guide* and the *Cisco IOS TCL Programming Guide*.

evt_feature_type

This existing information tag adds two new event feature types representing media inactivity notification and media activity notification. Note that the script needs to request only for report type media_inactivity. However, after the media inactivity is reported and the VoIP RTP starts receiving RTP or RTCP packets again, the event with type media_activity is automatically notified, indicating that the call is back alive.

Description

To return the feature type string when a feature event is received

Syntax

infotag get evt_feature_type

Mode

Read

Scope

ev_feature

Return Type

String

Direct Mapping

None

Event Names

fax

modem

modem_phase

hookflash

onhook

offhook

media_inactivity

media_activity

evt_feature_param

This is a new information tag added so that the Tcl application can pass parameters related to the feature back to the script. The Media Inactive Call Detection feature uses this new tag to pass on the information of whether an RTCP packet was received before the media inactive condition was met.

Description

To return a parameter related to a specific feature event

Syntax

infotag get evt_feature_param *parameter-name*

Mode

Read

Scope

ev_feature

Return Type

String

Direct Mapping

None

Event Parameter

media_inactivity_type—This parameter belongs to feature **media_inactivity**. The return string is:

- **no media received**—Media inactivity detected (no RTP or RTCP packets have been received for a configured amount of time). An RTCP packet has been received before media inactivity condition is met.
- **no control info received**—Media inactivity detected (no RTP or RTCP packets have been received for a configured amount of time). No RTCP packet has been received before media inactivity condition is met.

Example

infotag get evt_feature_param media_inactivity_type

media_inactivity_err

The internal error code used by the Tcl script for this feature is **media_inactivity_err** (common IEC error #8). This IEC is used to disconnect a call where media inactivity is detected and reported.

Configuring CallFeature Parameters for Media Inactive Call Detection

This feature is enabled by using application CLI to configure callfeature parameters. This section provides information to configure the parameters for media inactive call parameters:

- Configuring Media Inactivity Parameters at the Package Level, page 7
- Configuring Media Inactivity Parameters at the Service or Application Level, page 9
- Configuring Media Inactivity Parameters at the Dial-Peer Level, page 10
- Configure Long Duration Call Detection in Global Configuration Mode, page 12
- Configuring Long Duration Call Detection in Dial-Peer Configuration Mode, page 12
- Verifying the Configuration, page 13

Configuring Media Inactivity Parameters at the Package Level

This feature is introduced in Cisco IOS Release 12.4(4)T. This section describes the use of this feature to set parameters under the callfeature package. After the callfeature parameters are configured for the callfeature package, the parameters are applicable to all applications that use this package.

The parameters configured under a package will be overridden by application level parameters. Therefore, if a particular application wants to change the parameters configured under the package level, you can define the parameters under that particular service context. Whenever that application is used for a call, media inactivity detection is based on parameters under that application. If no parameters are specified by an application, the package level parameters are used by default.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application** *application-name*
4. **package callfeature**
5. **paramspace callfeature** *name*
   and
   paramspace callfeature *name*
   and
   paramspace callfeature *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`application`** *`application-name`* **Example:** `Router(config)# application session`	Enters application configuration mode and specifies the name of the predefined application that you want to enable on the dial peer.
Step 4	**`package callfeature`** **Example:** `Router(config-app)# package callfeature`	Enters application parameter configuration mode.

Step 5	**`paramspace callfeature`** *`name`*
	and
	`paramspace callfeature` *`name`*
	and
	`paramspace callfeature` *`name`*
	
	Example:
	`Router(config-app-param)# paramspace callfeature med-inact-det enable`
	
	`Router(config-app-param)# paramspace callfeature med-inact-disc-cause 44`
	
	`Router(config-app-param)# paramspace callfeature med-inact-action syslog`
		Enters the parameters to enable media inactivity detection at the package level. Enter the following parameters:
		• **med-inact-det enable**—Enables media inactive call detection (default is disabled).
		• **med-inact-disc-cause 44**—Indicates the cause code of media inactive call clearing if the call is disconnected (default is 16).
		• **med-inact-action syslog**—Enters the information for call clearing in the system log. (Options are ignore, syslog, or disconnnect. Default is ignore.)
		Note The following are other keywords available for callfeature parameters, but the three configured here are necessary to effect media inactivity detection at the package level.
		• **long-dur-disc-cause**—Cause code of long duration call clearing if a call is disconnected (default is 16).
		• **long-dur-duration**—Duration of the call to mark it as a long call (range is 1 to 1440 minutes, default is 1440).
		• **long-dur-action**—Action to be taken on a detected long duration call (Options are ignore, syslog, or disconnnect. Default is ignore.).
		• **long-dur-call-mon**—Enable or disable long duration call detection (default is disabled).

Configuring Media Inactivity Parameters at the Service or Application Level

A particular application can be configured with callfeature parameters to override the package level configuration. Therefore, if a particular application wants to change the parameters configured under the package level, you can define the parameters under that particular service context. Whenever that application is used for a call, media inactivity detection is based on parameters under that application. If no parameters are specified by an application, the package level parameters are used by default.

This procedure configures the application for media inactivity detection in Mode 2, meaning that detection uses RTP to send and receive direction status using DSP. If the media inactivity detect parameter under callfeature paramspace is disabled, the default action is to disconnect the call, similar to Mode 1.

After these parameters are configured, any incoming dial peer configured with this application triggers this feature.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application** *application-name*
4. **service** *name*
5. **paramspace callfeature** *name*
   and
   paramspace callfeature *name*
   and
   paramspace callfeature *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`application`** *`application-name`* **Example:** `Router(config)# application session`	Enters application configuration mode and specifies the name of the predefined application that you want to enable on the dial peer.
Step 4	**`service`** *`name`* **Example:** `Router(config-app)# service Default`	Enters application parameter configuration mode and specifies the application for which the callfeatures are to be configured. • The *name* argument identifies the specific service or application.

	Command or Action	Purpose

Step 5	**paramspace callfeature** *name*
	and
	paramspace callfeature *name*
	and
	paramspace callfeature *name*
	
	Example:
	`Router(config-app-param)# paramspace callfeature med-inact-det enable`
	
	`Router(config-app-param)# paramspace callfeature med-inact-disc-cause 44`
	
	`Router(config-app-param)# paramspace callfeature med-inact-action syslog`
		Enters the parameters to enable media inactivity detection at the package level. Enter the following parameters:
		• **med-inact-det enable**—Enables media inactive call detection (default is disabled).
		• **med-inact-disc-cause 44**—Indicates the cause code of media inactive call clearing if the call is disconnected (default is 16).
		• **med-inact-action syslog**—Enters the information for call clearing in the system log. (Options are ignore, syslog, or disconnnect. Default is ignore.)
		Note The following are other keywords available for callfeature parameters, but the three configured here are necessary to effect media inactivity detection at the package level.
		• **long-dur-disc-cause**—Cause code of long duration call clearing if a call is disconnected (default is 16).
		• **long-dur-duration**—Duration of the call to mark it as a long call (range is 1 to 1440 minutes, default is 1440).
		• **long-dur-action**—Action to be taken on a detected long duration call (Options are ignore, syslog, or disconnnect. Default is ignore.).
		• **long-dur-call-mon**—Enable or disable long duration call detection (default is disabled).

Configuring Media Inactivity Parameters at the Dial-Peer Level

Beginning in Cisco IOS Release 12.4(4)T, a new option allows you to configure callfeature parameters under a dial peer while in dial peer configuration mode. The incoming dial peer (either POTS or VoIP) needs to be configured using this method.

After the dial peer is configured with the parameters in the following procedure, all calls going through this dial peer are monitored for silent call detection. The dial-peer level media inactivity parameters take highest precedence over package or service level configurations.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. dial-peer voice *tag* {pots | voatm | vofr | voip}
4. **paramspace callfeature** *name*
   and
   paramspace callfeature *name*
   and
   paramspace callfeature *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`dial-peer voice`** *`tag`* {**`pots`** | **`voatm`** | **`vofr`** | **`voip`**} **Example:** `Router(config)# dial-peer voice 1 voip`	Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial-peer configuration mode. • The *tag* argument defines a particular dial peer. Range is from 1 to 2147483647. • The **pots** keyword indicates that this is a POTS peer that uses VoIP encapsulation on the IP backbone. • The **voip** keyword indicates that this is a VoIP peer that uses voice encapsulation on the POTS network. **Note** The other keywords are not relevant to this feature.

Step 4	**`paramspace callfeature`** *`name`*
	and
	`paramspace callfeature` *`name`*
	and
	`paramspace callfeature` *`name`*
	
	Example:
	`Router(config-dial-peer)# paramspace callfeature med-inact-det enable`
	
	`Router(config-dial-peer)# paramspace callfeature med-inact-action syslog`
	
	`Router(config-dial-peer)# paramspace callfeature med-inact-disc-cause 44`
		Enters the parameters to enable media inactivity detection at the package level. Enter the following parameters:
		• **med-inact-det enable**—Enables media inactive call detection (default is disabled).
		• **med-inact-action syslog**—Enters the information for call clearing in the system log. (Options are ignore, syslog, or disconnnect. Default is ignore.)
		• **med-inact-disc-cause 44**—Indicates the cause code of media inactive call clearing if the call is disconnected (default is 16).
		Note The following are other keywords available for callfeature parameters, but the three configured here are necessary to effect media inactivity detection at the package level.
		
		• **long-dur-disc-cause**—Cause code of long duration call clearing if a call is disconnected (default is 16).
		• **long-dur-duration**—Duration of the call to mark it as a long call (range is 1 to 1440 minutes, default is 1440).
		• **long-dur-action**—Action to be taken on a detected long duration call (Options are ignore, syslog, or disconnnect. Default is ignore.).
		• **long-dur-call-mon**—Enable or disable long duration call detection (default is disabled).

Configure Long Duration Call Detection in Global Configuration Mode

Beginning in Release 12.4(4)T, you can use the CLI to configure detection and monitoring of long duration calls. After the long duration call detection is configured with the following procedure, all calls are monitored for long duration. This section describes the procedure for configuring long duration call detection in global configuration mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **long-duration_call** *timer*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**long-duration_call** *timer* **Example:** `Router(config)# long-duration_call 20`	Activates the long duration call detection feature and sets the timer. • *timer*—This argument sets the length of time (in minutes) after which a call will be defined as long duration. • To disable the long duration call detection feature, use the **no** form of this command.

Configuring Long Duration Call Detection in Dial-Peer Configuration Mode

Beginning in Release 12.4(4)T, you can use the CLI to configure detection and monitoring of long duration calls. After the long duration call detection is configured with the following procedure, all calls are monitored for long duration. This section describes the procedure for configuring long duration call detection in dial-peer configuration mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. dial-peer voice *tag* {pots | voatm | vofr | voip}
4. **long-duration_call** *timer*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**dial-peer voice** *tag* {**pots** | **voatm** | **vofr** | **voip**} **Example:** `Router(config)# dial-peer voice 1 voip`	Defines a particular dial peer, specifies the method of voice encapsulation, and enters dial-peer configuration mode. • The *tag* argument defines a particular dial peer. Range is from 1 to 2147483647. • The **pots** keyword indicates that this is a POTS peer that uses VoIP encapsulation on the IP backbone. • The **voip** keyword indicates that this is a VoIP peer that uses voice encapsulation on the POTS network. **Note** The other keywords are not relevant to this feature.
Step 4	**long-duration_call** *timer* **Example:** `Router(config-dial-peer)# long-duration_call 20`	Activates the long duration call detection feature and sets the timer. • *timer*—This argument sets the length of time (in minutes) after which a call will be defined as long duration.

Verifying the Configuration

After you have completed the procedures to configure the callfeature parameters enabling the media inactive call detection feature, you can use the commands in this section to verify the call monitoring.

SUMMARY STEPS

1. **enable**
2. **show call active voice** [**brief**] [**called-number** *number* | **calling-number** *number* | **compact** | **echo-canceller** | **id** | **media-inactive** {**called-number** *number* | **calling-number** *number*} **long-dur-call**]
3. **clear call voice causecode** *number* {**id** *number* | **media-inactive** | **calling-number** *number* | **called-number** *number*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show call active voice** [**brief**] [**called-number** *number* | **calling-number** *number* | **compact** | **echo-canceller** | **id** | **media-inactive** {**called-number** *number* | **calling-number** *number*} | **long-dur-call**] **Example:** `Router# show call active voice brief long-dur-call`	Displays calls that have no RTP or RTCP activity. • The **brief** keyword shows the brief version of active voice calls. • The **called-number** keyword shows only the call with the specified called number pattern. • The **calling-number** keyword shows only the call with the specified calling number pattern. • The **compact** keyword shows a compact version of the active voice calls. • The **echo-canceller** keyword shows echo canceller data for an active voice call. • The **id** keyword shows only the call with the specified ID. • The **media-inactive** keyword shows only the calls with media inactive being detected and notified. • The **long-dur-call** keyword shows only the calls with long duration being detected and notified. • The *number* argument is a sequence of digits representing a full, recognizable telephone number.
Step 3	**clear call voice causecode** *number* {**id** *number* | **media-inactive** | **calling-number** *number* | **called-number** *number* } **Example:** Router# clear call voice causecode 100 calling-number 4085550100	Clears calls that show media inactive and can clear a specific call. • The **causecode** keyword is a Q.850 disconnect cause code. The cause code *number* can be specified as a number 1 through 127. • The **id** keyword can be used so that only the voice call with the specified ID is disconnected. • The range for the *number* argument is **1-FFFF** . This is a call identifier as shown in brief format. • The **media-inactive** keyword clears only calls with media inactivity detected and notified. • The **calling-number** keyword clears only calls with a specific calling number pattern. • The **called-number** keyword clears only calls with a specific called number pattern.

Examples

This section provides the following output examples:

- show call active voice brief Command: Example, page 15
- show running-config Command: Example, page 17
- Sample Tcl IVR script, page 21

show call active voice brief Command: Example

The existing **show call active voice brief** command has additional media inactive detection data in the IP call leg information:

```
Router# show call active voice brief

<ID>: <start>hs.<index> +<connect> pid:<peer_id> <dir> <addr> <state>
  dur hh:mm:ss tx:<packets>/<bytes> rx:<packets>/<bytes>
 IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms lost:<lost>/<early>/<late>
  delay:<last>/<min>/<max>ms <codec>
 media inactive detected:<y/n> media cntrl rcvd:<y/n> timestamp:<time>
 long_duration_call_detected:<y/n>long duration call duration:n/a timestamp:n/a
           <----------- the above line is new --------------
   MODEMPASS <method> buf:<fills>/<drains> loss <overall%> <multipkt>/<corrected>
   last <buf event time>s dur:<Min>/<Max>s
 FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 ATM <protocol> [int vpi/vci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
  <codec> (payload size)
 Tele <int>: tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l> i/o:<l>/<l> dBm
  MODEMRELAY info:<rcvd>/<sent>/<resent> xid:<rcvd>/<sent> total:<rcvd>/<sent>/<drops>
         speeds(bps): local <rx>/<tx> remote <rx>/<tx>
 Proxy <ip>:<audio udp>,<video udp>,<tcp0>,<tcp1>,<tcp2>,<tcp3> endpt: <type>/<manf>
 bw: <req>/<act> codec: <audio>/<video>
  tx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>
 rx: <audio pkts>/<audio bytes>,<video pkts>/<video bytes>,<t120 pkts>/<t120 bytes>


Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2

11DF : 239062hs.1 +2 pid:1 Answer 4085550100 active
 dur 00:01:04 tx:2383/89775 rx:1187/23318
 Tele 3:D:13: tx:45900/2110/0ms g729r8 noise:-69 acom:45  i/0:-71/-29 dBm

11DF : 239684hs.1 +830 pid:1800877 Originate 18005550100 active
 dur 00:00:49 tx:1066/20965 rx:2378/47215
 IP 10.1.57.5:17750 rtt:5ms pl:38190/0ms lost:0/1/0 delay:50/50/70ms g729r8
 media inactive detected:y media cntrl recv:y timestamp: 12595
 long_duration_call_detected:y long duration call duration:100 timestamp:8397702
       <------------------ the above line is new ------------------
Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

For calls without long_duration_call being detected, the display looks like the following:

```
...
11DF : 239062hs.1 +2 pid:1 Answer 4085550100 active
 dur 00:01:04 tx:2383/89775 rx:1187/23318
```

```
 Tele 3:D:13: tx:45900/2110/0ms g729r8 noise:-69 acom:45  i/0:-71/-29 dBm

11DF : 239684hs.1 +830 pid:1800877 Originate 18005550100 active
 dur 00:00:49 tx:1066/20965 rx:2378/47215
 IP 10.1.57.5:17750 rtt:5ms pl:38190/0ms lost:0/1/0 delay:50/50/70ms g729r8
 media inactive detected:y media cntrl recv:n/a timestamp: n/a
 long_duration_call_detected:n/a
       <------------------ the above line is new ------------------
Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

The long display form of active call records generated by the **show call active voice** command is also modified to add the media inactive detected information:

```
Router_5300# show call active voice

Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2

 GENERIC:
SetupTime=239062 ms
Index=1
PeerAddress=4085254616
PeerSubAddress=
PeerId=1
.
.
.
TELE:
ConnectionId=[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
IncomingConnectionId=[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
TxDuration=51250 ms
VoiceTxDuration=2110 ms
.
.
.
 GENERIC:
SetupTime=239684 ms
Index=1
PeerAddress=18005550100
PeerSubAddress=
.
.
.
VOIP:
ConnectionId[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
IncomingConnectionId[0xB21F398F 0x1DA111D4 0x800B85CB 0x3E43F332]
RemoteIPAddress=1.9.57.5
...
TranslatedRedirectCalledNumber=
TranslatedRedirectCalledOctet=0x7F
MediaInactiveDetected=yes
MediaInactiveTimestamp=12595
MediaControlReceived=yes
LongDurationCallDetected=yes <---- new
LongDurCallTimestamp=8397702 <---- new
LongDurCallDuration=100      <---- new
Username=Telephony call-legs: 1
SIP call-legs: 0
```

```
H323 call-legs: 1
Total call-legs: 2
```

Note For calls where no long duration call is detected and notified, the above three fields are displayed as follows:

```
TranslatedRedirectCalledNumber=
TranslatedRedirectCalledOctet=0x7F
MediaInactiveDetected=no
MediaInactiveTimestamp=
MediaControlReceived=
LongDurationCall=                    <---- new
Username=Telephony call-legs: 1
SIP call-legs: 0
H323 call-legs: 1
Total call-legs: 2
```

show running-config Command: Example

The following is sample output from a show running config command with the media inactive call detection feature enabled.

```
Router# show running-config

Building configuration...
This command has no effect on this line; use modem AT commands instead
This command has no effect on this line; use modem AT commands instead

Current configuration : 13850 bytes
!
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname "jc5400"
!
no boot startup-test
logging buffered 2000000 debugging
no logging console
enable secret 5 $1$afrj$LWwkVSLZ3cKak3OkHsAMt/
enable password lab
!
username 1111
username 2222 password 0 2222
username 123001 password 0 1001
username cisco
!
!
resource-pool disable
clock timezone GMT -8
tdm clock priority 1 6/0
spe default-firmware spe-firmware-1
aaa new-model
!
!
aaa authentication login h323 local group radius
aaa authentication login telnet none
aaa authorization exec h323 local group radius
aaa authorization exec telnet none
```

```
aaa accounting connection h323 start-stop group radius
aaa session-id common
ip subnet-zero
ip cef
ip ftp username dump
ip ftp password dump123
no ip domain lookup
ip host tftp-server1 10.1.1.211
ip host rtsp-server1 10.1.1.211
ip host radius-server1 10.1.1.211
ip host gwip-server1 10.1.1.211
ip host dump-server1 10.1.1.211
!
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
voice call carrier capacity active
!
!
!
voice cause-code
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
ivr prompt memory 16384
!
fax interface-type fax-mail
mta receive maximum-recipients 600
!
!
!
controller T1 6/0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
 no yellow generation
 no yellow detection
!
controller T1 6/1
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/2
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/3
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/4
 shutdown
 framing sf
```

```
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/5
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/6
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
!
controller T1 6/7
 shutdown
 framing sf
 linecode ami
 no yellow generation
 no yellow detection
gw-accounting aaa
!
!
!
interface FastEthernet0/0
 ip address 10.1.1.212 255.255.0.0
 no ip redirects
 no ip mroute-cache
 duplex auto
 speed auto
 no cdp enable
!
interface FastEthernet0/1
 no ip address
 no ip redirects
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
interface Serial0/0
 no ip address
 no ip mroute-cache
shutdown
 clockrate 2000000
 no cdp enable
!
interface Serial6/0
 no ip address
 shutdown
!
interface Serial0/1
 no ip address
 no ip mroute-cache
 shutdown
 clockrate 2000000
 no cdp enable
!
interface Serial6/0:23
```

```
 no ip address
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 isdn bchan-number-order ascending
 no keepalive
 no cdp enable
!
interface Group-Async0
 no ip address
 no ip mroute-cache
 group-range 1/00 1/107
!
ip classless
ip route 10.1.0.0 255.255.0.0 10.1.1.211
ip http server
!
ip pim bidir-enable
ip rtcp report interval 5000
!
!
no logging trap
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
!
radius-server host 10.1.1.211 auth-port 1645 acct-port 1646
radius-server key cisco
radius-server authorization permit missing Service-Type
radius-server vsa send accounting
radius-server vsa send authentication
!
call application voice testapp_JC tftp://10.1.1.211/Scripts/JC_app1.tcl
call rsvp-sync
!
voice-port 6/0:D
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1000 pots
 application testapp_JC
 incoming called-number 5550100
 direct-inward-dial
 port 6/0:D
!
!
dial-peer voice 3000 voip
 destination-pattern 5550100
 session target ipv4:10.1.1.213
 dtmf-relay h245-signal
codec g711ulaw
!
!
gateway
 timer receive-rtcp 5
!
sip-ua
!
!
line con 0
```

```
 exec-timeout 0 0
 logging synchronous
line aux 0
 logging synchronous
line vty 0 4
 password lab
 authorization exec telnet
 login authentication telnet
line 1/00 1/107
 no flush-at-activation
 modem InOut
!
exception core-file jc5400_core
exception protocol ftp
exception dump 10.1.1.211
scheduler allocate 10000 400
end
```

Sample Tcl IVR script

The following is a sample script that is provided for reference purposes only:

```
Router# silence_detect_demo.tcl
#------------------------------------------------------------------
# Copyright (c) 2003 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------
#
# This tcl demo script monitors Media Inactive Call. The Media Inactive Call
# Detection feature detects inactive (silent)H.323 or Sip call-legs on Cisco
# IOS based gateways, and reports this situation to the TCL IVR 2.0 application
# and TCL IVR application checks for the events and logs those events.
#
# This script is designed to place a call to the dnis if DID is configured.
# Otherwise, output dial-tone and collects digits from the caller against
# the dial-plan. If an inactive condition 'ev_feature' is detected then script
# begins to log the inactivity events.  However, if the VOIP RTP starts receiving
# RTP/RTCP packets again, then event with media_activity type will be automatically
# notified, indicating that the call is back alive.
#
#---------------------------------
#   Example Script
#---------------------------------

proc init { } {
    global param
    global timerFactor

    set param(interruptPrompt) true
    set param(abortKey) *
    set param(terminationKey) #
    set timerFactor 6
}

proc act_Setup { } {
    global dest

    if { [infotag get leg_isdid] } {
        set dest [infotag get leg_dnis]

        leg proceeding leg_incoming
        leg setup $dest callInfo leg_incoming
        fsm setstate PLACECALL
```

```
    } else {
        leg setupack leg_incoming
        playtone leg_incoming tn_dial

        set param(dialPlan) true
        leg collectdigits leg_incoming param
    }

}

proc act_GotDest { } {
     global dest

    set status [infotag get evt_status]
    if {  $status == "cd_004" } {
        set dest [infotag get evt_dcdigits]
        leg proceeding leg_incoming
        leg setup $dest callInfo leg_incoming
    } else {
        call close
    }
}

proc act_CallSetupDone { } {

    infotag set evt_feature_report media_inactivity
    infotag set media_timer_factor $timerFactor

    set status [infotag get evt_status]

    if { $status == "ls_000"} {

    } else {
        call close
    }
}

proc act_EvFeatureReceived { } {
   global timerFactor

   set featureType [infotag get evt_feature_type]

   if {  $featureType == "media_inactivity" } {
       log -s "media inactivity or silence is detected"

       set inactivity_type [infotag get evt_feature_param media_inactivity_type]
       if {  $inactivity_type == "no media received" } {
           log -s "media inactivity, RTCP packet was previously received"
       }
       if {  $inactivity_type == "no control info received" } {
           log -s "media inactivity, no RTCP packet was previously received "
       }

   } elseif { $featureType == "media_activity" } {
       log -s "media activity detected and call is back alive, VOIP RTP starts receiving
RTP/RTCP packets"

   } else {
       log -s "other Events have been detected"
   }
}

proc act_Cleanup { } {
```

```
    call close
}

init

#----------------------------------
#   State Machine
#----------------------------------
  set fsm(any_state,ev_disconnected)          "act_Cleanup           same_state"
  set fsm(CALL_INIT,ev_setup_indication)      "act_Setup             GETDEST"
  set fsm(GETDEST,ev_collectdigits_done)      "act_GotDest           PLACECALL"

  set fsm(PLACECALL,ev_setup_done)            "act_CallSetupDone    CALLACTIVE"
  set fsm(CALLACTIVE,ev_feature)              "act_EvFeatureReceived same_state"

  set fsm(CALLACTIVE,ev_disconnected)         "act_Cleanup           CALLDISCONNECT"
  set fsm(CALLDISCONNECT,ev_disconnected)     "act_Cleanup           same_state"
  set fsm(CALLDISCONNECT,ev_disconnect_done)  "act_Cleanup           same_state"

  fsm define fsm CALL_INIT

# End the application
```

Configuring Telephony Call-Redirect Features

Call-redirect features enable call transfer, call forwarding, and call disconnect capabilities on Cisco voice gateways. Voice applications support the Release-to-Pivot (RTPvt) and ISDN Two B-Channel Transfer (TBCT) call-redirect features, and have expanded abilities to receive and send generic transparency descriptor (GTD) information. These features are grouped under the feature name "Voice Application Call Control Enhancements."

For more information about this and related Cisco IOS voice features, see the following:

- Overview of Cisco IOS Tcl IVR and VoiceXML Applications
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for Telephony Call-Redirect Features

Release	Modification
12.3(1)	The Voice Application Call Control Enhancements feature was introduced.
12.3(8)T	The ETSI Call Transfer feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.

Contents

- Prerequisites, page 2
- Restrictions, page 2
- Information About Telephony Call-Redirect Features, page 2
- How to Configure Telephony Call-Redirect Features, page 5
- Configuration Examples, page 16
- Where to Go Next, page 18

- Additional References, page 18

Prerequisites

General

- Cisco voice gateway must have Cisco IOS Release 12.3(1) or later.
- Cisco voice gateway must have the prerequisite configuration that is described in the "Prerequisite" section of the *Cisco IOS TCL IVR and VoiceXML Application Guide*.

RTPvt

- Cisco voice gateway must support the Cisco SC2200 signaling controller. For configuration instructions, see the *Cisco SC2200 Signaling Controller* documentation.

TBCT

- PRI must subscribe to TBCT service from the ISDN switch provider.
- You must write a TCL IVR 2.0 script or VoiceXML 2.0 document that specifies carrier ID properties for TBCT trunks. For information, see the *TCL IVR API Version 2.0 Programmer's Guide* or *Cisco VoiceXML Programmer's Guide*.
- Authentication, Authorization, and Accounting (AAA) must be configured on the Cisco voice gateway. For instructions, see the *Cisco IOS Security Configuration Guide, Release 12.3*.
- ISDN switch must notify the Cisco voice gateway when calls clear to support billing.

GTDs

- GTD generation must be enabled on the Cisco voice gateway. For configuration steps, see the following resources:
  – *GTD for GKTMP Using SS7 Interconnect for Voice Gatekeeper Version 2.0*
  – *R2 and ISUP Transparency for Voice Gateways Version 2.0*

Restrictions

- TBCT supports the National ISDN-2 (NI-2) standard for T1 only. E1 interfaces are not supported.
- An outgoing TBCT call must use the same trunk group as the incoming call.
- GTD parameter transport is supported for H.323 only; it is not supported for SIP.

Information About Telephony Call-Redirect Features

To configure call-control features, you should understand the following concepts:

- Benefits, page 3
- GTD Parameters, page 3
- Release-to-Pivot, page 3
- Two B-Channel Transfer, page 4

- ETSI Call Transfer, page 5

Benefits

- RTPvt enables optimal routing of calls in ISUP networks.
- TBCT optimizes PRI resources by freeing up the gateway's bearer channels.
- User is charged only once for a redirected call.
- Cisco voice applications can append and override GTD parameters for redirected calls.

GTD Parameters

GTD objects are used to represent ISDN User Part (ISUP) messages, parameters, and R2 signals. These objects are encapsulated into existing signaling protocols (for example, H.225), facilitating end-to-end transport. Using GTD as a transport mechanism for signaling data in Cisco IOS software provides a common format for sharing signaling data among various components in a network and for interworking various signaling protocols.

GTD enhancements introduced in this feature include the following:

- VoiceXML applications can append and override GTD parameters on incoming call legs.
- VoiceXML applications can access GTD parameters on outgoing call legs after a transfer is complete using VoiceXML shadow variables.
- TCL IVR applications can append and override GTD parameters.

For detailed information about GTD parameters and how to implement them in a voice application, see the following resources:

- *TCL IVR API Version 2.0 Programmer's Guide*
- *Cisco VoiceXML Programmer's Guide*

Release-to-Pivot

Release-to-Pivot (RTPvt) is a call-redirect method for SS7 networks. It allows a switch to release a call to another switch located earlier in the call path when the originating switch determines that the call should be connected to a new destination number. The preceding switch reoriginates the call directly to the new destination address.

RTPvt provides optimal rerouting capabilities. It frees up trunking and switching resources by rerouting calls that would otherwise be hair-pinned. Hairpinning occurs when an incoming PSTN call cannot be delivered over IP so the call is looped back out to the PSTN. RTPvt is equivalent to the ISDN PRI Two B-Channel Transfer standard.

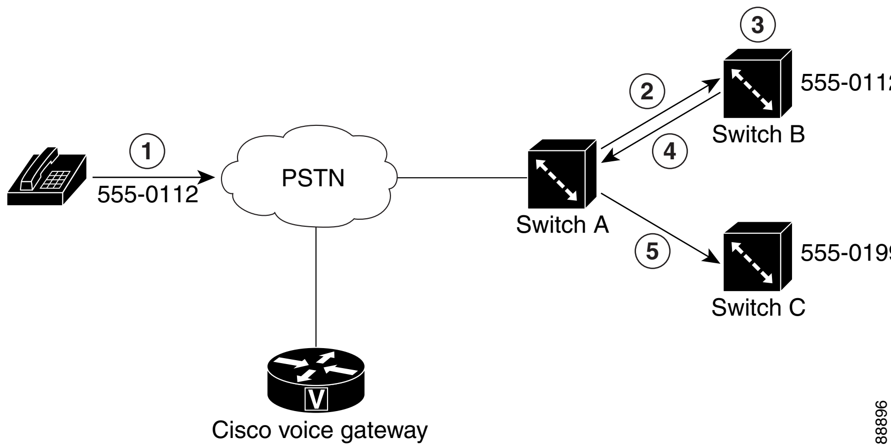
The following describes the call flow for a release to pivot scenario:

1. The PSTN caller dials a destination number on switch B, for example, 555-0112.
2. The originating switch A, which is pivot-capable, sends an ISUP message with the Called Party Number (555-0112) and Redirect Capability (RDC) parameter to switch B.
3. Switch B answers the call.
4. Switch B determines that the call must be redirected to a new destination number (555-0199).

5. Switch A performs the pivot reroute and sends an ISUP message with the new Called Party Number (555-0199) to switch C.

Figure 9-1 illustrates the preceding call-flow scenario:

Figure 9-1 Release-To-Pivot Example



For a TCL application, RTPvt can be invoked at different stages of the call flow, depending on the redirect capability of the originating switch. For a VoiceXML application, RTPvt can be invoked only after the call is answered.

Two B-Channel Transfer

Two B-Channel Transfer (TBCT) is a call-transfer standard for ISDN interfaces. This feature enables a Cisco voice gateway to request an NI-2 switch to directly connect two independent calls. The two calls can be served by the same PRI or by two different PRIs on the gateway. This feature is based on Telcordia GR-2865-CORE.

TBCT makes efficient use of resources by releasing two B channels after a call transfer. Blind transfer of PSTN calls can happen outside the Cisco gateway without tying up gateway resources.

Although the gateway is not involved after calls are transferred, billing continues as though the calls are still connected to the gateway. Customers using this feature need to have special agreements with their ISDN service provider regarding billing, or optionally, the gateway can subscribe to get notification from the switch when a transferred call clears.

To use TBCT, the following conditions must be met:

- PRI interface is subscribed to TBCT service from ISDN service provider.
- Both calls are voice calls.
- Both calls use the same PRI or both PRIs are part of the same trunk group.
- Incoming call is answered.
- Transfer-to number is placed as a separate call.

ETSI Call Transfer

Support is provided for European Telecommunications Standards Institute (ETSI) explicit call transfer functionality on Cisco IOS gateways. The ECT supplementary service enables a user who has two calls, each of which can be either an incoming call or an outgoing call, to connect one of the active calls to another active or alerting call.

Note As of Cisco IOS Release 12.3(8)T, ETSI call transfer is supported only for active calls, not for calls on hold.

ETSI call transfer is enabled using the TBCT for Trunk Groups feature. See Enabling TBCT for Trunk Groups, page 9 for more information.

How to Configure Telephony Call-Redirect Features

See the following sections for configuration tasks for call-control features.

- Configuring Call-Transfer Method for Voice Applications, page 5 (required)
- Enabling RTPvt, page 8 (required)
- Enabling TBCT for Trunk Groups, page 9 (required)
- Configuring Outbound Dial Peer for TBCT Calls, page 10 (required)
- Configuring TBCT Call Limits, page 11 (optional)
- Terminating Billing for Active TBCT Calls, page 12 (optional)
- Enabling ETSI Call Transfer (optional)

Configuring Call-Transfer Method for Voice Applications

You can configure the call-transfer method on the gateway or through properties in the VoiceXML document or Tcl script. For information on configuring the call-transfer method by using VoiceXML or Tcl properties, see the *Cisco VoiceXML Programmer's Guide* or *Tcl IVR API Version 2.0 Programmer's Guide*, respectively.

Note The call-transfer method specified in a VoiceXML document or Tcl script takes precedence over the method specified in the Cisco gateway configuration. Any value that is configured on the gateway is ignored if the same attribute is specified using a VoiceXML or Tcl property.

Configuring the Parameter Routing Method

The **param mode** command is used to configure the default behavior of the callsetup package during the initial attempt to place a call to the specified destination. The callsetup package is used by applications and other packages to place outbound call legs and interwork them with incoming call legs.

To configure the parameter routing method, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. service *service-name*
5. paramspace callsetup mode {**redirect** | **redirect-at-alert** | **redirect-at-connect** | **redirect-rotary** | **rotary**}
6. exit
7. package callsetup
8. **param mode** {**redirect** | **redirect-at-alert** | **redirect-at-connect** | **redirect-rotary** | **rotary**}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
application

Example: Router(config)# application
```

Step 4 Enter service parameter configuration mode:

```
service service-name

Example: Router(config-app)# service fax_detect
```

Step 5 Configure the callsetup parameterspace mode:

```
paramspace callsetup mode {redirect | redirect-at-alert | redirect-at-connect |
redirect-rotary | rotary}

Example: Router(config-app-param)# paramspace callsetup mode redirect-rotary
```

Step 6 Exit the parameterspace configuration mode:

```
exit

Example: Router(config-app-param)# exit
```

Step 7 Enter callsetup package parameter configuration mode:

```
package callsetup

Example: Router(config-app)# package callsetup
```

Step 8 Configure the parameter mode:

```
param mode {redirect | redirect-at-alert | redirect-at-connect | redirect-rotary |
rotary}
```

```
Example: Router(config-app-param)# param mode redirect
```

Configuring the Call Rerouting Method

The **param reroutemode** command is used to configure the default behavior of the callsetup package during the redirect attempt to place a call to the specified destination.This scenario occurs when the initial attempt at placing a call to the original destination is redirected by the destination endpoint to a new endpoint.

To configure the call rerouting method, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. service *service-name*
5. paramspace callsetup reroutemode {**redirect** | **redirect-at-alert** | **redirect-at-connect** | **redirect-rotary** | **rotary**}
6. exit
7. package callsetup
8. param reroutemode {**redirect** | **redirect-at-alert** | **redirect-at-connect** | **redirect-rotary** | **rotary**}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Enter application configuration mode to configure applications and services:

```
application

Example: Router(config)# application
```

Step 4 Enter service parameter configuration mode:

```
service service-name

Example: Router(config-app)# service fax_detect
```

Step 5 Configure the callsetup parameterspace mode:

```
{redirect | redirect-at-alert | redirect-at-connect | redirect-rotary | rotary}

Example: Router(config-app-param)# paramspace callsetup reroutemode redirect-rotary
```

Step 6 Exit parameter configuration mode:

exit

```
Example: Router(config-app-param)# exit
```

Step 7 Enter callsetup package parameter configuration mode:

package callsetup

```
Example: Router(config-app)# package callsetup
```

Step 8 Configure the parameter reroute mode:

param reroutemode {**redirect** | **redirect-at-alert** | **redirect-at-connect** | **redirect-rotary** | **rotary**}

```
Example: Router(config-app-param)# param reroutemode redirect
```

Verifying the Call-Transfer Method for Voice Applications

To verify the call-transfer method for voice applications, use the **show running-config** command.

```
application
.
.
.
!
 package callsetup
  param mode redirect-rotary
  param reroutemode redirect
.
.
.
service fax_detect flash:app_fax_detect.2.1.2.2.tcl
  paramspace callsetup mode redirect-rotary
  param pin-len 4
  paramspace app_debitcard warning-time 60
.
.
.
```

Troubleshooting Tips

If the call-transfer configuration is not successful, verify the following:

- The applications are loaded on the dial peer.
- The parameters and parameterspaces you are configuring are contained in the applications.

Enabling RTPvt

No additional Cisco IOS configuration is required on the gateway to implement RTPvt, other than configuring the call-transfer method as described in the "Configuring Call-Transfer Method for Voice Applications" section on page 5. A TCL IVR script or VoiceXML document triggers RTPvt in response to the GTD Redirect Capability (RDC) parameter.

For detailed information on triggering RTPvt through a TCL IVR script or VoiceXML document, see the following documentation:

- *TCL IVR API Version 2.0 Programmer's Guide*
- *Cisco VoiceXML Programmer's Guide*

Enabling TBCT for Trunk Groups

To enable TBCT on multiple PRIs, all PRIs must be configured as part of the same trunk group. To create the trunk group and enable TBCT, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **isdn switch-type** *switch-type*
4. **trunk group** *name*
5. **carrier-id** *name* [**cic**]
6. **isdn supp-service tbct** [**notify-on-clear**]
7. **exit**
8. **interface serial** *controller:timeslot*
9. **trunk-group** *name* [*preference_num*]
10. **exit**
11. Repeat Steps 8 to 10 for each interface added to the trunk group.

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Specify the central office switch type on the PRI:

```
isdn switch-type switch-type
Example: Router(config)# isdn switch-type primary-ni
```

To support TBCT, the switch type must conform to the NI-2 standard. For ETSI Call Transfer, the switch type should be primary-net5.

Step 4 Enter trunk group configuration mode for the named trunk group:

```
trunk group name
Example: Router(config)# trunk group 1
```

Step 5 Specify the carrier associated with the trunk group:

```
carrier-id name [cic]
Example: Router(config-trunk-group)# carrier-id north1
```

Step 6 Enable TBCT support for the trunk group:

```
isdn supp-service tbct [notify-on-clear]
Example: Router(config-trunk-group)# isdn supp-service tbct notify-on-clear
```

Note For ETSI Call Transfer, notify-on-clear is not supported.

Step 7 Exit trunk group configuration mode and return to global configuration mode:

```
exit
Example: Router(config-trunk-group)# exit
```

Step 8 Enter interface configuration mode for a PRI that you want to assign to the trunk group:

```
interface serial controller:timeslot
Example: Router(config)# interface serial 0:23
```

Note Controller value is platform-dependent.

Step 9 Assign the PRI to the trunk group:

```
trunk-group name [preference_num]
Example: Router(config-if)# trunk-group 1
```

Use the name of the trunk group that was defined by using the **trunk group** command in Step 4.

Step 10 Exit interface configuration mode and return to global configuration mode:

```
exit
Example: Router(config-if)# exit
```

Step 11 Repeat Steps 8 to 10 for each interfaces that you want to add to the trunk group.

Configuring Outbound Dial Peer for TBCT Calls

The gateway must send an outgoing TBCT call over the same trunk as the incoming call. To assign the trunk group to an outbound dial peer and select the outbound dial peer based on the carrier ID, perform the following steps.

Note You must also specify carrier ID properties in the TCL IVR script or VoiceXML document. For information on setting the carrier ID properties for TBCT, see the *TCL IVR API Version 2.0 Programmer's Guide* or *Cisco VoiceXML Programmer's Guide*.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *number* **pots**
4. **trunkgroup** *name*
5. **carrier-id target** *name*
6. **exit**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

```
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal

```
Example: Router# configure terminal
```

Step 3 Enter dial-peer configuration mode for a POTS dial peer:

dial-peer voice *number* **pots**

```
Example: Router(config)# dial-peer voice 100 pots
```

Step 4 Associate the trunk group with this dial peer:

trunkgroup *name*

```
Example: Router(config-dial-peer)# trunkgroup 1
```

Use the name of the trunk group that was defined by using the **trunk group** command in Step 4 of the "Enabling TBCT for Trunk Groups" section on page 9.

Step 5 Specify the carrier ID that is used as the match criteria when selecting an outbound dial-peer:

carrier-id target *name*

```
Example: Router(config-dial-peer)# carrier-id target north1
```

Set this target carrier ID to match the source carrier ID that was specified in Step 5 of the "Enabling TBCT for Trunk Groups" section on page 9.

Step 6 Exit dial-peer configuration mode and return to global configuration mode:

exit

```
Example: Router(config-trunk-group)# exit
```

Configuring TBCT Call Limits

This section includes the commands for setting limits on the number and duration of active TBCT calls that the gateway tracks.

Note Configuring TBCT call limits is relevant only if you have enabled the ISDN switch to notify the gateway when a call clears and you use the **notify-on-clear** keyword with the **isdn supp-service tbct** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **tbct max calls** *number*
4. **tbct max call duration** *minutes*

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router# configure terminal
```

Step 3 Specify the maximum number of active TBCT calls that are allowed by the gateway:

```
tbct max calls number
Example: Router(config)# tbct max calls 50
```

Step 4 Specify the maximum number of minutes before a TBCT call is cleared by the gateway:

```
tbct max call duration minutes
Example: Router(config)# tbct max call duration 10
```

Terminating Billing for Active TBCT Calls

SUMMARY STEPS

1. **enable**
2. **tbct clear call** {**all** | *interface* [*call-tag*]}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Terminate billing for one or more TBCT calls:

```
tbct clear call {all | interface [call-tag]}
Example: Router# tbct clear call T1-6/0
```

Verifying TBCT

SUMMARY STEPS

1. **show running-config**
2. **show call application voice** *application-name*
3. **show call active voice redirect tbct**

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that the application is configured on the gateway and that its transfer method is set to one of the redirect options, for example:

```
Router# show running-config
...
!
application
 service callme tftp://10.10.10.1/scripts/call.vxml
paramspace callsetup mode redirect
!
```

Step 2 Use the **show call application voice** command to verify that the application is loaded onto the gateway and to verify the redirect method. The following example shows output for the application named callme:

```
Router# show call application voice callme

VXML Application callme
    URL=tftp://10.10.10.1/scripts/call.vxml
    Security not trusted
    No languages configured
    It has: 0 calls active.
            0 incoming calls
            0 calls handed off to it
            0 call transfers initiated
            0 pages loaded,  0 successful
            0 prompts played
            0 recorded messages
            The transfer mode is 'redirect'
    Interpreted by Voice Browser Version 2.0 for VoiceXML 1.0 & 2.0.

The VXML Script is:
------------------
<?xml version="1.0"?>
<vxml version="1.0">

   <form id="record_to_ram">
      <record name="myrec"
              beep="true"
              maxtime="10s"
 --More--
Translating "Program"
              dtmfterm="true"
              finalsilence="10ms"
              type="audio/basic;codec=g711ulaw">
          <prompt><audio src="flash:record.au"/></prompt>

        <filled namelist="myrec">
            <prompt><value expr="myrec"/></prompt>
            <clear namelist="myrec"/>
        </filled>
      </record>
   </form>
</vxml>
```

Step 3 Use the **show call active voice redirect tbct** command to verify that an active call is using TBCT, for example:

```
Router# show call active voice redirect tbct

TBCT:
        Maximum no. of TBCT calls allowed:No limit
```

```
        Maximum TBCT call duration:No limit

Total number TBCT calls currently being monitored = 1

ctrl name=T1-2/0, tag=13, call-ids=(7, 8), start_time=*00:12:25.985 UTC Mon Mar 1 1993
```

Tip For a detailed description of the output from these commands, see the *Cisco IOS Voice Command Reference*, Release 12.3 T.

Enabling ETSI Call Transfer

To enable support for ETSI call transfer for multiple PRIs, you must configure the PRIs as part of the same trunk group on the gateway. To create the trunk groups and enable ETSI call transfer, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **isdn switch-type** *switch-type*
4. **trunk group** *name*
5. **isdn supp-service tbct**
6. exit
7. **dial-peer voice** *number* **pots**
8. **destination-pattern** [**+**] *string* [**T**]
9. **trunk-group** *name* [*preference_num*]
10. exit
11. **interface serial** *controller*:*timeslot*
12. **trunk-group** *name* [*preference_num*]
13. **exit**
14. Repeat Steps 11 to 12 for each interface added to the trunk group.

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable
`Example: Router> enable`
Enter your password if prompted.

Step 2 Enter global configuration mode:

configure terminal
`Example: Router# configure terminal`

Step 3 Specify the central office switch type on the PRI. For ETSI Call Transfer, the switch type must be primary-net5.

isdn switch-type *switch-type*

```
Example: Router(config)# isdn switch-type primary-net5
```

Step 4 Enter trunk group configuration mode for the named trunk group:

trunk group *name*

```
Example: Router(config)# trunk group 1
```

Step 5 Enable ETSI support for the trunk group:

isdn supp-service tbct

```
Example: Router(config-trunk-group)# isdn supp-service tbct
```

Note For ETSI Call Transfer, notify-on-clear is not supported.

Step 6 Exit trunk group configuration mode and return to global configuration mode:

exit

```
Example: Router(config-trunk-group)# exit
```

Step 7 Enter dial-peer configuration mode for a POTS dial peer:

dial-peer voice *number* **pots**

```
Example: Router(config)# dial-peer voice 100 pots
```

Step 8 Enter the destination pattern for the dial peer. For example, if you enter the telephone number of a user, and TBCT or ETSI Call Transfer is configured on the trunk group, calls to that user will be transferred using only PRIs in the trunk group that has TBCT or ETSI Call Transfer configured.

destination-pattern [**+**] *string* [**T**]

```
Example: Router(config-dial-peer)# destination-pattern +5557922
```

Step 9 Associate the trunk group with this dial peer:

trunkgroup *name*

```
Example: Router(config-dial-peer)# trunkgroup 1
```

Step 10 Exit dial-peer configuration mode:

exit

```
Example: Router(config)# exit
```

Step 11 Enter interface configuration mode for a PRI that you want to assign to the trunk group:

interface serial *controller:timeslot*

```
Example: Router(config)# interface serial 0:23
```

Note Controller value is platform-dependent.

Step 12 Assign the PRI to the trunk group:

trunk-group *name* [*preference_num*]

```
Example: Router(config-if)# trunk-group 1
```

Use the name of the trunk group that was defined by using the **trunk group** command in Step 4.

Step 13 Exit interface configuration mode and return to global configuration mode:

exit

```
Example: Router(config-if)# exit
```

Step 14 Repeat Steps 11 to 12 for each interfaces that you want to add to the trunk group.

Configuration Examples

This section provides the following configuration examples:

- TBCT Trunk Group Example
- TBCT with Notify on Clear Example
- ETSI Call Transfer Example

TBCT Trunk Group Example

```
!
isdn switch-type primary-ni
!
!
trunk group  1
 carrier-id north1
 isdn supp-service tbct
!
trunk group  2
 carrier-id south2
 isdn supp-service tbct

...

controller T1 1/0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1/1
 framing esf
 clock source line secondary 1
 linecode b8zs
 pri-group timeslots 1-24
!
interface Serial1/0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice modem
 trunk-group 1
 no cdp enable
!
interface Serial1/1:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn incoming-voice modem
 trunk-group 2
 no cdp enable

...

application
 service tbct-app tftp://server1/scripts/TBCTalert.vxml
!
dial-peer voice 100 pots
 service tbct-app
```

```
 incoming called-number 5550112
 direct-inward-dial
 port 1/1:23
!
dial-peer voice 101 pots
 trunkgroup 1
 carrier-id target north1
!
dial-peer voice 102 pots
 trunkgroup 2
 carrier-id target south2
!
```

TBCT with Notify on Clear Example

```
...

interface Serial1/1:23
 no ip address
 isdn supp-service tbct notify-on-clear
 isdn switch-type primary-ni
 isdn incoming-voice modem
!
```

ETSI Call Transfer Example

The following is a sample configuration of the ETSI Call Transfer Feature:

Trunk Group Configuration

```
...
trunk group 1
isdn supp-service tbct
...
dial-peer voip 10 pots
! Configures the POTS dial peer and points it to trunk group 1
...
destination-pattern 000000001
trunkgroup 1
...
int s0:23
trunk-group 1
! Includes interface s0:23 in trunk group 1
...
int s1:23
trunk-group 1
! Includes interface s1:23 in trunk group 1
...
int s2:23
trunk-group 2
! Includes interface s2:23 in trunk group 2
…
```

PRI Configuration

```
...
int s0:23
isdn supp-service tbct
...
dial-peer voice 20 pots
```

```
! Configures the POTS dial peer and points it to trunk group 1
...
destination-pattern 000000002
port 0:D
! Points interface s0:23 to this dial peer
...
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl IVR and VoiceXML Applications" on page 1.
- To configure voice recording using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward" on page 1.
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties" on page 1.
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML" on page 1.
- To configure session interaction for a TCL IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- , page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, TCL IVR and VoiceXML features for that release
- Overview of Cisco IOS Tcl IVR and VoiceXML Applications, page 1—Describes underlying Cisco IOS TCL IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance

Configuring Tcl IVR 2.0 Session Interaction

The Tcl IVR 2.0 Session Interaction feature allows different instances of Tcl IVR applications (sessions) to communicate with other sessions on the same gateway and for applications to dynamically bridge call legs between different sessions. This enables different callers on the same gateway to be notified of each others' presence and to interact. You can also start a session without an active call leg so that a session can act as an application service for other sessions. This feature is useful for implementing a call-monitoring "server" application that is responsible for monitoring incoming calls and dynamically connecting selected callers.

For more information about this and related Cisco IOS voice features, see the following:

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for Tcl IVR 2.0 Session Interaction

Release	Modification
12.3(4)T	This feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.

Contents

- Prerequisites for Session Interaction, page 2
- Restrictions for Session Interaction, page 2
- Information About Session Interaction, page 2
- How to Configure Session Interaction, page 3
- Configuration Examples for Session Interaction, page 9

- Where to Go Next, page 14
- Additional References, page 15

Prerequisites for Session Interaction

- Install Cisco IOS Release 12.3(4)T or later on the voice gateway.
- Configure basic Tcl IVR 2.0 functionality on the gateway. For information, see "Configuring Basic Functionality for Tcl IVR and VoiceXML Applications" on page 1.
- Write a Tcl IVR 2.0 script that implements the desired feature. For information, refer to the *Tcl IVR API Version 2.0 Programmer's Guide*.

Restrictions for Session Interaction

- Application instances can be started for Tcl IVR 2.0 applications only. Session interaction is not supported for VoiceXML applications or for Tcl 1.0 applications.
- A single Tcl IVR 2.0 session can handle a maximum of 12 objects at one time. Objects include a call leg, a digit collection operation, or a conference. If a session is full, it cannot set up a new call, initiate digit collection on a leg, set up a conference, or receive the handoff of a call leg. For example, if a session is handling eight call legs that are in four conferences, the session is full and another leg-setup command fails. Or, if a session is handling six call legs that are all running digit collection, the session is full.

Information About Session Interaction

To use the session interaction feature on Cisco gateways, you should understand the following concepts:

- Tcl IVR 2.0 Session Interaction and Service Registry, page 2
- Benefits of Session Interaction, page 3

Tcl IVR 2.0 Session Interaction and Service Registry

A Tcl IVR 2.0 or VoiceXML application that is configured on a Cisco voice gateway is typically triggered by an incoming call. The application then delivers IVR services to the caller and can create and control one or more call legs. When a voice call invokes an application, it starts an instance, or session, of that application. The application instance executes the code of the application script and can place or transfer a call to a different application. A call can initiate one or more application instances, depending on how your system is configured. A single application instance can manage multiple voice calls.

In Cisco IOS Release 12.3(x)T, you can manually start an instance of a Tcl IVR 2.0 application on the gateway, without a call leg. This enables you to launch an application session on the gateway without requiring an incoming call. For example, you might write an application that monitors the status of a server group, to provide a keep-alive service. An instance of this application interacts with other application sessions by passing status information to other applications that are handling incoming calls. This type of service application runs on the gateway without being triggered by a call.

You can start an instance of a Tcl IVR 2.0 application on the gateway by using the application session configuration submode, or the **call application session start** command in privileged EXEC mode. For configuration information, see the "Starting a New Tcl IVR 2.0 Application Instance (Session)" section on page 3.

Note It is not possible to start an instance of a VoiceXML application because VoiceXML applications using Cisco IOS software require an active call leg to run.

Session interaction enables an application instance to communicate with other sessions on the same gateway and for calls to be bridged between different sessions. Multiple call legs and the ability of sessions to interact with each other fit the Tcl structure because of its strong call-control capabilities. Session interaction is not supported between VoiceXML sessions. For information about Tcl sessions interacting with VoiceXML sessions, refer to the "Hybrid Applications" section in the *Cisco VoiceXML Programmer's Guide*.

Tcl IVR 2.0 application instances can also register as a service. A services registry is essentially a database that keeps track of every application instance that registers as a service. Any other Tcl application can then find and communicate with any registered Tcl application. Registering a session as a service is done from within the Tcl script using a new Tcl verb. For information, refer to the *Tcl IVR API Version 2.0 Programmer's Guide*. You can display a list of application sessions that have registered as services by using the **show call application services registry** command.

Benefits of Session Interaction

- Supports presence-based applications by enabling Tcl IVR 2.0 application sessions to communicate with other sessions and to bridge calls between sessions on the same gateway. This allows different callers on the same gateway to be notified of each others' presence, and for those callers to interact.
- Allows a single application to act as a service for sessions that are handling incoming calls. A service can track which callers are active and which external servers are active, and maintain statistics for the application.

How to Configure Session Interaction

This section contains the following procedures:

- Starting a New Tcl IVR 2.0 Application Instance (Session), page 3 (required)
- Verifying That an Application Instance is Running, page 6 (optional)
- Stopping an Application Instance, page 7 (optional)

Starting a New Tcl IVR 2.0 Application Instance (Session)

There are two ways to start an application instance: from global configuration mode and from privileged EXEC mode. Starting the application session from global configuration mode starts the session and also adds the configuration to the gateway. Starting a session from privileged EXEC mode starts the session until the gateway reboots, or the session is stopped by the **call application session stop** command. The application session restarts at system reboot only if the **start** command is used in application session configuration submode.

Choose one of the following tasks, depending on whether you want to start the session as part of the gateway configuration or you want to start the instance temporarily, for instance during testing.

- Starting an Application Instance in the Configuration, page 4
- Starting an Application Instance in Privileged EXEC Mode, page 5

Starting an Application Instance in the Configuration

This section describes how to start a new instance of a Tcl IVR 2.0 application in the gateway configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** *application-name location*
5. **session**
6. **start** *instance-name application-name*
7. **end**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
    enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router(config)# configure terminal
```

Step 3 Load a Tcl script and specify its application name:

```
    application
      service application-name location

Example:
Router(config)# application
Router(config-app)# service my_app flash:demo1.vxml
```

Step 4 Start an instance of a Tcl application:

```
    application
      session
        start instance-name application-name
```

Use the name that was assigned to the application in Step 3.

```
Router(config)# application
Router(config-app)# session
Router(config-app-session)# start my_instance my_app
```

Step 5 Exit to privileged EXEC mode:

```
    end
```

```
Example: Router(config)# end
```

Starting an Application Instance in Privileged EXEC Mode

This section describes how to start an application instance in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** *application-name location*
5. **exit**
6. **call application session start** *instance-name* [*application-name*]

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router(config)# configure terminal
```

Step 3 Load a Tcl script and specify its application name:

```
application
service application-name location
Router(config)# application
Router(config-app)# service my_app flash:demo1.vxml
```

Step 4 Exit global configuration mode and return to privileged EXEC mode:

```
exit
Example: Router(config)# exit
```

Step 5 Start an instance of a Tcl application:

```
call application session start instance-name [application-name]
```

You do not have to enter the application name if the application instance was previously started and stopped.

```
Example: Router# call application session start my_instance my_app
```

Verifying That an Application Instance is Running

SUMMARY STEPS

1. **show running-config**
2. **show call application sessions** [**callid call-id** | **id session-id** | **name instance-name**]

3. Follow the steps in the "Verifying Loading of Service" section on page 26.

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that the application is configured on the gateway with the application configuration submodes, for example:

```
!
application
 service my_app tftp://10.10.1.1/sample.tcl
!
 session
  start my_instance my_app
!
!
```

Step 2 Use the **show call application sessions** command to verify that the application is running on the gateway. The following example shows output for the application named *serv1*:

```
Router# show call application sessions name serv1

Session named serv1 is in the start list in state running
  It is configured to start on GW reboot
  The application it runs is sample_service
  Handle is TCL_HAND*1653710732*0*3193204
TCL Session ID B
                 App: sample_service
                 URL: tftp://dev/demo/scripts/sample_service.tcl
        Session name: serv1
      Session handle: TCL_HAND*1653710732*0*3193204
           FSM State: start_state
    ID for 'show call active voice id' display: 0
                Legs:
            Services: data_service
```

Tip If the output shows that the application is in the "stopped" state, the script might have a syntax error that prevents it from running. The **show call application sessions** command does not display a stopped session if the script was started by using the **call application session start** command in privileged EXEC mode instead of global configuration mode.

Step 3 Follow the steps in the "Verifying Loading of Service" section on page 26 to verify that the voice application is loaded and running on the gateway.

Troubleshooting Tips

If the application instance does not successfully start on the gateway, see Table 10-1 for some possible causes and actions.

Table 10-1 Application Instance Does Not Start Running on Gateway

Possible Causes	Suggested Actions
Tcl script contains an error that prevents the script from running.	• Enable the **debug voip application error** and **debug voip application script** commands and then retry the **call application session start** command. These debug commands provide information about why the script cannot run. For example, an error message is displayed if the script contains bad syntax. • An error in the script can also prevent it from loading onto the gateway. With the above debug commands enabled, try reloading the script by using the **call application voice load** command. To verify the contents of the application script or document, use the **show call application voice** command.
Cisco gateway cannot access the external server to download the Tcl script.	Ping the corresponding server to make sure that the gateway has connectivity.
An application instance by the same name is already running.	Use the **show call application session** command to verify whether another instance with the same name is running. If you try to start a new session using the same name as a session that is currently running, the gateway displays the error message, "Cannot restart session instance *name*, it is running." You can start another instance for the same application by using a different name.

Stopping an Application Instance

There are two ways to stop an applications instance: from global configuration mode and from privileged EXEC mode. Stopping the application session from global configuration mode stops the session and removes the **call application session start** command from the gateway's configuration. Stopping a session from privileged EXEC mode stops the session until the gateway reboots, if the session is configured to start by the **call application session start** command in global configuration mode.

Choose one of the following tasks, depending on whether you want to stop this instance temporarily, for instance during testing, or you want to remove the session configuration from the gateway.

- Stopping an Application Instance in the Configuration, page 7
- Stopping an Application Instance in Privileged EXEC Mode, page 8

Stopping an Application Instance in the Configuration

This section describes how to stop an application instance and remove it from the gateway configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **session**
5. **no start** *instance-name*
6. **exit**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

```
enable
Example: Router> enable
```

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
configure terminal
Example: Router(config)# configure terminal
```

Step 3 Stop an instance of a Tcl IVR application:

```
application
  session
    no start instance-name
```

Note This command stops the instance of the application and removes the configuration from the gateway. VoiceXML sessions cannot be stopped with the **no session start** command because VoiceXML sessions cannot be started with Cisco IOS commands.

Tip Use the **show call application sessions** command to see a list of currently running instances.

Step 4 Exit global configuration mode and return to privileged EXEC mode:

```
exit
Example: Router(config)# exit
```

Stopping an Application Instance in Privileged EXEC Mode

This section describes how to stop an application instance in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **call application session stop** {**callid** *call-id* | **handle** *handle* | **id** *session-id* | **name** *instance-name*}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

enable

Example: Router> enable

Enter your password if prompted.

Step 2 Stop an instance of a VoiceXML or Tcl IVR application:

call application session stop {**callid** *call-id* | **handle** *handle* | **id** *session-id* | **name** *instance-name*}

Configuration Examples for Session Interaction

This section provides the following configuration example:

- Tcl IVR Application Sessions Example, page 9

Tcl IVR Application Sessions Example

In the following example, the Tcl IVR 2.0 script sample_service.tcl is configured to run as a session on the gateway without a call leg. The **call application voice** command loads the script onto the gateway and assigns it the application name sample_service. The **call application session start** command starts a legless session of the application on the gateway. The script registers itself as a service by using the Tcl service register command. As a service, the script simply keeps track of how many calls come into the gateway. When a call comes in, the script service_user.tcl, which is configured in the dial peer to handle the call, finds and exchanges a message with the service. The caller hears nothing. You can use the **debug voip ivr scripts** command to display output showing the process.

Note To see the contents of the Tcl scripts used in this example, refer to the *Tcl IVR API Version 2.0 Programmer's Guide*.

Gateway Configuration

```
!
version 12.2
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
no service password-encryption
service internal
!
hostname sblab160
!
logging buffered 2000000 debugging
aaa new-model
!
!
aaa authentication login h323 local
aaa session-id common
enable password lab
!
username admin nopassword
username cisco nopassword
```

```
username 100 password 101
username 123 password 321
!
!
resource-pool disable
clock timezone pst -8
clock summer-time pdt recurring
!
ip subnet-zero
ip domain-name cisco.com
ip host rtsp-ws.cisco.com 1.7.153.4
ip host dirt 223.255.254.254
ip host px1-sun.cisco.com 1.7.100.1
ip host ts 1.7.100.1
ip host CALLGEN-SECURITY-V2 48.92.30.60 79.63.0.0
ip name-server 172.29.248.16
ip name-server 171.69.187.13
ip name-server 1.7.100.1
!
!
isdn switch-type primary-net5
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
ivr prompt memory 16000
ivr prompt streamed http
http client cache refresh 2
fax interface-type modem
mta receive maximum-recipients 0
!
controller T1 0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 2
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 3
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
!
!
interface Ethernet0
 ip address 1.7.102.39 255.255.0.0
 ip helper-address 223.255.254.254
 no ip redirects
 no ip route-cache
 no ip mroute-cache
```

```
 load-interval 30
!
interface Serial0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 no cdp enable
!
interface Serial1:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn protocol-emulate network
 isdn incoming-voice modem
 no isdn T309-enable
 isdn disconnect-cause 1
 no cdp enable
!
interface Serial2:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn protocol-emulate network
 isdn incoming-voice modem
 no isdn T309-enable
 isdn disconnect-cause 1
 no cdp enable
!
interface Serial3:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn protocol-emulate network
 isdn incoming-voice modem
 no isdn T309-enable
 isdn disconnect-cause 1
 no cdp enable
!
interface FastEthernet0
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
!
ip default-gateway 1.7.0.1
ip classless
ip route 223.255.254.0 255.255.255.0 1.7.0.1
no ip http server
ip pim bidir-enable
!
!
access-list 2 deny   1.1.1.1
access-list 2 permit any
!
!
call rsvp-sync
!
application
 service service_user tftp://dev/demo/scripts/service_user.tcl
!
 service sample_service tftp://dev/demo/scripts/sample_service.tcl
```

```
!
!
 session
  start serv1 sample_service
!
voice-port 0:D
!
voice-port 1:D
!
voice-port 2:D
!
voice-port 3:D
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1 pots
 service service_user
 incoming called-number 52944
!
dial-peer voice 2 pots
service rate
service sessvars out-bound
 destination-pattern 12.
 port 2:D
!
dial-peer voice 190 voip
service k00
 destination-pattern 190
 session target ipv4:1.7.102.38
 incoming called-number 123
 dtmf-relay cisco-rtp
 codec g711ulaw
 no vad
!
dial-peer voice 3 pots
service rate
 shutdown
 destination-pattern 12.
 port 1:D
!
dial-peer voice 4 pots
service rate
 shutdown
 destination-pattern 12.
 port 3:D
!
dial-peer voice 7 voip
 destination-pattern .......
 session target ipv4:1.7.104.85
!
!
line con 0
 exec-timeout 0 0
 privilege level 15
 transport preferred none
line aux 0
 exec-timeout 0 0
 privilege level 15
line vty 0
```

```
 exec-timeout 0 0
 password 7 09404F0B
 notify
line vty 1 4
 exec-timeout 0 0
 privilege level 15
!
no scheduler max-task-time
end
```

Cisco IOS Command Output

Session Start Output

The following examples show output when the session is started:

```
Router# show call application sessions

TCL Sessions
     SID  Name        Called     Calling                App Name              Legs
       B  serv1                                         sample_service

VXML Sessions
     SID  Called                 Calling                App Name              Legs
    No running VXML sessions

Router# show call application session name serv1

Session named serv1 is in the start list in state running
  It is configured to start on GW reboot
  The application it runs is sample_service
  Handle is TCL_HAND*1653710732*0*3193204
TCL Session ID B
                App: sample_service
                URL: tftp://dev/demo/scripts/sample_service.tcl
       Session name: serv1
     Session handle: TCL_HAND*1653710732*0*3193204
          FSM State: start_state
    ID for 'show call active voice id' display: 0
               Legs:
           Services: data_service
```

Incoming Call Output

The following example shows output from the **show call application sessions** command listing the application service_user as active when a call comes into the gateway:

```
Router# show call application sessions

TCL Sessions
     SID  Name        Called     Calling                App Name              Legs
       B  serv1                                         sample_service
       C                  52944  8009673822             service_user         8

VXML Sessions
     SID  Called                 Calling                App Name              Legs
    No running VXML sessions
```

Tcl Puts Output

The following example shows output from the **debug voip application script** command, displaying the Tcl puts commands imbedded in the script, when a call comes into the gateway:

```
Router# debug voip application script
```

```
ivr script debugging is on

*May 28 13:17:04.779: //-1//TCL2:HN0030B974:/tcl_PutsCmd: Timer went off. Get data, and
reset the timer.

*May 28 13:17:04.779:
*May 28 13:17:08.139: //8//TCL2:/tcl_PutsCmd: sample_service_user.tcl is handling callid
8
*May 28 13:17:08.139:
*May 28 13:17:08.139: //-1//TCL2:HN0030B974:/tcl_PutsCmd: sample_service.tcl got a
message. Respond with the data.
*May 28 13:17:08.139:
*May 28 13:17:08.139: //-1//TCL2:HN0030B974:/tcl_PutsCmd: Send of data to
TCL_HAND*1672011776*0*3526560 returned success
*May 28 13:17:08.139:
*May 28 13:17:08.139: //-1//TCL2:HN0035CFA0:/tcl_PutsCmd: sample_service_user.tcl for
callid  8 got a response:
*May 28 13:17:08.143:
*May 28 13:17:08.143: //-1//TCL2:HN0035CFA0:/tcl_PutsCmd:      data(call_count)=1
*May 28 13:17:08.143:
*May 28 13:17:08.143: //-1//TCL2:HN0035CFA0:/tcl_PutsCmd:      data(run_time)=330
*May 28 13:17:08.143:

*May 28 13:17:34.779: //-1//TCL2:HN0030B974:/tcl_PutsCmd: Timer went off. Get data, and
reset the timer.

*May 28 13:17:34.779:
*May 28 13:17:36.227: //8//TCL2:/tcl_PutsCmd:
      Script for callids < 8> got event ev_disconnected
              Closing up now
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl IVR and VoiceXML Applications" on page 1.
- To configure voice recording using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward" on page 1.
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties" on page 1.
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML" on page 1.
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features" on page 1.
- To configure support for SIP and TEL URLs, see "Configuring SIP and TEL URL Support" on page 245.
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications" on page 1.

Additional References

- , page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release
- Overview of Cisco IOS Tcl IVR and VoiceXML Applications, page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance

Configuring SIP and TEL URL Support

The Configuring SIP and TEL URL Support feature enables Cisco gateways to direct incoming calls to a voice application based on the Uniform Resource Locator (URL) and Tcl Interactive Voice Response (IVR) 2.0 and VoiceXML applications to place outbound calls to a SIP or TEL URL. This feature expands call-control capabilities by allowing voice applications to use URL destinations and by implementing dialing plans using SIP or TEL URLs, rather than telephone numbers.

Note In this document, the terms uniform resource identifier (URI) and URL are used interchangeably.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Configuring SIP and TEL URL Support" section on page 31.

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

Contents

- Prerequisites for Configuring SIP and TEL URL Support, page 2
- Restrictions for Configuring SIP and TEL URL Support, page 2
- Information About SIP and TEL URL Support, page 2
- How to Configure SIP and TEL URL Support, page 3
- Configuration Examples for SIP and TEL URL Support, page 21
- Where to Go Next, page 30
- Additional References, page 31
- Feature Information for Configuring SIP and TEL URL Support, page 31

Prerequisites for Configuring SIP and TEL URL Support

- Install Cisco IOS Release 12.3(4)T or a later release on the voice gateway.
- Configure basic voice application functionality on the gateway. For information, see "Configuring Basic Functionality for Tcl and VoiceXML Applications".
- Write a Tcl IVR 2.0 script or VoiceXML document that implements Session Iniation Protocol (SIP) or telephone (TEL) URL support. For information, see the *Tcl IVR API Version 2.0 Programmer's Guide* or *Cisco VoiceXML Programmer's Guide.*

Restrictions for Configuring SIP and TEL URL Support

- Access is provided only to headers in the SIP Invite, Subscribe, and Notify messages and H.323 Setup message.
- SIP and TEL URLs are not supported destinations for H.450 or SIP call transfers or call forwarding using Tcl IVR or VoiceXML applications.
- A destination URL can have a maximum length of 1024 bytes.
- H.323 rejects a call if the URL is more than 512 characters because H.225 limits the URL length to 512 bytes.

Information About SIP and TEL URL Support

To configure voice application enhancements on a Cisco gateway, you should understand the following concepts:

- SIP and TEL URL Support for Voice Applications, page 2
- Benefits of SIP and TEL URL Support, page 3

SIP and TEL URL Support for Voice Applications

Cisco IOS Release 12.3(4)T allows Tcl IVR 2.0 and VoiceXML applications to place calls to SIP and TEL URL destinations in addition to E.164 telephone numbers. For outgoing calls, the voice application provides the destination as either an URL or an E.164 number. Incoming calls are matched to a dial peer based on the URL, triggering the voice application that is configured in the dial peer. New dial-peer matching rules are implemented for URLs. The standard dial-peer matching rules apply to telephone number destinations. VoiceXML applications can use the transfer tag to place calls to URL destinations. Tcl IVR 2.0 scripts can do a leg setup to a URL instead of a telephone number.

For configuration information, use the following references:

- To create a voice class that defines the criteria for matching a URL, see the "Configuring a Voice Class URI" section on page 3.
- To assign the URL voice class to an inbound or outbound dial peer, see the "Configuring an Inbound Dial Peer to Match on a URI" section on page 7 or the "Configuring an Outbound Dial Peer for URI Destinations" section on page 15.
- For a description of the new dial-peer matching rules for URLs, see to the **destination uri** command and **incoming uri** command in the *Cisco IOS Voice Command Reference*.

- To enable voice applications to read and pass SIP headers, see the *Cisco IOS SIP Configuration Guide*.
- To specify SIP and TEL URLs in a Tcl IVR 2.0 script or VoiceXML document, see the *Tcl IVR API Version 2.0 Programmer's Guide* or *Cisco VoiceXML Programmer's Guide*.

Benefits of SIP and TEL URL Support

- Enhances call-control features by supporting SIP and TEL URL destinations for a VoiceXML transfer or Tcl leg setup. Incoming calls can be directed to voice applications based on a SIP or TEL URL, and outbound calls can be placed using a URL to route the call.
- Complies with the W3C VoiceXML 2.0 Working Draft requirement for TEL URL support, enabling additional information such as the account number to be provided when a call is transferred to or from a VoiceXML application.
- Allows dialing plans to be implemented using SIP URLs instead of telephone numbers, simplifying the management of SIP telephony networks.

How to Configure SIP and TEL URL Support

This section contains the following procedures:

- Configuring a Voice Class URI, page 3 (required)
- Configuring an Inbound Dial Peer to Match on a URI, page 7 (required)
- Verifying Dial-Peer Configuration for an Incoming URI Based Dial-Peer Match, page 12 (optional)
- Configuring an Outbound Dial Peer for URI Destinations, page 15 (required)
- Verifying Dial-Peer Configuration for an Outgoing URI, page 16 (optional)

Configuring a Voice Class URI

To configure a URI voice class and assign it to an inbound or outbound dial peer, choose one of the following tasks, depending on whether a call has a SIP or TEL URI:

- Configuring a Voice Class SIP URI, page 3
- Configuring a Voice Class TEL URL, page 6

You define sets of related dial-peer attributes in voice classes. The Configuring SIP and TEL URL Support feature introduces a new type of voice class for URI attributes. A URI voice class defines the criteria for matching the URI in a call. You can configure a voice class to match either the entire URI or specific fields within the URI, but, not both. You then assign the voice class to an inbound or outbound dial peer. Calls are matched to the dial peer based on the URI.

Configuring a Voice Class SIP URI

This section describes how to configure a voice class to match on a SIP URI.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **voice class uri** *tag* **sip**
4. **pattern** *uri-pattern*
5. **host** *hostname-pattern* or **host** {**ipv4:***ipv4-address* | **ipv6:***ipv6-address* | **dns:***domain-name* | *hostname-pattern*}
6. **user-id** *username-pattern*
7. **phone context** *context-pattern*
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`voice class uri`** *`tag`* **`sip`** **Example:** `Router(config)# voice class uri ab200 sip`	Enter voice class configuration mode for a SIP URI.
Step 4	**`pattern`** *`uri-pattern`* **Example:** `Router(config-voice-uri-class)# pattern ^408`	(Optional) Specifies a regular expression pattern for matching the entire TEL URI. **Note** When you configure the **pattern** command, you cannot configure the **phone number** or **phone context** command, because the **pattern** command matches the entire URI.

Step 5	**`host`** *`hostname-pattern`*
	or
	`host` {**`ipv4:`***`ipv4-address`* | **`ipv6:`***`ipv6-address`* | **`dns:`***`domain-name`*}
	
	Example:
	`Router(config-voice-uri-class)# host server1`
	or
	`Router(config-voice-uri-class)# host ipv4:10.0.0.0`
		(Optional) Specifies a regular expression pattern for matching the hostname field in the SIP URI.
		• Only one instance of the command can be configured at any given time.
		Specifies the **host** command by assigning an IPv4 address, IPv6 address, or Domain Name Server (DNS) name. You can specify up to ten instances of this command.
		Note You can use either the **host** *hostname* or **host** {**ipv4:***ipv4-address* | **ipv6:***ipv6-address* | **dns:***domain-name*} command.

Step 6	**`user-id`** *`username-pattern`* **Example:** `Router(config-voice-uri-class)# user-id elmo`	(Optional) Specifies a regular expression pattern for matching the username field in the SIP URI.
Step 7	**`phone context`** *`context-pattern`* **Example:** `Router(config-voice-uri-class)# phone context 408`	(Optional) Specifies that only calls with a matching phone-context field in the URI are selected. **Note** You must use the **phone context** command with either the **host** or **user-id** command or both. Using this command alone does not result in any matches on the voice class.
Step 8	**`end`** **Example:** `Router(config-voice-uri-class)# end`	Exits voice class URI configuration mode and enters privileged EXEC mode.

What to Do Next

- To use the URI voice class to handle incoming calls, continue with the "Configuring an Inbound Dial Peer to Match on a URI" section on page 7.
- To use the URI voice class to handle outbound calls, continue with the "Configuring an Outbound Dial Peer for URI Destinations" section on page 15.

Configuring a Voice Class TEL URL

This task describes how to configure a voice class to match a TEL URI.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class uri** *tag* **tel**
4. **pattern** *uri-pattern*
5. **phone number** *phone-number-pattern*
6. **phone context** *context-pattern*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice class uri** *tag* **tel** **Example:** `Router(config)# voice class uri ab200 tel`	Enters voice class URI configuration mode for a TEL URI.
Step 4	**pattern** *uri-pattern* **Example:** `Router(config-voice-uri-class)# pattern ^408`	(Optional) Specifies a regular expression pattern for matching the entire TEL URI. **Note** If you use the **pattern** command you cannot use the **phone number** or **phone context** command in the following steps because the **pattern** command matches the entire URI.
Step 5	**phone number** *phone-number-pattern* **Example:** `Router(config-voice-uri-class)# phone number ^8555`	(Optional) Specifies a regular expression pattern for matching the phone number field in the TEL URI.
Step 6	**phone context** *context-pattern* **Example:** `Router(config-voice-uri-class)# phone context 555`	(Optional) Specifies that only calls with a matching phone-context field are selected. **Note** You must use the **phone context** command with the **phone number** command. Using this command alone does not result in any matches on the voice class.
Step 7	end **Example:** `Router(config-voice-uri-class)# end`	Exits voice class configuration mode and enters privileged EXEC mode.

Configuring an Inbound Dial Peer to Match on a URI

To configure an inbound dial peer to match an URI, choose one of the following tasks, depending on whether the dial peer uses SIP or H.323.

- Configuring an Inbound Dial Peer to Match the URI in H.323 Calls, page 8
- Configuring an Inbound Dial-Peer to Match the URI on SIP Calls, page 9

Prerequisites

- Enable SIP header passing. For information, see the *Cisco IOS SIP Configuration Guide*, Release 15.1
- Write a Tcl IVR 2.0 script or VoiceXML document that accepts a SIP or TEL URI. For information, see the *Tcl IVR API Version 2.0 Programmer's Guide* or *Cisco VoiceXML Programmer's Guide*.

Configuring an Inbound Dial Peer to Match the URI in H.323 Calls

This task describes how to configure an inbound dial peer to match the URI in a H.323 call.

SUMMARY STEPS

1. enable
2. **configure terminal**
3. **dial-peer voice** *tag* **voip**
4. **service** *name*
5. **incoming uri** {**called** | **calling**} *tag*
6. **end**

DETAILED STEPS.

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`dial-peer voice`** *`tag`* **`voip`** **Example:** `Router(config)# dial-peer voice 2 voip`	Enters dial-peer configuration mode for a VoIP dial peer.
Step 4	**`service`** *`name`* **Example:** `Router(config-dial-peer)# service vapp1`	Associates an application with this VoIP dial peer. **Note** When you configure the **pattern** command, you cannot configure the **phone number** or **phone context** command, because the **pattern** command matches the entire URI.
Step 5	**`incoming uri`** `{`**`called`** `|` **`calling`**`}` *`tag`* **Example:** `Router(config-dial-peer)# incoming uri called ab400`	Specifies the URI voice class that matches calls to this dial peer based on the destination URI or source URI in the H.225 message. **Note** This URI voice class must already be configured by using the **voice class uri** command. **Note** Incoming calls with a URI that matches the configured voice class are linked to the dial peer and to the VoiceXML or Tcl IVR application that you configured in Step 4.

Step 6	`end`
	
	Example:
	`Router(config-dial-peer)# end`
		Exits the dial peer configuration mode and enters privileged EXEC mode.

Configuring an Inbound Dial-Peer to Match the URI on SIP Calls

This task describes how to configure an inbound dial-peer to match the URI in a SIP call.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **voice class uri** *tag* **sip**
4. **host** *hostname-pattern* or **host** {**ipv4:***ipv4-address* | **ipv6:***ipv6-address* | **dns:***dns-address* | *hostname-pattern*}
5. **exit**

6. **dial-peer voice** *tag* **voip**
7. **session protocol sipv2**
8. **incoming uri** {**from** | **request** | **to** | **via**} *tag*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**voice class uri** *tag* **sip'** **Example:** `Router(config)# voice class uri ab200 sip`	Creates a voice class for matching dial peers to a SIP and enters voice URI class configuration mode.

Step 4	**host** *hostname-pattern*
	or
	host {**ipv4:***ipv4-address* | **ipv6:***ipv6-address* | **dns:***domain-name*}
	
	Example:
	`Router(config-voice-uri-class)# host server1`
	or
	`Router(config-voice-uri-class)# host ipv4:10.0.0.0`
		(Optional) Specifies a regular expression pattern for matching the hostname field in the SIP URI.
		• This command can have a single instance only.
		Specifies the host command by assigning the IPv4 address, IPv6 address, or DNS name.
		• You can specify up to ten instances of this command.
		Note You can use either the **host** *hostname* or **host** {**ipv4:***ipv4-address* | **ipv6:***ipv6-address* | **dns:***domain-name*} command.

Step 5	**exit** **Example:** `Router(config-voice-uri-class)# exit`	Enters global configuration mode.
Step 6	**dial-peer voice** *tag* **voip** **Example:** `Router(config)# dial-peer voice 6000 voip`	Enters dial peer voice configuration mode.
Step 7	**session protocol sipv2** **Example:** `Router(config-dial-peer)# session protocol sipv2`	Configures SIP as the session protocol type.
Step 8	**incoming uri** {**from** | **request** | **to** | **via**} *tag* **Example:** `Router(config-dial-peer)# incoming uri via ab200`	Specifies the voice class used to match a VoIP dial peer to the URI of an incoming call.
Step 9	**end** **Example:** `Router(config-dial-peer)# end`	Exits dial peer voice configuration mode and enters privileged EXEC mode.

Verifying Dial-Peer Configuration for an Incoming URI Based Dial-Peer Match

Perform this task to display information about the dial-peer configuration for an incoming URI based dial-peer match. These **show** commands need not be entered in any specific order.

SUMMARY STEPS

1. **show voice class uri** [*tag* | **summary**]
2. **show dial-peer voice**
3. **show dialplan incall uri sip** {*from* | *request* | *to* | *via*} **uri**

DETAILED STEPS

Step 1 **show voice class uri** [*tag* | **summary**]

Use the **show voice class uri summary** command to display summary information about the configured URI voice classes. The following sample output shows the configured host instances under the URI voice class:

```
Router# show voice class uri 1
Voice URI class: 1
SNMP status = Active
Schema = sip
user-id =
host =
phone context =
Host instances:
 ipv4:10.0.0.0
 ipv6:[2001:0DB8:0:1:FFFF:1234::5]
 dns:ogw.example.com
 exam.example.com
```

Step 2 **show dial-peer voice**

Use the **show dial-peer voice** command to display information for voice dial peers. The following sample output shows the voice class URI 2 being set into dial peer 2:

```
Router# show dial-peer voice 2
VoiceOverIpPeer2
        description = `',
        tag = 2, destination-pattern = `',
        URI classes:
        Incoming (Request) =
        Incoming (Via) = 1
        Incoming (To) =
        Incoming (From) =
        Destination
```

Step 3 **show dialplan incall uri sip** {*from* | *request* | *to* | *via*} **uri**

Use the **show dialplan incall uri** to display the dial peer that is matched for a specific URI in an incoming voice call. The following sample output shows the dial peer that is matched for an incoming call, based on the selected URI:

```
Router# show dialplan incall uri sip via sip:9.13.38.83
 Inbound VoIP dialpeer matching based on SIP URI's

VoiceOverIpPeer100
        peer type = voice, system default peer = FALSE, information type = voice,
        description = `',
```

```
tag = 100, destination-pattern = `',
voice reg type = 0, corresponding tag = 0,
allow watch = FALSE
answer-address = `', preference=0,
CLID Restriction = None
CLID Network Number = `'
CLID Second Number sent
CLID Override RDNIS = disabled,
rtp-ssrc mux = system
source carrier-id = `',target carrier-id = `',
source trunk-group-label = `',target trunk-group-label = `',
numbering Type = `unknown'
group = 100, Admin state is up, Operation state is up,
incoming called-number = `', connections/maximum = 0/unlimited,
DTMF Relay = disabled,
modem transport = system,
URI classes:
    Incoming (Request) =
    Incoming (Via) = 1
    Incoming (To) =
    Incoming (From) =
    Destination =
huntstop = disabled,
in bound application associated: 'DEFAULT'
out bound application associated: ''
dnis-map =
permission :both
incoming COR list:maximum capability
outgoing COR list:minimum requirement
outgoing LPCOR:
Translation profile (Incoming):
Translation profile (Outgoing):
incoming call blocking:
translation-profile = `'
disconnect-cause = `no-service'
advertise 0x40 capacity_update_timer 25 addrFamily 4 oldAddrFamily 4
mailbox selection policy: none
type = voip, session-target = `',
technology prefix:
settle-call = disabled
ip media DSCP = ef, ip media rsvp-pass DSCP = ef
ip media rsvp-fail DSCP = ef, ip signaling DSCP = af31,
ip video rsvp-none DSCP = af41,ip video rsvp-pass DSCP = af41
ip video rsvp-fail DSCP = af41,
ip defending Priority = 0, ip preemption priority = 0
ip policy locator voice:
ip policy locator video:
UDP checksum = disabled,
session-protocol = sipv2, session-transport = system,
req-qos = best-effort, acc-qos = best-effort,
req-qos video = best-effort, acc-qos video = best-effort,
req-qos audio def bandwidth = 64, req-qos audio max bandwidth = 0,
req-qos video def bandwidth = 384, req-qos video max bandwidth = 0,
RTP dynamic payload type values: NTE = 101
Cisco: NSE=100, fax=96, fax-ack=97, dtmf=121, fax-relay=122
       CAS=123, TTY=119, ClearChan=125, PCM switch over u-law=0,
       A-law=8, GSMAMR-NB=117 iLBC=116, AAC-ld=114, iSAC=124
       lmr_tone=0, nte_tone=0
       h263+=118, h264=119
       G726r16 using static payload
       G726r24 using static payload
RTP comfort noise payload type = 19
fax rate = voice,   payload size =  20 bytes
fax protocol = system
```

```
            fax-relay ecm enable
            Fax Relay ans enabled
            Fax Relay SG3-to-G3 Enabled (by system configuration)
            fax NSF = 0xAD0051 (default)
            codec = g729r8,   payload size =  20 bytes,
            video codec = None
            voice class codec = `'
            voice class sip session refresh system
            voice class sip rsvp-fail-policy voice post-alert mandatory keep-alive interval
30
            voice class sip rsvp-fail-policy voice post-alert optional keep-alive interval 30
            voice class sip rsvp-fail-policy video post-alert mandatory keep-alive interval
30
            voice class sip rsvp-fail-policy video post-alert optional keep-alive interval 30
            text relay = disabled
            Media Setting = forking (disabled) flow-through (global)
            Expect factor = 10, Icpif = 20,
            Playout Mode is set to adaptive,
            Initial 60 ms, Max 1000 ms
            Playout-delay Minimum mode is set to default, value 40 ms
            Fax nominal 300 ms
            Max Redirects = 1, signaling-type = cas,
            VAD = enabled, Poor QOV Trap = disabled,
            Source Interface = NONE
            voice class sip url = system,
            voice class sip tel-config = system,
            voice class sip rel1xx = system,
            tvoice class sip outbound-proxy = system,
            voice class sip asserted-id = system,
            voice class sip privacy = system,
            voice class sip e911 = system,
            voice class sip history-info = system,
            voice class sip pass-thru headers = system,
            voice class sip pass-thru content unsupp = system,
            voice class sip pass-thru content sdp = system,
            voice class sip anat = system,
            voice class sip g729 annexb-all = system,
            voice class sip early-offer forced = system,
            voice class sip negotiate cisco = system,
            voice class sip reset timer expires 183 = system,
            voice class sip block 180 = system,
            voice class sip block 181 = system,
            voice class sip block 183 = system,
            voice class sip preloaded-route = system,
            voice class sip random-contact = system,
            voice class sip random-request-uri validate = system,
            voice class sip call-route p-called-party-id = system,
            voice class sip call-route history-info = system,
            voice class sip privacy-policy send-always = system,
            voice class sip privacy-policy passthru = system,
            voice class sip privacy-policy strip history-info = system,
            voice class sip privacy-policy strip diversion = system,
            voice class sip bandwidth audio  = system,
            voice class sip bandwidth video  = system,
            voice class sip error-code-override options-keepalive failure = system,
            voice class sip encap clear-channel  = system,
            voice class sip map resp-code 181 = system,
            voice class sip bind control = system,
            voice class sip bind media = system,
            voice class sip authenticate redirecting-number = system,
            redirect ip2ip = disabled
            local peer = false
            probe disabled,
            Secure RTP: system (use the global setting)
```

```
        voice class perm tag = `'
        Time elapsed since last clearing of voice call statistics never
        Connect Time = 0, Charged Units = 0,
        Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
        Accepted Calls = 0, Refused Calls = 0,
        Last Disconnect Cause is "",
        Last Disconnect Text is "",
        Last Setup Time = 0.
        Last Disconnect Time = 0.
Matched:    Digits: 0
Target:
```

Configuring an Outbound Dial Peer for URI Destinations

This task describes how to configure an outbound dial peer for calls placed to an URI destination.

Prerequisites

Write a Tcl IVR 2.0 script or VoiceXML document that implements call transfer to a SIP or TEL URI. For information, see the *TCL IVR API Version 2.0 Programming Guide* or *Cisco VoiceXML Programmer's Guide*.

Restrictions

- Outbound calls to a SIP URL cannot be placed to a public switched telephone network (PSTN) leg because dial-peer matching does not support this. Outbound calls to a TEL URL can be placed to a PSTN call leg. The telephone number is pulled from the URL and used as the destination number.
- Underscores are not supported in the From header or in the callinfo originationNum field of a SIP URI in a Tcl script or VoiceXML document.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **dial-peer voice** *tag* **voip**
4. **session target ipv4:***ip-address*
5. **session protocol sipv2**
6. **destination uri** *tag*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`dial-peer voice`** *`tag`* **`voip`** **Example:** `Router(config)# dial-peer voice 2 voip`	Enters dial peer configuration mode for a VoIP dial peer.
Step 4	**`session target ipv4:`***`ip-address`* **Example:** `Router(config-dial-peer)# session target ipv4:10.10.1.1`	(Optional) Specifies the IP address of the terminating gateway **Note** The **session target** command is required when placing a call to a TEL URI; it is optional for a SIP URI. If you do not configure a session target for a SIP URI, the call is sent to the host address in the SIP URI that is passed from the application initiating the outbound call.
Step 5	**`session protocol sipv2`** **Example:** `Router(config-dial-peer)# session protocol sipv2`	(Optional) Specifies the session protocol if you are using SIP for calls between the local and remote gateways. **Note** If you are using H.323 do not configure the **session protocol** command.
Step 6	**`destination uri`** *`tag`* **Example:** `Router(config-dial-peer)# destination uri 700`	Specifies the URI class that links voice calls to this dial peer. **Note** Before configuring the **destination uri** command, you should configure the URI voice class by using the **voice class uri** command.
Step 7	**`end`** **Example:** `Router(config-dial-peer)# end`	Exits dial peer voice configuration mode and enters privileged EXEC mode.

Verifying Dial-Peer Configuration for an Outgoing URI

Perform this task to display information about the dial-peer configuration for an outgoing URI. These **show** commands need not be entered in any specific order.

SUMMARY STEPS

1. **show running-config**

2. **show voice class uri** [*tag* | **summary**]
3. **show dialplan uri h323** *uri*

DETAILED STEPS

Step 1 **show running-config**

Use the **show running-config** command to display the configuration of the URI voice class and the outbound dial peer.

The following example shows that voice class 500 is assigned to dial peer 599, which matches the calls that have a pattern beginning with 123 in the outgoing SIP URI:

```
Router# show running-config
.
.
.
voice class uri  500 sip
 pattern 123...
!
!
dial-peer voice 599 voip
 session protocol sipv2
 session target ipv4:10.10.1.1
 destination uri 500
 codec g711ulaw
!
```

Step 2 **show voice class uri** [*tag* | **summary**]

Use the **show voice class uri** command to display the configuration of the voice class:

```
Router# show voice class uri 500

Voice URI class: 500
    Schema = sip
    pattern = 123...
```

To display a list of all voice classes, use the **show voice class uri summary** command.

```
Router# show voice class uri summary
__________________________________________
Class Name                       Schema
------------------------------------------
100                              sip
300                              tel
500                              sip
__________________________________________
```

Step 3 **show dialplan uri**

Use the **show dialplan uri** command to verify that the correct dial peer is matched for the URI.

The following example shows that dial peer 599 is linked to voice class 500, which matches the specified URI. It also shows that the operational state of the dial peer is up:

```
Router# show dialplan uri sip:123456

 Outbound dialpeer matching based on destination URI

VoiceOverIpPeer599
        peer type = voice, information type = voice,
        description = `',
        tag = 599, destination-pattern = `',
```

```
        answer-address = `', preference=0,
        CLID Restriction = None
        CLID Network Number = `'
        CLID Second Number sent
        source carrier-id = `', target carrier-id = `',
        source trunk-group-label = `',  target trunk-group-label = `',
        numbering Type = `unknown'
        group = 599, Admin state is up, Operation state is up,
        incoming called-number = `', connections/maximum = 0/unlimited,
        DTMF Relay = disabled,
        modem transport = system,
        URI classes:
            Incoming (Request) =
            Incoming (To) =
            Incoming (From) =
            Destination = 500
        huntstop = disabled,
        in bound application associated: 'DEFAULT'
        out bound application associated: ''
        dnis-map =
        permission :both
        incoming COR list:maximum capability
        outgoing COR list:minimum requirement
        Translation profile (Incoming):
        Translation profile (Outgoing):
        incoming call blocking:
        translation-profile = `'
        disconnect-cause = `no-service'
        type = voip, session-target = `10.10.1.1',
        technology prefix:
        settle-call = disabled
        ip media DSCP = ef, ip signaling DSCP = af31, UDP checksum = disabled,
        session-protocol = sipv2, session-transport = system, req-qos = best-ef
        acc-qos = best-effort,
        RTP dynamic payload type values: NTE = 101
        Cisco: NSE=100, fax=96, fax-ack=97, dtmf=121, fax-relay=122
               CAS=123, ClearChan=125, PCM switch over u-law=0,A-law=8
        RTP comfort noise payload type = 19
        fax rate = voice,   payload size =  20 bytes
        fax protocol = system
        fax-relay ecm enable
        fax NSF = 0xAD0051 (default)
        codec = g711ulaw,   payload size =  20 bytes,
        Expect factor = 0, Icpif = 20,
        Playout Mode is set to default,
        Initial 60 ms, Max 300 ms
        Playout-delay Minimum mode is set to default, value 40 ms
        Fax nominal 300 ms
        Max Redirects = 1, signaling-type = ext-signal,
        VAD = enabled, Poor QOV Trap = disabled,
        Source Interface = NONE
        voice class sip url = system,
        voice class sip rel1xx = system,
        voice class perm tag = `'
        Time elapsed since last clearing of voice call statistics never
        Connect Time = 0, Charged Units = 0,
        Successful Calls = 0, Failed Calls = 0, Incomplete Calls = 0
        Accepted Calls = 0, Refused Calls = 0,
        Last Disconnect Cause is "",
        Last Disconnect Text is "",
        Last Setup Time = 0.
Matched:    Digits: 0
Target:
```

Note For a description of the fields shown in this output, see the **show dialplan uri** command in the *Cisco IOS Voice Command Reference.*

Troubleshooting Tips

- Do not use underscores in the From header or in the callinfo originationNum field of a SIP URI in a Tcl script or VoiceXML document. Using underscores in these fields, as shown in the following examples, causes call transfer to fail:

```
set callInfo(originationNum) "sip:firstname_lastname@example.com"

set headers(From) "sip:firstname_lastname"
```

- Use Tcl **puts** commands or VoiceXML log commands in your script to help with debugging. To display the output from these commands, use the **debug voip ivr script** command for Tcl IVR 2.0 scripts, or the **debug vxml puts** command for VoiceXML documents.

  For information on using the Tcl **puts** command, see the *TCL IVR API Version 2.0 Programming Guide*. For information about the VoiceXML log command, see the *Cisco VoiceXML Programmer's Guide*.

Troubleshooting URI Matching

Perform this task to troubleshoot URI matching. These **debug** commands need not be entered in any specific order.

SUMMARY STEPS

1. **debug dialpeer**
2. **debug voice uri**
3. **debug voice ccapi error**

DETAILED STEPS

Step 1 **debug dialpeer**

Use the **debug dialpeer** command to determine whether there is a dial peer with a voice class that matches the URI in the Tcl script or VoiceXML document.

In the following example, the incoming call is matched to dial peer 1000, then the gateway searches through the dial peers looking for a voice class that matches the URI in the outbound script:

```
Router# debug dialpeer

dialpeer detailed info debugging is on

*Mar  1 00:56:40.711: dpAssociateIncomingPeerSPI:
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_INCOMING_DNIS)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match incoming called number; called
(50267)
*Mar  1 00:56:40.711: dpMatchCore:
*Mar  1 00:56:40.711: dpMatchCore: dialstring(50267); expanded string(50267); calling()
```

```
*Mar  1 00:56:40.711: FillTarget: pDest(50267T) pPatn(50267)
*Mar  1 00:56:40.711: MatchNextPeer: peer 1000 matched
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=0 after DP_MATCH_INCOMING_DNIS;
peers (0x62F2E454)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerSPI:
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_INCOMING_DNIS)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match incoming called number; called
(50267)
*Mar  1 00:56:40.711: dpMatchCore:
*Mar  1 00:56:40.711: dpMatchCore: dialstring(50267); expanded string(50267); calling()
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=-1 after DP_MATCH_INCOMING_DNIS;
peers (0x0)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_ANSWER)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match answer address; calling (50006)
*Mar  1 00:56:40.711: dpMatchCore:
*Mar  1 00:56:40.711: dpMatchCore: dialstring(); expanded string(); calling(50006T)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=-1 after DP_MATCH_ANSWER; peers
(0x0)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_ORIGINATE)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match destination pattern; calling
(50006)
*Mar  1 00:56:40.711: dpMatchCore:
*Mar  1 00:56:40.711: dpMatchCore: dialstring(); expanded string(); calling(50006T)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=-1 after DP_MATCH_ORIGINATE;
peers (0x0)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_PORT)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=-1 after DP_MATCH_PORT; peers
(0x0)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_SRC_CARRIER)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=-1 after DP_MATCH_SRC_CARRIER;
peers (0x0)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerSPI:
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match rule (DP_MATCH_INCOMING_DNIS)
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Match incoming called number; called
(50267)
*Mar  1 00:56:40.711: dpMatchCore:
*Mar  1 00:56:40.711: dpMatchCore: dialstring(50267); expanded string(50267); calling()
*Mar  1 00:56:40.711: FillTarget: pDest(50267T) pPatn(50267)
*Mar  1 00:56:40.711: MatchNextPeer: peer 1000 matched
*Mar  1 00:56:40.711: dpAssociateIncomingPeerCore: Result=0 after DP_MATCH_INCOMING_DNIS;
peers (0x62F2E454)
*Mar  1 00:56:49.203: dpMatchPeersMoreArg:
*Mar  1 00:56:49.203: dpMatchPeersCore:
*Mar  1 00:56:49.203: dpMatchPeersCore: Match rule (DP_MATCH_DEST_URI_AND_TGT_CARRIER)
*Mar  1 00:56:49.203: dpMatchPeersCore: Result=-1 after DP_MATCH_DEST_URI_AND_TGT_CARRIER
*Mar  1 00:56:49.203: dpMatchPeersCore: Match rule (DP_MATCH_DEST_AND_TGT_CARRIER)
*Mar  1 00:56:49.203: dpMatchPeersCore: Result=-1 after DP_MATCH_DEST_AND_TGT_CARRIER
*Mar  1 00:56:49.203: dpMatchPeersCore: Match rule (DP_MATCH_DEST_URI)
*Mar  1 00:56:49.203: dpMatchPeersCore: Match Dest. URI; URI
(sip:xxx_no_under@example.com?Subject=Hello&Priority=Urgent&testID=AL_FEAT_SIP_URL_O_RV_11
)
*Mar  1 00:56:49.203: dpMatchCore:
*Mar  1 00:56:49.203: dpMatchCore: dialstring(); expanded string(); calling()
*Mar  1 00:56:49.203: dpMatchPeersCore: Result=-1 after DP_MATCH_DEST_URI
*Mar  1 00:56:49.203: dpMatchPeersCore: Match rule (DP_MATCH_DEST)
*Mar  1 00:56:49.203: dpMatchPeersCore: Result=-1 after DP_MATCH_DEST
*Mar  1 00:56:49.203: dpMatchPeersCore: Match rule (DP_MATCH_TGT_CARRIER)
*Mar  1 00:56:49.203: dpMatchPeersCore: Result=-1 after DP_MATCH_TGT_CARRIER
*Mar  1 00:56:49.203: //-1//TCL2:HN0033E40C:/tcl_PutsCmd:
 STATUS=ls_004
*Mar  1 00:56:49.203:
```

A message showing Result=-1 indicates that a dial peer was not matched. Because this output does not display another dial peer match statement, you know that the gateway failed to find a match.

Step 2 **debug voice uri**

Use the **debug voice uri** command to verify whether there is a voice class that matches the URI in the Tcl script or VoiceXML document.

In the following example, the gateway failed to match the URI in the script to the only configured voice class, 805:

```
Router# debug voice uri

Voice URI debugging is enabled
Router#
*Mar  1 01:09:03.319: vuri_match_class: tag (805)
*Mar  1 01:09:03.319: vuri_match_class_sip: Match with phone context
*Mar  1 01:09:03.319: vuri_match_class_sip: input ()
*Mar  1 01:09:03.319: vuri_match_class_sip: Match with host
*Mar  1 01:09:03.319: vuri_match_class_sip: input (example.com)
*Mar  1 01:09:03.319: vuri_match_class_sip: Match with user-id
*Mar  1 01:09:03.319: vuri_match_class_sip: input (xxx_no_under)
*Mar  1 01:09:03.319: vuri_match_class_sip: Match failed
```

Step 3 **debug voice ccapi error**

Use the **debug voice ccapi error** command to display any errors that might occur during the call transfer:

```
Router# debug voice ccapi error

*Mar  1 01:39:59.319: //32/7FAB684B8022/CCAPI/cc_get_associated_stream: Matched the
                                    Stream CallID 0x21::
*Mar  1 01:40:04.279: //32/7FAB684B8022/CCAPI/cc_api_call_disconnect_done:
cause=28,retry=0,vcCauseCode=0
```

Any output from this command means that the call transfer did not complete, although the reason for the error might not be obvious from the messages.

Configuration Examples for SIP and TEL URL Support

This section provides the examples that show the configuration for an originating and terminating gateway, each configured to handle both SIP and TEL URIs.

- Example: Outbound Dial-Peer Originating Gateway, page 21
- Example: Outbound Dial-Peer Terminating Gateway, page 26

Note For information about reading and passing SIP headers, see the *Cisco IOS SIP Configuration Guide*.

Example: Outbound Dial-Peer Originating Gateway

This following sections provide examples of SIP and TEL URIs with Header passing using the respective protocols in the originating gateway.

- Example: URIs with Header Passing Using SIP Protocol and H.323 Protocol, page 27

Example: URIs with Header Passing Using SIP Protocol and H.323 Protocol

In the following example, when a SIP call comes into the originating gateway, the application sip_headers_tcl asks the caller to enter an account number. After the account number is collected, it is assigned to a header named AccountInfo. The AccountInfo is passed to the leg setup along with other standard and user-defined headers in the destination URL.

The outgoing call matches on dial peer 766, which is configured with the **destination uri** command to match on voice class 766. Voice class 766 is configured with the **voice class uri** command to match destination SIP URI sip:elmo@sip.example.com. The gateway places a call to sip:elmo@sip.example.com.

When the call setup request is received by the gateway, the headers that are passed from the application are included in the SIP INVITE message.

In the following example, when an H.323 call comes into the originating gateway, the application tel_headers_vxml asks the caller to enter an account number. After the account number is collected, it is assigned to a header named AccountInfo. The AccountInfo is passed to the leg setup along with other user-defined headers in the destination URL.

The outgoing call matches on dial peer 767, which is configured with the **destination uri** command to match on voice class 767. Voice class 767 is configured with the **voice class uri** command to match telephone number 555-0100 and a phone context of 408. The voice class can also match against a URL pattern. This example uses the phone number and phone context as a matching criteria. The gateway places a call to a TEL URL with the header of tel:555-0100;phone-context=408;tsp=example.com;Subject=HelloTelVXML;To=oscar@example.com; From=nobody; Priority=urgent'+';AccountInfo='+acctInfo.

When the call setup request is received by the gateway, the destination URL with headers and parameters is passed in the setup message as part of the destination address.

```
!
version x.x
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
no service password-encryption
service internal
!
hostname 10.7.102.32
!
no logging buffered
enable password lab
!
username 1111
username 2222 password 0 2222
!
!
resource-pool disable
!
aaa new-model
!
!
aaa authentication login h323 local group radius
aaa authorization exec h323 group radius
aaa accounting connection h323 start-stop group radius
aaa session-id common
ip subnet-zero
ip ftp username dump
ip ftp password dump123
ip host px1-sun 10.0.1.0
ip host rtsp-ws 10.1.1.0
ip host dev 10.3.0.1
```

```
ip host px1-exm.example.com 10.0.1.1
ip host example.com 10.1.1.2
!
!
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
!
voice service voip
 sip
!
!
voice class uri  766 sip
 pattern elmo@sip.example.com*
!
voice class uri  767 tel
 phone number 767....
 phone context 408
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
ivr prompt memory 16384
ivr record memory system 256000
ivr record memory session 256000
ivr asr-server rtsp://nuance-asr/recognizer
ivr tts-server rtsp://nuance-asr/synthesizer
ivr record memory system 256000
ivr record memory session 256000
ivr asr-server rtsp://nuance-asr/recognizer
ivr tts-server rtsp://nuance-asr/synthesizer
rtsp client session history duration 1
rtsp client session history records 1
http client cache memory pool 0
http client cache memory file 10000
http client cache refresh 30
http client connection timeout 10
http client response timeout 10
fax interface-type modem
mta receive maximum-recipients 0
call-history-mib retain-timer 0
call-history-mib max-size 0
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
 clock source line secondary 1
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
controller T1 2
 framing esf
```

```
 clock source line secondary 2
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
controller T1 3
 framing esf
 clock source line secondary 3
 linecode b8zs
 cablelength short 133
 pri-group timeslots 1-24
!
gw-accounting h323
gw-accounting h323 vsa
gw-accounting voip
!
!
interface Ethernet0
 ip address 209.165.201.1 255.255.255.224
 ip helper-address 10.1.201.30
 no ip route-cache
 no ip mroute-cache
 no cdp enable
!
interface Serial0
 no ip address
 shutdown
 clockrate 2015232
 no fair-queue
 no cdp enable
!
interface Serial1
 no ip address
 shutdown
 clockrate 2015232
 no fair-queue
 no cdp enable
!
interface Serial2
 no ip address
 shutdown
 clockrate 2015232
 no fair-queue
 no cdp enable
!
interface Serial3
 no ip address
 shutdown
 clockrate 2015232
 no fair-queue
 no cdp enable
!
interface Serial0:23
 no ip address
 no logging event link-status
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 isdn disconnect-cause 1
 fair-queue 64 256 0
 no cdp enable
!
interface Serial1:23
 no ip address
```

```
 no logging event link-status
 isdn switch-type primary-5ess
 no cdp enable
!
interface Serial2:23
 no ip address
 no logging event link-status
 isdn switch-type primary-5ess
 no cdp enable
!
interface Serial3:23
 no ip address
 no logging event link-status
 isdn switch-type primary-5ess
 no cdp enable
!
interface FastEthernet0
 no ip address
 shutdown
 duplex auto
 speed auto
 no cdp enable
!
ip default-gateway 10.255.255.255
ip classless
ip route 209.165.200.20 255.255.255.224
no ip http server
ip pim bidir-enable
!
!
no cdp run
!
!
radius-server retransmit 3
radius-server authorization permit missing Service-Type
call rsvp-sync
!
application
 service sip_headers_tcl tftp://dev/demo/TCL/scripts/sip_headers.tcl
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://dev/demo/AUDIO/en/
 !
 service sip_headers_vxml tftp://dev/demo/VXML/scripts/sip_headers.vxml
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://dev/demo/AUDIO/en/
 !
 service tel_headers_tcl tftp://dev/demo/TCL/scripts/tel_headers.tcl
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://dev/demo/AUDIO/en/
 !
 service tel_headers_vxml tftp://dev/demo/VXML/scripts/tel_headers.vxml
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://dev/demo/AUDIO/en/
 !
!
voice-port 0:D
!
voice-port 1:D
!
voice-port 2:D
```

```
!
voice-port 3:D
!
mgcp modem passthrough voip mode ca
no mgcp timer receive-rtcp
!
mgcp profile default
!
dial-peer cor custom
!
!
dial-peer voice 1 pots
 service sip_headers_tcl
 incoming called-number 52948
 port 0:D
!
dial-peer voice 2 pots
 application tel_headers_vxml
 incoming called-number 52950
 port 0:D
!
dial-peer voice 767 voip
 session target ipv4:10.0.0.1
 destination uri 767
 codec g711ulaw
!
dial-peer voice 766 voip
 session protocol sipv2
 session target ipv4:10.0.0.1
 destination uri 766
 codec g711ulaw
!
dial-peer voice 7671234 voip
 service get_headers_vxml out-bound
 destination-pattern .......
 session protocol sipv2
 session target ipv4:10.0.0.1
 codec g711ulaw
!
sip-ua
 sip-server ipv4:10.0.1.1
!
!
line con 0
 exec-timeout 0 0
line aux 0
line vty 0 4
!
exception core-file special3
exception dump 10.7.100.1
end
```

Example: Outbound Dial-Peer Terminating Gateway

This section provides examples of SIP and TEL URIs with Header passing using the respective protocols in the terminating gateway.

- Example: URIs with Header Passing Using SIP Protocol and H.323 Protocol, page 27

Example: URIs with Header Passing Using SIP Protocol and H.323 Protocol

In the following example, when the call arrives at the terminating gateway and dial peer 766 is matched, the gateway stores all headers received in the incoming INVITE message so these can be accessed by the application.

The inbound dial peer can be configured to match the request-URI, or the “To” or “From” header in the incoming INVITE message. This example uses the request-URI for matching. The incoming call matches on dial peer 766, which is configured with the **incoming uri request** command to match on voice class 766. Voice class 766 is configured to match the incoming SIP request-URI sip:elmo@sip.example.com.

When the call is handed to the application configured in the inbound dial peer, get_headers_tcl, this Tcl application can read any header that is part of the incoming INVITE message.

In the following example, when the call arrives at the terminating gateway and dial peer 767 is matched, the gateway stores the incoming URI so it can be accessed by the application.

The inbound dial peer can be configured to match the entire TEL URL pattern, the E.164 number portion, or the phone context of the TEL URL. This example uses the phone number and phone context for matching. The incoming call matches on dial peer 767, which is configured with the **incoming uri called** command to match on voice class 767. Voice class 767 is configured to match the incoming called URL with the header of tel:555-0100;phone-context=408;tsp=example.com;Subject=HelloTelVXML;To=oscar@example.com;From=nobody;Priority=urgent'+';AccountInfo='+acctInfo.

When the call is handed to the application configured in the inbound dial peer, get_headers_vxml, this VoiceXML application can read any header which is part of the incoming called URI received in the setup indication.

```
!
version x.x
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
!
hostname as5300-09
!
enable secret 5 $1$KRsb$cFAQFOylLr9j5Fof.eLgx1
enable password lab
!
!
!
resource-pool disable
clock timezone PDT -8
clock calendar-valid
!
ip subnet-zero
no ip domain lookup
ip domain name fieldlabs.example.com
ip host dcl1server 10.7.108.2
ip host px1-sun 10.14.99.1
ip host dirt 192.168.254.254
ip host jurai 192.168.254.254
ip host dclserver 10.7.108.2
ip host dcl2server 10.7.112.2
ip host ts 10.7.100.1
!
!
isdn switch-type primary-5ess
isdn voice-call-failure 0
```

```
!
!
voice service voip
 sip
  header-passing
!
!
voice class uri  766 sip
 pattern elmo@sip.example.com*
!
voice class uri  767 tel
 phone number 767....
 phone context 408
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
!
ivr record memory system 100000
ivr record memory session 100000
ivr record memory system 100000
ivr record memory session 100000
fax interface-type modem
mta receive maximum-recipients 0
!
controller T1 0
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing sf
 clock source line secondary 1
 linecode ami
!
controller T1 2
 framing sf
 linecode ami
!
controller T1 3
 framing sf
 linecode ami
!
!
!
interface Ethernet0
 ip address 209.165.200.225 255.255.255.224
 ip helper-address 192.168.254.254
 no ip route-cache
 no ip mroute-cache
 no cdp enable
!
interface Serial0:23
 no ip address
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 fair-queue 64 256 0
 no cdp enable
```

```
!
interface FastEthernet0
 ip address 209.165.201.28 255.255.255.224
 no ip route-cache
 no ip mroute-cache
 duplex half
 speed 10
 no cdp enable
!
ip default-gateway 10.7.0.1
ip classless
ip route 209.168.201.1 255.255.255.224 10.165.196.1
ip route 209.168.201.5 255.255.255.224 10.165.0.1
no ip http server
ip pim bidir-enable
!
!
no cdp run
!
!
call rsvp-sync
!
application
 service get_headers_tcl tftp://dev/demo/TCL/scripts/get_headers.tcl
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://dirt/cchiu/AUDIO/en/
 !
 service voice get_headers_vxml tftp://dev/demo/VXML/scripts/get_headers.vxml
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://dirt/cchiu/AUDIO/en/
 !
!
voice-port 0:D
!
mgcp ip qos dscp cs5 media
mgcp ip qos dscp cs3 signaling
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1 pots
 service test
 incoming called-number 52950
 port 0:D
!
dial-peer voice 767 voip
 service get_headers_vxml
 session target ipv4:10.0.0.1
 incoming uri called 767
 codec g711ulaw
!
dial-peer voice 766 voip
 service get_headers_tcl
 session protocol sipv2
 session target ipv4:10.0.0.0
 incoming uri request 766
 codec g711ulaw
!
dial-peer voice 2 pots
```

```
 destination-pattern 767....
 port 0:D
 prefix 9767
!
sip-ua
!
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
line vty 0 4
 password lab
 login
!
scheduler interval 1000
end
```

Where to Go Next

- To configure properties for audio files, see "Configuring Audio File Properties for Tcl and VoiceXML Applications".
- To configure voice recording using a VoiceXML application, see "Configuring VoiceXML Voice Store and Forward".
- To configure properties for speech recognition or speech synthesis, see "Configuring ASR and TTS Properties".
- To configure a VoiceXML fax detection application, see "Configuring Fax Detection for VoiceXML".
- To configure telephony call-redirect features for voice applications, see "Configuring Telephony Call-Redirect Features".
- To configure session interaction for a Tcl IVR 2.0 application, see "Configuring Tcl IVR 2.0 Session Interaction".
- To monitor and troubleshoot voice applications, see "Monitoring and Troubleshooting Voice Applications".

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Cisco IOS Voice commands	Cisco IOS Voice Command Reference
Overview of Cisco Tcl IVR and VoiceXML Applications	*Cisco IOS Tcl IVR and VoiceXML Application Guide*

MIBs

MIB	MIBs Link
• CISCO-VOICE-DIAL-CONTROL-MIB • CISCO-VOICE-DNIS-MIB	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring SIP and TEL URL Support

Table 11-1 lists the features in this module and provides links to specific configuration information.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 11-1 lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 11-1 Feature Information for Configuring SIP and TEL URL Support

Feature Name	Releases	Feature Information
SIP and TEL URL support	12.3(4)T 12.3(14)T	SIP and TEL URL support enables Cisco gateways to direct incoming calls to a voice application based on the URL and Tcl IVR 2.0 and VoiceXML applications to place outbound calls to a Session Initiation Protocol (SIP) or telephone (TEL) URL. It expands call-control capabilities by allowing voice applications to use URL destinations and dialing plans to be implemented using SIP or TEL URLs rather than telephone numbers. The following sections provide information about this feature: • Configuring an Inbound Dial Peer to Match on a URI, page 7 • Configuring an Outbound Dial Peer for URI Destinations, page 15
Support Inbound Dial-peer Match Based on Remote IP Address on SIP Trunks	15.1(2)T	The Support Inbound Dial-peer Match Based on Remote IP Address on SIP Trunks feature supports the expansion of inbound dial-peer matching logic to include matching based on the source IP address of inbound signaling on a SIP trunk. This feature enables enforcement of specific call-treatment, security, and routing policies on each SIP trunk. The following section provide information about this feature: • Configuring an Inbound Dial-Peer to Match the URI on SIP Calls, page 9 The following commands were introduced or modified: **dial-peer voice, voice-class uri**.

Monitoring and Troubleshooting Voice Applications

Event logs and statistics introduced in the Voice Application Monitoring and Troubleshooting Enhancements feature enable detailed monitoring of voice application instances and call legs. Records for terminated application instances and call legs are saved in history to assist in fault isolation. This comprehensive management information helps you diagnose problems in the network and identify the causes.

Tip To immediately begin using this feature, proceed to the "Enabling Event Logging and Statistics Globally for Voice Applications" section on page 10.

For more information about this and related Cisco IOS voice features, see the following:

- "Overview of Cisco IOS Tcl IVR and VoiceXML Applications" on page 1
- Entire Cisco IOS Voice Configuration Library—including library preface and glossary, other feature documents, and troubleshooting documentation—at http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_voice_configuration_library_glossary/vcl.htm.

Note For releases prior to Cisco IOS Release 12.3(14)T, see the previous version of the *Cisco Tcl IVR and VoiceXML Application Guide* at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/tcl_leg/index.htm

Feature History for Voice Application Monitoring and Troubleshooting Enhancements

Release	Modification
12.3(8)T	This feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.

Contents

- Prerequisites for Voice Application Monitoring and Troubleshooting, page 2
- Restrictions for Voice Application Monitoring and Troubleshooting, page 2

- Information About Voice Application Monitoring and Troubleshooting Enhancements, page 2
- How to Configure Monitoring for Voice Applications, page 10
- Additional References, page 32

Prerequisites for Voice Application Monitoring and Troubleshooting

- A Tcl IVR 2.0 or VoiceXML application must be configured on the voice gateway as described in Configuring Basic Functionality for Tcl IVR and VoiceXML Applications, or you must use one of the call applications contained in Cisco IOS software.
- Incoming telephony call legs including voice, fax, or modem.

Restrictions for Voice Application Monitoring and Troubleshooting

- Event logs for IP call legs are not supported.
- Tcl IVR 1.0 is not supported.
- Statistics and event logs for dynamically loaded scripts that are not configured on the gateway are not supported.

Information About Voice Application Monitoring and Troubleshooting Enhancements

To monitor and troubleshoot voice applications on the Cisco voice gateway, you should understand the following concepts:

- Description of Voice Application Monitoring and Troubleshooting Features, page 3
- Counters and Gauges for Voice Application Statistics, page 4
- Monitoring Levels for Voice Applications, page 5
- Benefits of Voice Application Monitoring and Troubleshooting Enhancements, page 8
- Guidelines for Enabling Statistics and Event Logging for Voice Applications, page 9

Description of Voice Application Monitoring and Troubleshooting Features

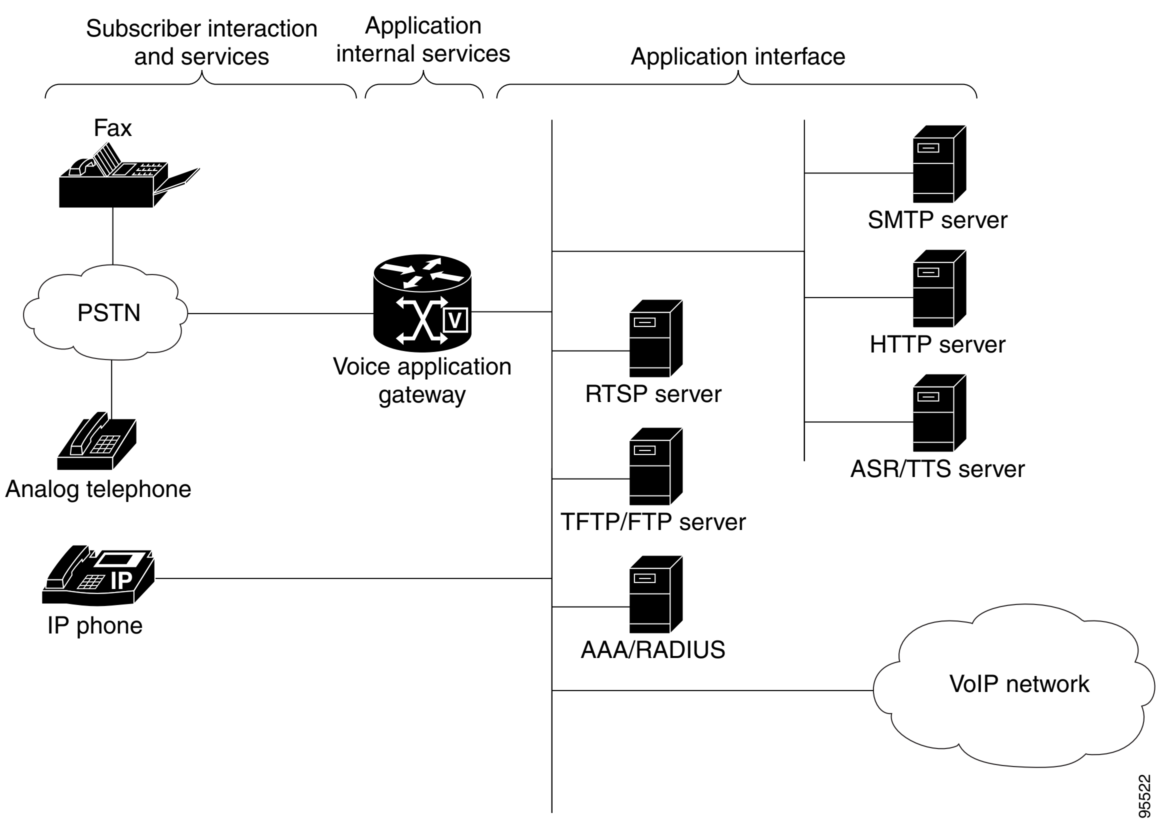
Voice calls are setup, maintained, and terminated by different subsystems within Cisco IOS software. Before Cisco IOS Release 12.3(x)T, managing calls from a voice application perspective could be challenging when diagnosing call progress, delays, exceptions, and failures during post call-analysis. Although debug commands are available for the individual subsystems, the volume of output and the expertise required to understand all the subsystems make it difficult to use **debug** commands for identifying the cause of a failure or gathering specific details about a call.

This feature enables detailed monitoring of call applications by providing command-based event logging and statistics collection for voice application instances, application interfaces, and call legs on the Cisco voice gateway. The event logs and statistics provide a high-level view of application transactions in simple, nontechnical language. Event logs include subscriber dialog interactions and back-end server transactions. This event logging is separate from the generic message logging available with the **logging buffered** command. You can enable monitoring globally for all voice applications and application interfaces, or for individual applications, interfaces, or servers. These comprehensive management features help you diagnose performance problems and isolate faults in a production-level network.

After an application instance terminates, the event log and statistics information is moved from the active table to the history table in the application information system (AIS) database. You can view the event logs and statistics by using **show** commands on the voice gateway or by using SNMP to access the MIB call application tables. You can set the size of the event log buffer to meet your needs and the memory resources of your voice gateway. You can also configure the voice gateway to write the event logs to an external FTP server. The gateway saves the event logs to the selected server when the event log buffer becomes full or when the application instance or call leg terminates.

Figure 12-1 shows the major subsystem components involved in voice application calls.

Figure 12-1 Voice Application Network Components



The voice application monitoring features provide:

- Statistics and event logs for voice applications including Tcl IVR 2.0, VoiceXML, T.37 (on-ramp/off-ramp), and the applications contained in Cisco IOS software such as the default session application.

- Counters maintained per application instance. Gauges maintained for application instance, application, and gateway levels.
- Counters and gauges for various external interfaces used by applications including AAA, ASR, HTTP, RTSP, SMTP, TFTP, and TTS servers, and flash memory and the gateway memory.
- Event logs in plain English for each application instance.
- Counters and gauges accessible through SNMP by call application MIB, or through Cisco IOS commands and posted to FTP server on request.
- Event logs for voice call legs.

Counters and Gauges for Voice Application Statistics

Statistics for voice applications and call legs are presented as counters and gauges. These measurements are based on event counts or rates. They measure the total count of a specific event, or the derived rate of change of events during a period of time. Counters track totals, such as the number of errors, and do not reset to zero unless cleared. Gauges reflect real-time information for active calls, such as the number of currently connected calls. Gauges fluctuate over time and can return to zero. For example, a call is included in an active call gauge while it is active. When the call terminates, the active call gauge is decreased while counters in the history table increase. Gauges are available at the gateway level and application level for active calls and instances.

The accumulated statistics are stored in the gateway. You can configure the maximum duration to store the historical records according to your needs and the availability of system memory. You can display the application statistics on the gateway by using **show** commands.

Monitoring Levels for Voice Applications

Statistics and event logs for voice application instances are organized in a hierarchical model, providing a top-down approach to monitoring. You can detect faults or performance irregularities at a global level, then drill down to more specific details to isolate a fault. Data is available for both active application instances and terminated instances. After an application instance terminates, its data records move from active to history.

You monitor voice applications at three levels. Data is collected for each application instance at the lowest, most detailed level and propagated up to the highest level where the data is consolidated and presented in summary form. The three monitoring levels are:

- Gateway level—High-level summary of all applications running on the Cisco voice gateway. The statistics are cumulative compiled from all instances of all applications on the gateway. Because there is only one record per gateway, the amount of information is brief and does not impact gateway resources so you can request it regularly. In most scenarios, you begin the monitoring process by displaying the gateway-level statistics with the **show call application gateway-level** command. An error condition might be shown or a value might exceed a threshold, indicating a fault in one or more application instances on the gateway. After an application instance terminates, its gateway-level statistics are added to the history counters.
- Application level—Statistics representing all instances of a particular application. There may be multiple records, one record for each configured application. If several records show the same error, you can identify which application is involved and ignore the others without the error. You display application-level statistics with the **show call application app-level** command. Statistics from this

level are propagated up to the gateway level where they are consolidated with statistics for all other applications on the gateway. After an application instance terminates, its application-level statistics are added to the history counters.

- Application instance (session) level—Detailed information for specific application instances. You display the data at this level by using the **show call application session-level** command. You can focus on the application instances that are affected by looking for records with similar errors. Event logs are also available at this level providing further details about an application instance. Statistics from this level are propagated up to the application level where they are consolidated by application. After an application instance terminates, its session records are appended to history.

You can also display application interface statistics for transactions between applications and back-end servers. For information, see the **show call application interface** command.

Table 12-1 lists the type of records available at each monitoring level.

Table 12-1 Statistics and Event Log Output

Level	Statistics		Event-Log	
	Active	**History**	**Active**	**History**
Gateway	Gauges	Counters	No	No
Application	Gauges	Counters	No	No
Application instance	Gauges, counters	Counters	Yes	Yes

Figure 12-2 illustrates the monitoring levels for identifying application instances affected by an error.

Figure 12-2 Application Monitoring Levels

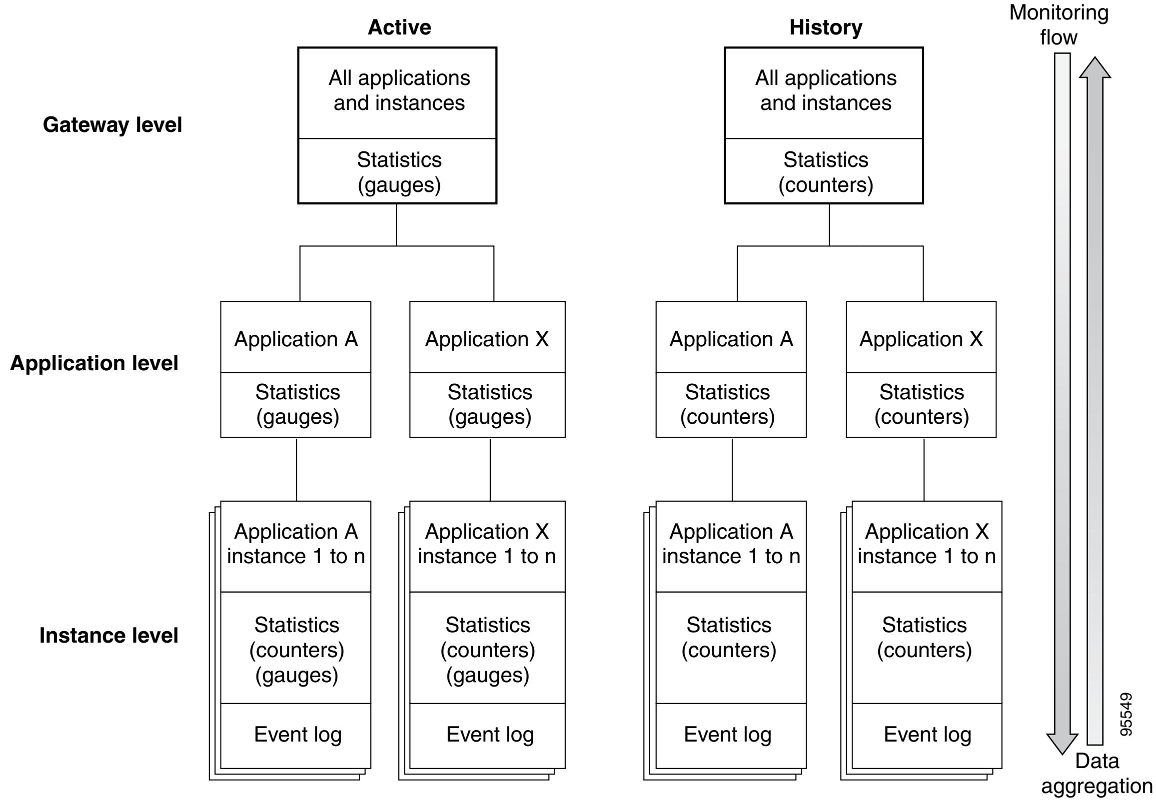


Figure 12-3 shows the type of information available for application interfaces.

Figure 12-3 Application Interface Event Logs and Statistics

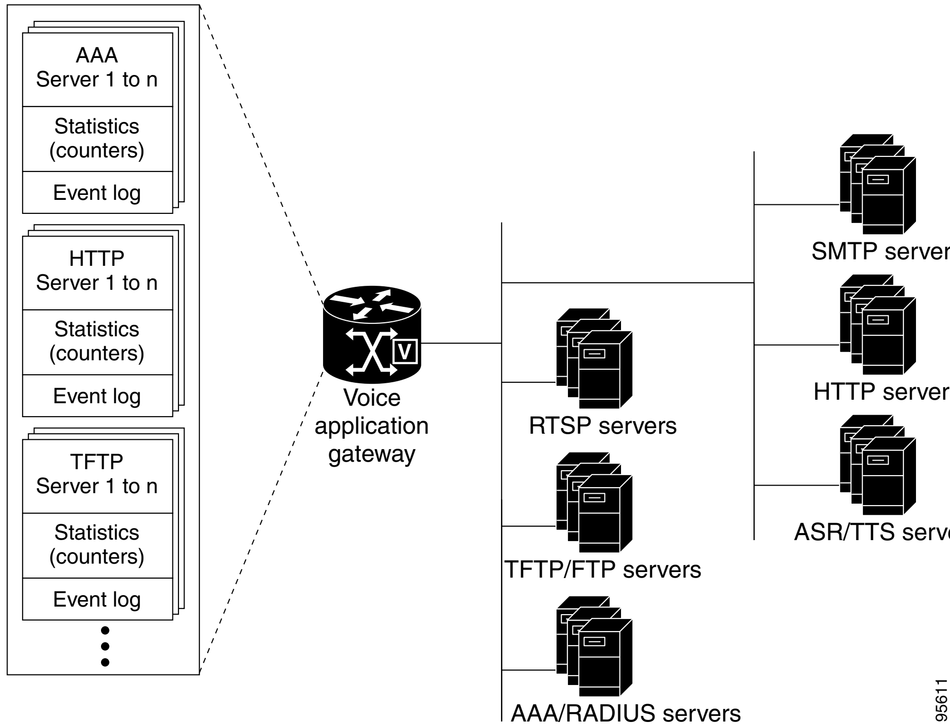
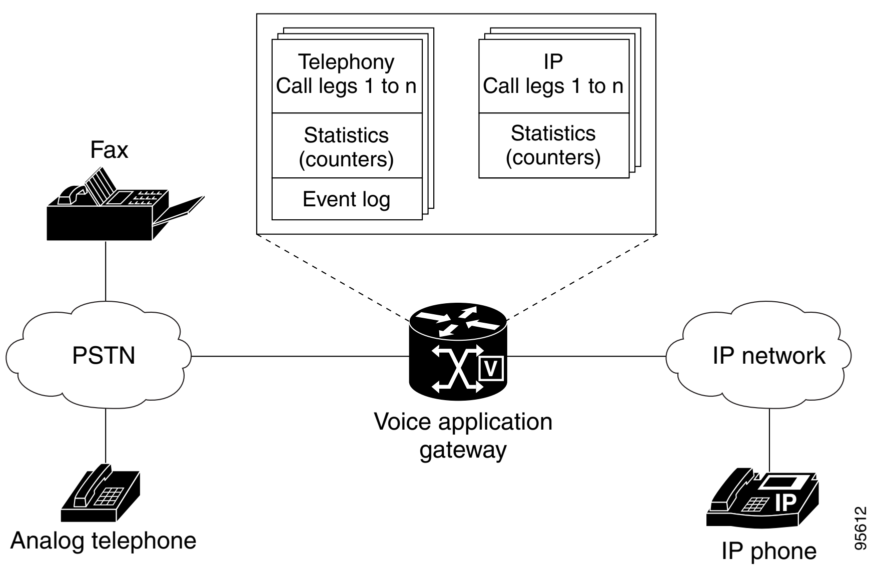


Figure 12-4 shows the type of information available for voice call legs.

Figure 12-4 Call Leg Event Logs and Statistics



Application Instance Statistics

The following statistics are available for application instances.

Subscriber Interaction

- DTMF attempts, matches, no matches, no input, and long pound
- ASR attempts, matches, no matches, and no input
- AAA authentication successes and failures
- AAA authorization successes and failures

Subscriber Services

- For incoming and outgoing PSTN and IP call legs:
  - Call legs setup, currently connected, and total legs connected
  - Call legs handed off, incoming and outgoing
  - Call legs handed off and returned, incoming and outgoing
  - Call legs disconnected for cause of normal, user error, or system error
- Media attempts, successes, failures, and currently active sessions for prompt playout, recording, and TTS on call legs seen by an application instance.

Application Internal Services

- Bridged handoffs, returned bridged handoffs, blind handoffs, and handoff failures for application instances
- Place call requests, successes, and failures
- Document fetch requests, successes, and failures
- Document submit requests, successes, and failures
- ASNL subscription attempts, successes, pending, failures, and notifications received

For a description of each statistic and an example of the output, see the **show call application session-level** command in the *Cisco IOS Voice Command Reference, Release 12.3T* at http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123tcr/123tvr/index.htm.

Application Interface Statistics

Statistics for application interfaces are available at the gateway level. The statistics are summarized for the gateway and can be viewed by service type or for a specific server. The following statistics are provided for each of the service types: AAA, ASR, flash memory, HTTP, gateway memory, RTSP, SMTP, TFTP, TTS.

- Read requests, successes, and failures
- Write requests, successes, and failures
- Bytes transmitted and received
- Rate (minimum, maximum, and average)

For a description of the specific statistics supported for each interface type, and an example of the output, see the **show call application interface** command in the *Cisco IOS Voice Command Reference, Release 12.3T* at http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123tcr/123tvr/index.htm.

Benefits of Voice Application Monitoring and Troubleshooting Enhancements

- Event logs are captured at run time unlike debug commands that are enabled after a problem occurs.

- Event log output uses simple wording unlike debug output that includes information specific to the internal system.
- Event logs can be selectively started and saved in the voice gateway startup configuration unlike debug commands that must be executed after each gateway reload.

Guidelines for Enabling Statistics and Event Logging for Voice Applications

The voice application monitoring and troubleshooting enhancements provide separate commands for enabling statistics and event logs. Statistics are useful for both monitoring and troubleshooting and you can enable statistics collection without a noticeable impact on system performance.

Event logging is designed primarily for troubleshooting faults after an error is indicated in the statistics. Depending on the amount of gateway traffic and the type of event logs or event log options that are enabled, the gateway could experience performance issues because of the impact on processor and memory resources. Use the following guidelines to selectively enable event logging:

- Enable event logging only when troubleshooting a problem
- Enable only the specific type of event log that is required to troubleshoot the problem (application, back-end server, call leg)
- Enable event logging only for a specific application if using application event logging

Throttling Mechanism for Event Logging

Event logging consumes processor memory to store the event logs. To prevent event logging from adversely impacting system resources for production traffic, the gateway uses a throttling mechanism. When free processor memory drops below 20%, the gateway automatically disables event logging for all new application sessions, application interfaces, and call legs for which logging is enabled. It resumes event logging when free memory rises above 30%.

While throttling is occurring, the gateway does not capture any new event logs even if event logging is enabled through Cisco IOS commands. You should monitor free memory on the gateway and enable event logging only when necessary for isolating faults.

Memory Requirements for Writing Event Logs to FTP

You can enable the gateway to write event logs to an external FTP server. This could adversely impact gateway memory resources in some scenarios, for example, when:

- The gateway is consuming high processor resources and FTP does not have enough processor resources to flush the logged buffers to the FTP server.
- The designated FTP server is not powerful enough to perform FTP transfers quickly enough
- Bandwidth on the link between the gateway and the FTP server is not large enough
- The gateway is receiving a high volume of short-duration calls or calls that are failing

You should enable FTP dumping only when necessary and not enable it in situations where it might adversely impact your system performance.

How to Configure Monitoring for Voice Applications

This section contains the following procedures:

- Enabling Event Logging and Statistics Globally for Voice Applications, page 10 (required)
- Enabling Event Logging for a Specific Voice Application or Interface, page 12 (required)
- Monitoring Voice Applications for Active Calls, page 14 (required)
- Monitoring Voice Applications for Terminated Calls, page 15 (required)
- Monitoring Voice Call Legs, page 20 (optional)
- Clearing Event Logs and Statistics for Application Instances and Interfaces, page 22 (optional)
- Displaying Event Logs for Applications or Call Legs in Real-Time, page 22 (optional)
- Modifying Event Log Settings for Application Instances, page 23 (optional)
- Modifying Event Log Settings for Application Interfaces, page 25 (optional)
- Modifying Event Log Settings for Call Legs, page 26 (optional)
- Modifying Event Log History Limits, page 27 (optional)

Enabling Event Logging and Statistics Globally for Voice Applications

Perform this task to enable event logs and statistics globally for all voice application instances, application interface types, and call legs.

Note This procedure enables event logs and statistics globally for all applications, application interface types, and call legs. To enable or disable event logs for specific applications or interface types, see the "Enabling Event Logging for a Specific Voice Application or Interface" section on page 12.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **monitor**
5. **event-log**
6. **stats**
7. **interface event-log**
8. **interface stats**
9. **exit**
10. **exit**
11. **call event-log**
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**application** **Example:** `Router(config)#application`	Enters application configuration mode.
Step 4	monitor **Example:** `Router(config-app)# monitor`	Enters application configuration monitor mode.
Step 5	**event-log** **Example:** `Router(config-app-monitor)# event-log`	Enables event logging for voice application instances.
Step 6	**stats** **Example:** `Router(config-app-monitor)# stats`	Enables statistics collection for voice applications.
Step 7	**interface event-log** **Example:** `Router(config-app-monitor)# interface event-log`	Enables event logging for all external interfaces used by voice applications.
Step 8	**interface stats** **Example:** `Router(config-app-monitor)# interface stats`	Enables statistics collection for application instances.
Step 9	**exit** **Example:** `Router(config-app-monitor)# exit`	Exits the application configuration monitor mode.
Step 10	**exit** **Example:** `Router(config-app)# exit`	Exits the application configuration mode.

	Command or Action	Purpose

Step 11	`call leg event-log`
	
	Example:
	`Router(config)# call leg event-log`
		Enables transaction event logging for voice, fax, and modem call legs.

Step 12	**`exit`** **Example:** `Router# exit`	Exits the current mode.

Enabling Event Logging for a Specific Voice Application or Interface

Perform this task to enable event logging and statistics collection for a specific voice application, application interface type, or server.

Note This procedure enables or disables event logs for specific applications, interface types, or servers. If you have already enabled event logs globally by performing the task in the "Enabling Event Logging and Statistics Globally for Voice Applications" section on page 10, you can skip this task unless you want to individually disable selected applications, interface types, or servers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** *application-name*
5. **paramspace appcommon event-log** [**enable** | **disable**]
6. **exit**
7. **monitor**
8. **interface event-log** {**aaa** | **asr** | **flash** | **http** | **ram** | **rtsp** | **smtp** | **tftp** | **tts**} [**server** *server*] [**disable**]
9. **exit**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**application** **Example:** `Router(config)#application`	Enters application configuration mode.
Step 4	**service** *application-name* **Example:** `Router(config-app)# service session`	Enter parameter configuration mode for the application.
Step 5	**paramspace appcommon event-log** [**enable** | **disable**] **Example:** `Router# paramspace appcommon event-log enable`	Enables event logging for a specific voice application.
Step 6	**exit** **Example:** `Router(config-app)# exit`	Exits configuration submode.
Step 7	**monitor** **Example:** `Router(config-app)#  monitor`	Enters application monitor configuration mode.
Step 8	**interface event-log** {**aaa** | **asr** | **flash** | **http** | **ram** | **rtsp** | **smtp** | **tftp** | **tts**} [**server** *server*] [**disable**]] **Example:** `Router(config-app-monitor)# interface event-log http disable`	Enables event logging for a specific external interface used by voice applications.
Step 9	**exit** **Example:** `Router# exit`	Exits the current mode.

Monitoring Voice Applications for Active Calls

Perform this task to do basic load monitoring of voice applications when detecting faults or unacceptable ranges while a call is still active.

SUMMARY STEPS

1. **enable**
2. **show call application active gateway-level**
3. **show call application active app-level summary**
4. **show call application active app-level app-tag** *application-name*
5. **show call application active session-level summary**
6. **show call application active session-level session-id** *session-id*
7. **show call application interface summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`show call application active gateway-level`** **Example:** `Router# show call application active gateway-level`	Displays gateway-level statistics for all currently active voice application instances.
Step 3	**`show call application active app-level summary`** **Example:** `Router# show call application active app-level summary`	Displays application-level statistics for all active voice applications.
Step 4	**`show call application active app-level app-tag`** *`application-name`* **Example:** `Router# show call application active app-level app-tag sample_app`	Displays application-level statistics for a specific voice application.
Step 5	**`show call application active session-level summary`** **Example:** `Router# show call application active session-level summary`	Displays statistics for all currently active voice application instances.

Step 6	**show call application active session-level session-id** *session-id* **Example:** `Router# show call application active session-level session-id 5`	Displays event logs and statistics for a specific voice application instance.
Step 7	**show call application interface summary** **Example:** `Router# show call application interface summary`	Displays aggregated statistics for all application interfaces.

For examples of the command output for the steps in this task, see the "Examples for Monitoring Voice Applications" section on page 16.

Monitoring Voice Applications for Terminated Calls

Perform this task to monitor voice applications, detect faults, and identify the source of faults after a call terminates.

SUMMARY STEPS

1. **enable**
2. **show call application history gateway-level**
3. **show call application history app-level summary**
4. **show call application history app-level app-tag** *application-name*
5. **show call application history session-level summary**
6. **show call application history session-level session-id** *session-id*
7. **show call application interface summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show call application history gateway-level** **Example:** `Router# show call application history gateway-level`	Displays gateway-level statistics for all voice application instances in the history table.
Step 3	**show call application history app-level summary** **Example:** `Router# show call application history app-level summary`	Displays summarized statistics from the history table for all voice applications.

Step 4	**show call application history app-level app-tag** *application-name* **Example:** `Router# show call application history app-level app-tag sample_app`	Displays statistics from the history table for a specific voice application.
Step 5	**show call application history session-level summary** **Example:** `Router# show call application history session-level summary`	Displays statistics from the history table for all voice application instances.
Step 6	**show call application history session-level session-id** *session-id* **Example:** `Router# show call application history session-level session-id 5`	Displays event logs and statistics for the specific voice application instance.
Step 7	**show call application interface summary** **Example:** `Router# show call application interface summary`	Displays aggregated statistics for all application interfaces.

Examples for Monitoring Voice Applications

This section includes output examples demonstrating the task for isolating faults in your voice application network. In this sample scenario, one of the audio prompts used by an application is not available from the specified TFTP server, either because the gateway does not have connectivity to the server or the audio file is not located on the specified server.

Sample Output for show call application history gateway-level Command

In the following example, the **show call application history gateway-level** command shows that a media failure occurred during a prompt playout. The counters in the output are an aggregation of all application instances that were run and terminated on the gateway. You must next determine which application had the error.

```
Router# show call application history gateway-level

Gateway level statistics for history application sessions:
Sessions w/ stats:              7
Last reset time:                *Aug 26 17:56:34 PST
 Subscriber Service - Call
                                                   PSTN              VOIP
                                           Incoming Outgoing  Incoming Outgoing
Legs setup:                                7        0         0        0
Total legs connected:                      7        0         0        0
Legs handed in:                            0        0         0        0
Legs handed in returned back:              0        0         0        0
Legs handed out:                           0        0         0        0
Legs handed out came back:                 0        0         0        0
Legs disconnected normally:                7        0         0        0
Legs disconnected for user error:          0        0         0        0
Legs disconnected for system error:        0        0         0        0

 Subscriber Service - Media
```

```
                                          Play       Record     TTS
Media attempts:                           9          0          1
Media successes:                          8          0          1
Media aborts:                             0          0          0
Media failures:                           1          0          0
Total media duration (in seconds):        57         0          17

 Application Internal Service - Document Read-Write
                                          Read       Write
Doc requests:                             1          0
Doc successes:                            1          0
Doc failures:                             0          0

 Subscriber Interaction - DTMF
DTMFs not matched:                        0
DTMFs matched:                            3
DTMFs no input:                           0
DTMFs long pound:                         0
```

Sample Output for show call application history app-level summary Command

In the following example, the **show call application history app-level summary** command lists all applications that are loaded on the gateway and shows that an error occurred with the application named menu. You can then view statistics for this specific application.

```
Router# show call application history app-level summary

Application level history Info:
                                   Sessions                     Last Reset
App Name                 Stats w/ Stats   Total      Errors     Time
session                  N      0          0          0
fax_hop_on               N      0          0          0
clid_authen              N      0          0          0
clid_authen_collect      N      0          0          0
clid_authen_npw          N      0          0          0
clid_authen_col_npw      N      0          0          0
clid_col_npw_3           N      0          0          0
clid_col_npw_npw         N      0          0          0
Default                  N      0          0          0
lib_off_app              N      0          0          0
fax_on_vfc_onramp_app    N      0          0          0
debitcard_20             N      0          0          0
aaa_tcl                  N      0          0          0
coapp                    N      0          0          0
doc_write                Y      1          1          0          *Aug 26 18:21:07
generic                  Y      3          3          0          *Aug 26 18:05:26
menu                     Y      3          3          1          *Aug 26 18:42:48
```

Sample Output for show call application history app-level app-tag Command

In the following example, the **show call application history app-level app-tag** command for the application named menu shows that there was a media failure when the application tried to play an audio prompt. You must next determine which instance of the application had the error.

```
Router# show call application history app-level app-tag menu

Application level history Info:

Application name:             menu
URL:                          tftp://sample/menu.vxml
Total sessions:               3
Sessions w/ stats:            3
Last reset time:              *Aug 26 18:42:48 PST
```

```
 Subscriber Service - Call
                                           PSTN               VOIP
                                    Incoming Outgoing  Incoming Outgoing
Legs setup:                         3        0         0        0
Total legs connected:               3        0         0        0
Legs handed in:                     0        0         0        0
Legs handed in returned back:       0        0         0        0
Legs handed out:                    0        0         0        0
Legs handed out came back:          0        0         0        0
Legs disconnected normally:         3        0         0        0
Legs disconnected for user error:   0        0         0        0
Legs disconnected for system error: 0        0         0        0

 Subscriber Service - Media
                                    Play       Record     TTS
Media attempts:                     6          0          0
Media successes:                    5          0          0
Media aborts:                       0          0          0
Media failures:                     1          0          0
Total media duration (in seconds):  29         0          0

 Subscriber Interaction - DTMF
DTMFs not matched:                  0
DTMFs matched:                      3
DTMFs no input:                     0
DTMFs long pound:                   0
```

Sample Output for show call application history session-level summary Command

In the following example, the **show call application history session-level summary** command shows all the instances in history for the menu application. You can next review the statistics and event log for the specific instance that had the error, which has a session ID of 14.

```
Router# show call application history session-level summary

SID  Application Name       Stat Err Cnt    Log Stop Time       Duration
7    generic                Y    0          Y   *Aug 26 18:05:2 00:00:12
8    generic                Y    0          Y   *Aug 26 18:05:3 00:00:10
9    generic                Y    0          Y   *Aug 26 18:05:4 00:00:09
D    doc_write              Y    0          Y   *Aug 26 18:21:0 00:00:17
12   menu                   Y    0          Y   *Aug 26 18:42:4 00:00:11
13   menu                   Y    0          Y   *Aug 26 18:43:0 00:00:13
14   menu                   Y    1          Y   *Aug 26 18:43:1 00:00:09
```

Sample Output for show call application history session-level session-id Command

In the following example, the **show call application history session-level session-id** command for session 14 shows the event log for this instance. The event log shows the name and location of the audio prompt for which the error occurred. You can next review statistics for your application interfaces.

```
Router# show call application history session-level session-id 14

Session Info:
Session id:          14
Session name:
Application name:    menu
Application URL:     tftp://sample/menu.vxml
Start time:          *Aug 26 18:43:09 PST
Stop time:           *Aug 26 18:43:19 PST

Statistics:
 Subscriber Service - Call
                                           PSTN               VOIP
                                    Incoming Outgoing  Incoming Outgoing
```

```
Legs setup:                            1        0         0        0
Total legs connected:                  1        0         0        0
Legs handed in:                        0        0         0        0
Legs handed in returned back:          0        0         0        0
Legs handed out:                       0        0         0        0
Legs handed out came back:             0        0         0        0
Legs disconnected normally:            1        0         0        0
Legs disconnected for user error:      0        0         0        0
Legs disconnected for system error:    0        0         0        0

 Subscriber Service - Media
                                       Play       Record     TTS
Media attempts:                        2          0          0
Media successes:                       1          0          0
Media aborts:                          0          0          0
Media failures:                        1          0          0
Total media duration (in seconds):     9          0          0

 Subscriber Interaction - DTMF
DTMFs not matched:                     0
DTMFs matched:                         1
DTMFs no input:                        0
DTMFs long pound:                      0

Event log:
buf_size=4K, log_lvl=INFO
<ctx_id>:<timestamp>:<seq_no>:<severity>:<msg_body>
14:1061952189:450:INFO: Session started for App-type = menu, URL = tftp://sample/menu.vxml
14:1061952189:451:INFO: Incoming Telephony call received, LegID = 35
14:1061952189:452:INFO: LegID = 35: Calling = 4089023198, called = 52927, dial peer = 1
14:1061952189:453:INFO: LegID = 35: Leg State = LEG_INCCONNECTED
14:1061952189:454:INFO: Playing prompt #1: tftp://sample/audio/menu.au
14:1061952198:458:INFO: Prompt playing finished successfully.
14:1061952199:459:INFO: DTMF digit matched pattern v0, user input = 2
14:1061952199:460:INFO: Playing prompt #1: tftp://demo/audio/spanish_menu.au
14:1061952199:463:ERR : Prompt play setup failure.
14:1061952199:464:INFO: Script received event = "error.badfetch"
14:1061952199:465:INFO: LegID = 35: Call disconnected, cause = normal call clearing (16)
14:1061952199:468:INFO: Session done, terminating cause =
```

Sample Output for show call application interface summary Command

In the following example, the s**how call application interface summary** command shows statistics for each of your interface types including TFTP where the error occurred. You can next display the event log for the specific TFTP server.

```
Router# show call application interface summary

Aggregated statistics for tts service:
Stats last reset time *Aug 26 18:41:01 PST
Read requests:                0
Read successes:               0
Read failures:                0
Read aborts:                  0

Aggregated statistics for asr service:
Stats last reset time *Aug 26 18:41:01 PST
Read requests:                0
Read successes:               0
Read failures:                0
Read aborts:                  0

Aggregated statistics for tftp service:
Stats last reset time *Aug 26 18:41:01 PST
```

```
Read requests:                  2
Read successes:                 5
Read failures:                  1
Read aborts:                    0
Total bytes read:               255047
```

Sample Output for show call application interface Command

In the following example, the **show call application interface** command with the **tftp** keyword shows the event log for the TFTP server named demo. The event log shows that the gateway could not play the audio prompt because it could not find the TFTP server.

```
Router# show call application interface tftp

Server name:         sample

Statistics:
Last reset time *Aug 26 18:41:01 PST
Read requests:                  2
Read successes:                 2
Read failures:                  0
Read aborts:                    0
Total bytes read:               255047

Event log:
Last reset time *Aug 26 18:05:13 PST
buf_size=4K, log_lvl=INFO
<ctx_id>:<timestamp>:<seq_no>:<severity>:<msg_body>
sample:1061949913:173:INFO: ID = 653088C8: Read requested for URL =
tftp://sample/audio/menu.au
sample:1061949919:176:INFO: ID = 653088C8: Streamed read transaction Successful URL =
tftp://sample/audio/menu.au
sample:1061952156:416:INFO: ID = 651E5D6C: Read requested for URL =
tftp://sample/audio/welcome_test.au
sample:1061952166:423:INFO: ID = 651E5D6C: Streamed read transaction Successful URL =
tftp://sample/audio/welcome_test.au
--------------------------------------------------------------------------------
Server name:         demo

Statistics:
Last reset time *Aug 26 18:41:01 PST
Read requests:                  1
Read successes:                 0
Read failures:                  1
Read aborts:                    0
Total bytes read:               0

Event log:
Last reset time *Aug 26 18:38:05 PST
buf_size=4K, log_lvl=INFO
<ctx_id>:<timestamp>:<seq_no>:<severity>:<msg_body>
demo:1061952199:461:INFO: ID = 6542D968: Read requested for URL =
tftp://demo/audio/spanish_menu.au
demo:1061952199:462:ERR : ID = 6542D968: Read transaction failed URL =
tftp://demo/audio/spanish_menu.au, reason = IFS error 65540=Invalid IP address or hostname
--------------------------------------------------------------------------------
```

Monitoring Voice Call Legs

Perform this task to do basic monitoring of voice call legs. This can help you diagnose problems that occur during call setup before the call reaches the application.

Note Statistics for call legs are enabled by default. If you have previously enabled call leg event logs by following the steps in the "Enabling Event Logging and Statistics Globally for Voice Applications" section on page 10, you can skip directly to Step 5 of this task.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call leg event-log**
4. **exit**
5. **show call leg** {**active** | **history**} [**summary** | [**last** *number* | **leg-id** *leg-id*] [**event-log** | **info**]]

DETAILED STEPS

	Command or Action	Purpose

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

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

Step 3	`call leg event-log`
	Example:
	`Router(config)# call leg event-log`
		Enables transaction event logging for voice, fax, and modem call legs.

Step 4	`exit`
	Example:
	`Router# exit`
		Exits the current mode.

Step 5	**show call leg** {**active** | **history**} [**summary** | [**last** *number* | **leg-id** *leg-id*] [**event-log** | **info**]] **Example:** `Router# show call leg history last 2`	Displays call leg event logs or statistics.

What To Do Next

- To clear accumulated statistics and event logs for applications and application interfaces, see the "Clearing Event Logs and Statistics for Application Instances and Interfaces" section on page 22.
- To monitor events for application instances or call legs as they are occur, see the "Displaying Event Logs for Applications or Call Legs in Real-Time" section on page 22.

- To modify the default settings for event logs, see the following sections:
  - Modifying Event Log Settings for Application Instances, page 23
  - Modifying Event Log Settings for Application Interfaces, page 25
  - Modifying Event Log Settings for Call Legs, page 26
  - Modifying Event Log History Limits, page 27

Clearing Event Logs and Statistics for Application Instances and Interfaces

Perform this task to reset statistic counters to zero.

SUMMARY STEPS

1. **enable**
2. **clear call application** [**app-tag** *application-name*] **stats**
3. **clear call application interface** [[{**aaa** | **asr** | **flash** | **http** | **ram** | **rtsp** | **smtp** | **tftp** | **tts**} [**server** *server*]] [**event-log** | **stats**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear call application** [**app-tag** *application-name*] **stats** **Example:** `Router# clear call application app-tag sample_app stats`	Clears application-level statistics in the history table and subtracts the statistics from the gateway level, for all applications or a specific application.
Step 3	**clear call application interface** [[{**aaa** | **asr** | **flash** | **http** | **ram** | **rtsp** | **smtp** | **tftp** | **tts**} [**server** *server*]] [**event-log** | **stats**]] **Example:** `Router# clear call application interface http event-log`	Clears application interface statistics or event logs for all interfaces or a specific interface type or server.

Displaying Event Logs for Applications or Call Legs in Real-Time

Perform this task to display event logs for active calls as the events occur. If you are debugging or testing a script in a production-level network, the high call volume can make it difficult to select and view a specific event log while a call is still active. This task enables dynamic logging so that you can view events as they happen for active application instances or call legs. The output continues until the call terminates or you stop the display by using the **stop** keyword.

SUMMARY STEPS

1. **enable**
2. **monitor call application event-log** {**app-tag** *application-name* {**last** | **next**} | **session-id** *session-id* [**stop**] | **stop**}
3. **monitor call leg event-log** {**leg-id** *leg-id* [**stop**] | **next** | **stop**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**monitor call application event-log** {**app-tag** *application-name* {**last** | **next**} | **session-id** session-*id* [**stop**] | **stop**} **Example:** `Router# monitor call application event-log session-id 5`	Displays events for the most recently active call or the next new call for a specific application, or for a specific instance.
Step 3	**monitor call leg event-log** {**leg-id** *leg-id* [**stop**] | **next** | **stop**} **Example:** `Router# monitor call leg event-log leg-id 5`	Displays events for next active call leg, or specific call leg.

Modifying Event Log Settings for Application Instances

Perform this task to modify the default settings for the event logs that are generated for voice application instances.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **monitor**
5. **event-log dump ftp** *server*[**:***port*]**/***file* **username** *username* **password** [*encryption-type*] *password*
6. **event-log max-buffer-size** *kbytes*
7. **event-log error-only**
8. **exit**
9. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**application** **Example:** `Router(config)#application`	Enters application configuration mode.
Step 4	monitor **Example:** `Router(config-app)# monitor`	Enters application configuration monitor mode.
Step 5	**event-log dump ftp** *server*[**:***port*]**/***file* **username** *username* **password** [*encryption-type*] *password* **Example:** `Router(config-app-monitor)# event-log dump ftp ftp-server/elogs/app_elogs.log username myname password 0 mypass`	(Optional) Specifies the location of the external file to which the gateway writes the contents of the event log buffer.
Step 6	**event-log max-buffer-size** *kbytes* **Example:** `Router(config-app-monitor)# event-log max-buffer-size 8`	(Optional) Sets the maximum size of the event log buffer for each application instance.
Step 7	**event-log error-only** **Example:** `Router(config-app-monitor)# event-log error-only`	(Optional) Logs only error events for application instances.
Step 8	**exit** **Example:** `Router(config-app-monitor)# exit`	Exits the application configuration monitor mode.
Step 9	**exit** **Example:** `Router(config-app)# exit`	Exits the application configuration mode.

Modifying Event Log Settings for Application Interfaces

Perform this task to modify the default settings for event logs generated for types of server interfaces that communicate with voice applications.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **monitor**
5. **interface event-log dump ftp** *server*[**:***port*]**/***file* **username** *username* **password** [*encryption-type*] *password*
6. **interface event-log max-buffer-size** *kbytes*
7. **interface max-server-records** *number*
8. **interface event-log error-only**
9. **exit**
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**`enable`** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**`configure terminal`** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**`application`** **Example:** `Router(config)#application`	Enters application configuration mode.

Step 4	`monitor`
	
	Example:
	`Router(config-app)# monitor`
		Enters application configuration monitor mode.

	Command or Action	Purpose
Step 5	**event-log dump ftp** *server*[**:***port*]**/***file* **username** *username* **password** [*encryption-type*] *password* **Example:** `Router(config-app-monitor)# interface event-log dump ftp ftp-server/elogs/int_elogs.log username myname password 0 mypass`	(Optional) Specifies the location of the external file to which the gateway writes the contents of the interface event log buffer.
Step 6	**interface event-log max-buffer-size** *kbytes* **Example:** `Router(config-app-monitor)# interface event-log max-buffer-size 50`	(Optional) Sets the maximum size of the event log buffer for each application interface.
Step 7	**interface max-server-records** *number* **Example:** `Router(config-app-monitor)# interface max-server-records 50`	(Optional) Sets the maximum number of interface event-log records that are saved.
Step 8	**interface event-log error-only** **Example:** `Router(config-app-monitor)# interface event-log error-only`	(Optional) Limits interface event logging to error events only.
Step 9	**exit** **Example:** `Router(config-app-monitor)# exit`	Exits the application configuration monitor mode.

Modifying Event Log Settings for Call Legs

Perform this task to modify the default settings for event logs generated for voice call legs.

Restrictions

Event logs are available for telephony call legs only. Event logs for IP call legs are not supported.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **call leg event-log dump ftp** *server*[**:***port*]**/***file* **username** *username* **password** [*encryption-type*] *password*
4. **call leg event-log max-buffer-size** *kbytes*
5. **call leg event-log error-only**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**call leg event-log dump ftp** *server*[**:***port*]**/***file* **username** *username* **password** [*encryption-type*] *password* **Example:** `Router(config)# call leg event-log dump ftp ftp-server/elogs/leg_elogs.log username myname password 0 mypass`	(Optional) Specifies the location of the external file to which the voice gateway writes the contents of the event log buffer.
Step 4	**call leg event-log max-buffer-size** *kbytes* **Example:** `Router(config)# call leg event-log max-buffer-size 50`	(Optional) Sets the maximum size of the event log buffer for each call leg.
Step 5	**call leg event-log error-only** **Example:** `Router(config)# call leg event-log error-only`	(Optional) Enables transaction event logging for error events only.
Step 6	**exit** **Example:** `Router# exit`	Exits the current mode.

Modifying Event Log History Limits

Perform this task to modify the default settings for saving event logs to history.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **monitor**
5. **history session event-log save-exception-only**
6. **history session max-records** *number*

7. **history session retain-timer** *minutes*
8. **exit**
9. **exit**
10. **call leg history event-log save-exception-only**
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**application** **Example:** `Router(config)#application`	Enters application configuration mode.

Step 4	`monitor`
	
	Example:
	`Router(config-app)# monitor`
		Enters application configuration monitor mode.

Step 5	**history session event-log save-exception-only** **Example:** `Router(config-app-monitor)# history session event-log save-exception-only`	(Optional) Saves event logs to history only for application sessions with exceptions or errors.
Step 6	**history session max-records** *number* **Example:** `Router(config-app-monitor)# history session max-records 50`	(Optional) Sets the maximum number of session records that are saved in the event-log history table.
Step 7	**history session retain-timer** *minutes* **Example:** `Router(config-app-monitor)# history session retain-timer 30`	(Optional) Sets the maximum number of minutes for which session history records are saved.
Step 8	**exit** **Example:** `Router(config-app-monitor)# exit`	Exits the application configuration monitor mode.

	Command or Action	Purpose
Step 9	**exit** **Example:** `Router(config-app)# exit`	Exits the application configuration mode.
Step 10	**call leg history event-log save-exception-only** **Example:** `Router# call leg history event-log save-exception-only`	(Optional) Saves event logs only for call legs that have exceptions or errors.
Step 11	**exit** **Example:** `Router# exit`	Exits the current mode.

Configuration Examples for Monitoring Voice Applications

This section includes the following examples:

- Enabling Event Logs and Statistics Globally: Example, page 29
- Customizing Event Logs and Statistics Example, page 31

Enabling Event Logs and Statistics Globally: Example

```
!
version 12.3
service timestamps debug datetime msec
service timestamps log uptime
no service password-encryption
!
hostname Router
!
!
resource-pool disable
tdm clock priority 1 1/0
spe default-firmware spe-firmware-1
aaa new-model
!
!
ip domain name domain.com
ip host speech-asr 10.10.10.111
ip host ftp-server 10.10.10.119
!
isdn switch-type primary-5ess
!
!
voice service voip
 fax protocol t38 ls-redundancy 7 hs-redundancy 3 fallback none
!
ivr asr-server rtsp://speech-asr/recognizer
ivr tts-server rtsp://speech-asr/synthesizer
!
fax receive called-subscriber $d$
```

```
fax send transmitting-subscriber 5550122
fax send left-header Date: $a$
fax send center-header $d$
fax send right-header $s$
fax send coverpage email-controllable
fax send coverpage comment Cover Page comment
fax interface-type fax-mail
mta send server 10.10.10.112 port 25
mta send subject sample subject
mta send postmaster postmaster@domain.com
mta send mail-from hostname Router.domain.com
mta send mail-from username user1
mta receive aliases [10.10.10.100]
mta receive maximum-recipients 1000
dial-control-mib retain-timer 10
dial-control-mib max-size 2
!
!
controller T1 1/0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
interface FastEthernet0/0
 ip address 10.10.10.100 255.255.0.0
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
 no cdp enable
!
interface FastEthernet0/1
 ip address 11.11.11.100 255.255.0.0
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
!
!
interface Serial1/0:23
 no ip address
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 no cdp enable
!
interface Group-Async0
 no ip address
 no ip route-cache
 no ip mroute-cache
 no peer default ip address
 group-range 3/00 3/107
!
!
interface Dialer1
 no ip address
 no ip route-cache
 no ip mroute-cache
!
ip default-gateway 10.10.10.1
ip classless
ip route 10.0.0.0 255.0.0.0 10.10.10.1
ip route 11.0.0.0 255.0.0.0 11.11.11.1
no ip http server
!
```

```
snmp-server community password RW
snmp-server enable traps tty
!
call leg event-log
!
application
    service onramp tftp://demo/router/TCLware.2.0.1/app_libretto_onramp9.2.0.0.tcl
    !
    service offramp tftp://demo/router/TCLware.2.0.1/app_faxmail_offramp.2.0.1.1.tcl
    !
    service generic tftp://demo/scripts/master/generic.vxml
    !
    monitor
    interface stats
    interface event-log
    stats
    event-log
!
voice-port 1/0:D
!
dial-peer cor custom
!
dial-peer voice 1 pots
 service generic
 incoming called-number .
 direct-inward-dial
!
dial-peer voice 2 mmoip
 service fax_on_vfc_onramp_app out-bound
 destination-pattern .
 information-type fax
 session target mailto:$e$@[10.10.10.112]
!
line con 0
 exec-timeout 0 0
 logging synchronous
line aux 0
 logging synchronous
line vty 5 105
line 2/00 3/107
 no flush-at-activation
 modem InOut
!
scheduler allocate 10000 400
!
end
```

Customizing Event Logs and Statistics Example

```
.
.
.
call leg event-log errors-only
call leg event-log max-buffer-size 10
call leg event-log dump ftp ftp-server/leg_elogs.log username myname password mypass
call leg event-log
!
application
    service onramp tftp://demo/router/TCLware.2.0.1/app_libretto_onramp9.2.0.0.tcl
    !
    service offramp tftp://demo/router/TCLware.2.0.1/app_faxmail_offramp.2.0.1.1.tcl
    !
```

```
    service generic tftp://demo/scripts/master/generic.vxml
    !
    monitor
    interface stats
    interface event-log
    interface event-log error-only
    interface event-log max-buffer-size 50
    interface event-log dump ftp ftp-server/int_elogs.log username myname password mypass
    interface event-log ram disable
    interface max-server-records 20
    stats
    event-log
    event-log error-only
    event-log max-buffer-size 8
    event-log dump ftp ftp-server/app_elogs.log username myname password mypass
!
```

Additional References

- “” on page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco release, Tcl IVR and VoiceXML features for that release
- “Overview of Cisco IOS Tcl IVR and VoiceXML Applications” on page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance

MGCP Scripting Support for Cisco IOS VoiceXML

Cisco IOS VoiceXML supports Cisco's Media Gateway Control Protocol (MGCP) scripting feature to run VoiceXML documents. The MGCP scripting support for VoiceXML is based on existing support for Tcl. An MGCP call agent (CA) can signal the gateway to run a VoiceXML document on a channel. An argument string can be passed from the call agent to the running document, and the document can return an argument string to the call agent on completion.

For more information on MGCP, refer to the *Cisco IOS MGCP and Related Protocols Configuration Guide*, Release 12.3.

The following sections describe specific details related to using MGCP with VoiceXML.

Syntax

- Because there is no VoiceXML symbol currently defined in the MGCP scripting specification, the MGCP signal uses Tcl in the syntax. The filename of the VoiceXML document must contain the vxml extension. The syntax for the MGCP signal is:

```
S:script/tcl(uri.vxml,arguments)
```

- The string returned in the <exit> tag of the document is returned in the report on completion to the call agent. The namelist is reported in the form <name>=<value>. The return status is reported as:

```
"return-status=<success or failure>".
```

  The syntax is:

```
O:script/oc(return-status={success or failure}, <name>=<value>, ...)
```

- When the MGCP call agent sends a command to the Cisco gateway telling it to run a VoiceXML document, the response can be either success (OC) or failure (OF). The VoiceXML code includes a return status code in the OC response.
- The argument string from the call agent to the VoiceXML application can be up to 150 characters.

Call Legs

VoiceXML documents can only handle one call leg. If the call agent tries to run a VoiceXML document on a connection, or on a call leg connected to another call leg, the attempt fails.

Note The Cisco MGCP feature creates a virtual, or place-holder, second call leg with the CRCX signal. The CRCX signal cannot be used by the call agent to run a VoiceXML document because the VoiceXML code fails with two call legs. The RQNT signal should be used instead.

Transfer

VoiceXML documents are not expected to run the <transfer> tag, because a call agent application normally initiates all call signaling. The VoiceXML document, however, is not prevented from running the transfer. If there are ports on the gateway that accept local signaling, the transfer can place a call.

Caching

VXML documents loaded from MGCP are cached; caching is not checked on every call.

The default cache check time is one minute; the check time can be configured with the **cache reload time** command in the application global configuration submode. The cache check is on top of the HTTP protocol, which is fixed for fast rather than safe operation. Thus, the **http client cache refresh** command also affects MGCP document cache reloading if the load uses HTTP. TFTP uses the **cache reload time** command.

If the document is not used for 30 minutes, it is deleted from memory. This TTL is not configurable.

After the VoiceXML document has been loaded and run, use the **show call application voice** command (using the summary keyword) to display it. Normally, the syntax of the document name is *filename*.vxml, but if an existing application from a different URL uses that name, it will be given a suffix of _1.vxml.

You can preload the VoiceXML document into RAM using the application configuration commands. The name must be *filename*.vxml, and the URL must be identical to the URL in the S:command from the call agent for the MGCP application to use the loaded application. This permits the configuration of the language for the application.

Example of MGCP Scripting for VoiceXML

This section contains an example of a call agent running a simple VoiceXML document on a channel, including:

- Debug output on the gateway when the document runs
- Notification request from the call agent to the gateway
- Notification from the gateway back to the call agent
- A **show** command to look at the document

The following steps outline the process when the VoiceXML document is run:

1. The VoiceXML document is debug.vxml:

```
<vxml version="1.0" base="http://sunhost1/vxml/">
       <!-- Document to just output debug -->
       <!-- version 2 -->
       <var name="exitstring" expr="'ran this document'"/>
       <form id="dir_search" scope="document">
              <block>
              <cisco-puts>debug.vxml is running this document\n</cisco-puts>
              <cisco-puts>handoff_string=<cisco-putvar
                namelist="session.handoff_string"/>\n</cisco-puts>
              <exit namelist="exitstring"/>
             </block>
       </form>
       </vxml>
```

Note <cisco-puts> and <cisco-putvar> are Cisco private elements. For more information, see the *Cisco VoiceXML Programmer's Guide*.

2. The call agent sends this message to the gateway to run the document debug.vxml on channel 3 of the second T1:

```
RQNT 32052 ds1-2/3 MGCP 0.1
X:234
R:script/oc, script/of
S:script/tcl(http://sunhost1/sample/debug.vxml,arg1,arg2,arg3)
```

3. On the gateway, if you enter the **debug vxml puts** command, it displays:

```
Oct 18 09:49:47.061:debug.vxml is running this document
Oct 18 09:49:47.061:handoff_string=arg1,arg2,arg3
```

4. The MGCP notify message returned to the call agent is:

```
NTFY 12 ds1-2/3@sblab160.cisco.com MGCP 0.1
X:234
O:SCRIPT/oc(return-status=success, exitstring=ran this document)
```

5. Use the **show call application voice** command to view the entire document in memory:

```
Router# show call application voice debug.vxml

VoiceXML Application debug.vxml
    URL=http://sunhost1/sample/debug.vxml
    No languages configured
    It has 0 calls active.
    Interpreted by Voice Browser Version 1.0 for VoiceXML 1.0.
```

Symbols

.au files
- codec mappings 123
- data-length correction utility 124
- format support 121

.wav files
- codec mappings 123
- data-length correction utility 124
- format support 121

A

ASR
- configuration example 169
- configuring MRCP records 168
- description 154
- server requirements 154
- specifying server location 156, 160
- troubleshooting 157
- verifying configuration 158, 163
- verifying functionality 158

audio files
- codec complexity 112
- codec mappings 123
- codec restrictions 101
- configuring
  - dynamic prompt language and location 107
  - streaming 111
- data-length correction utility 118
- playout
  - memory-based 102
  - methods 102
  - playing prompts from gateway 102
  - restrictions 100
  - RTSP streamed 103
  - TTS streamed 103
- recording
  - codec support 122
  - destinations 119
  - memory limits 109
  - restrictions 100
  - scenario 120
- troubleshooting 110
- volume and rate controls 104

C

call admission control 7

call handling between TCL IVR and VoiceXML 23

call-transfer
- configuring for voice applications 213
- configuring outbound dial peer 70
- description 23

codecs
- codec complexity 112
- configuring in dial peer 72
- file header mappings 123
- recording support 122
- restrictions 100

configuration examples
- ASR and TTS 169
- fax detection for VoiceXML 180
- SIP and TEL URL support 262
- TBCT 224
- TCL IVR 2.0 85

TCL IVR session interaction 237
VoiceXML inbound application 91
VoiceXML outbound application 94
VoiceXML voice store and forward 144
configuring
ASR and TTS media server locations 156, 160
audio prompt streaming 111
call-transfer method 213
codec complexity 112
DNIS maps 75
dynamic prompts language and location 107
fax detection for VoiceXML 177
HTTP client settings 80
inbound applications 54
inbound dial peer to match URI 250
legless TCL IVR applications 232
loading mail application 136
loading voice application onto gateway 39, 43
memory recording limits 109
MRCP client history records 168
off-ramp gateway for voice store and forward
description 133
MMoIP dial peer 141
POTS dial peer 139
on-ramp gateway for voice store and forward
description 125
POTS dial peer 128
outbound applications 67
outbound dial peer for call transfers 70
outbound dial peer to match URI 255
RTPvt 217
TBCT
call limits 220
enabling for trunk groups 217
outbound dial peer 219
terminating billing 221
voice class for SIP URLs 248
voice class for TEL URLs 249
VoiceXML voice store and forward 117

D

data-length correction utility for audio files 118
dial peers
MMoIP
configuring for off-ramp gateway 141
description 56
overview 55
POTS
configuring for off-ramp gateway 139
configuring for on-ramp gateway 128
description 55
verifying configuration 59
VoIP
configuring for call transfers 70
description 55
DID, description 57
DNIS maps
configuring 75
description 75
limits 76
URLs 76
DSN
description 135
dynamic prompts
configuring
language and location 107
description 103

E

Examples 237

F

fax detection for VoiceXML
configuration example 180
configuring 177
description 176

verifying configuration 178
fax interface-type command 126, 138

G

GTD parameters 211

H

HTTP client
caching refresh value 26
HTTP 1.1 headers 26
modifying default settings 80
supported features 25
verifying settings 84
HTTP server
enabling chunked transfer 9
installing Apache PHP server 8
requirements 8

I

Inbound 251
inbound applications 54
information tags 188
Interface 138
IVR applications
configuration examples 85
configuration tasks 39
overview 20

L

language for dynamic prompts
configuring 107
language modules description 104

M

mail application
call scenario 134
downloading TCL off-ramp mail script 136
loading onto gateway 136
off-ramp description 133
MDN
configuring in dial peer 128
description 135
media inactive call detection 185
media servers
configuring location 156, 160
MRCP session records, setting limits 168
requirements 154
troubleshooting 157
verifying location 158, 163
memory
recording description 102
recording limits 109
memory requirements 6
message URL http
//www.cisco.com/en/US/products/ps6441/prod_configuration_guide09186a0080565f8a.html 151
MGCP scripting
description 23
VoiceXML support 305
MMoIP dial peers
description 56
off-ramp configuration 141
MRCP client
configuring client history 168
description 154
MTA
configuring
receiving MTA 138
sending MTA 126

O

off-ramp gateway
- configuring
  - downloading mail trigger script 136
  - interface type for voice mail 138
  - MMoIP dial peer 141
  - overview 133
  - POTS dial peer 139
  - receiving MTA 138
- mail trigger scenario 134
- verifying configuration 142

on-ramp gateway
- configuring
  - overview 125
  - POTS dial peer 128
- verifying configuration 129

outbound applications
- call scenario 67
- configuring 67
- description 67
- verifying configuration 73

P

POTS dial peers
- description 55
- inbound application configuration 54
- off-ramp configuration 139
- on-ramp configuration for recording 128

prerequisites 6

R

recognition servers
- configuring location 156, 160
- requirements 154

recording
- restrictions 100
- setting memory limits 109
- verifying configuration 132

Release-to-Pivot
- see RTPvt 211

restrictions 17, 100

RTPvt
- configuring 209
- description 211
- enabling in a voice application 217

RTSP
- streaming audio files 103
- troubleshooting 160

S

service registry 230

SIP URL
- creating a voice class 248
- matching on an inbound call 252

speech recognition
- configuring server location 156, 160
- description 154
- server requirements 154

speech synthesis
- configuring server location 156, 160
- description 154
- server requirements 154

streaming audio files
- configuring 111
- using RTSP 103

T

TBCT
- configuring 209
- configuring call limits 220
- configuring outbound dial peers 219
- description 212

enabling for multiple PRIs 217
terminating billing 221
TCL IVR applications
audio prompt playout 102
audio recordings 100
configuration examples 85
configuration tasks 39
configuring
inbound applications 54
language and location of dynamic prompts 107
outbound applications 67
dial peer overview 55
interaction with VoiceXML applications 23
language modules 104
loading application onto gateway 39, 43
matching called numbers 56
MGCP scripting overview 23
overview 22
restrictions 17
service registry 230
session interaction 229, 230
SIP and TEL URL support 246
starting a legless instance 231
TEL URL
creating a voice class 249
matching on an inbound call 251
transfering calls 70
TTS
configuration example 169
configuring MRCP records 168
description 154
server requirements 154
specifying server location 156, 160
troubleshooting 157
verifying configuration 158, 163
verifying functionality 158
Two B-Channel Transfer
see TBCT 212

U

URI voice class, creating 247
URL support for voice applications 245, 246
configuring inbound dial peer 250
configuring outbound dial peer 255

V

VCWare version 7
verifying gateway configuration 60
voice store and forward
configuration examples 144
configuring
off-ramp gateway 133
off-ramp description 120
on-ramp description 120
on-ramp gateway
configuring 125
overview 119
recording and playback media locations 119
VoiceXML applications
audio files
data-length correction utility 118
prompt playout 102
recording restrictions 100
call admission control 7
call scenario examples 21
configuration examples 91
configuration tasks 39
configuring
call transfer 70
DNIS maps 75
fax detection 177
inbound applications 54
language and location of dynamic prompts 107
on-ramp gateway
POTS dial peer 128
sending MTA 126

outbound applications 67
dial peer overview 55
DNIS maps
configuring 75
description 75
document development 16
gateway software requirements 6
HTTP server requirements 8
interaction with TCL IVR applications 23
language modules 104
loading applications onto gateway 39, 43
matching called numbers 56
media server requirements 154
memory requirements 6
MGCP scripting overview 23
overview 20
prerequisites
gateway software 6
VCWare version 7
restictions 17
security 22
SIP and TEL URL support 246
TCL IVR 2.0 overview 22
verifying configuration 60, 178
volume and rate controls 104
VoiceXML document loops 22
VoIP dial peers
configuring for call transfers 70
configuring for outbound applications 67
description 55
volume and rate controls 104