
# 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



## 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

- 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

# 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

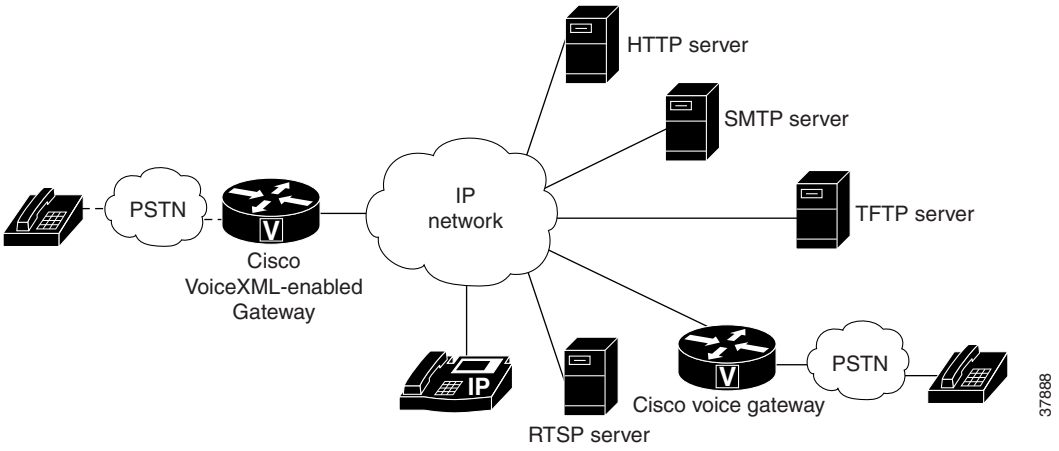
Applications written in Voice eXtensible Markup Language (VoiceXML) provide access through a voice browser to content and services over the telephone, just as Hypertext Markup Language (HTML) provides access through a web browser running on a PC. The universal accessibility of the telephone and its ease 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.

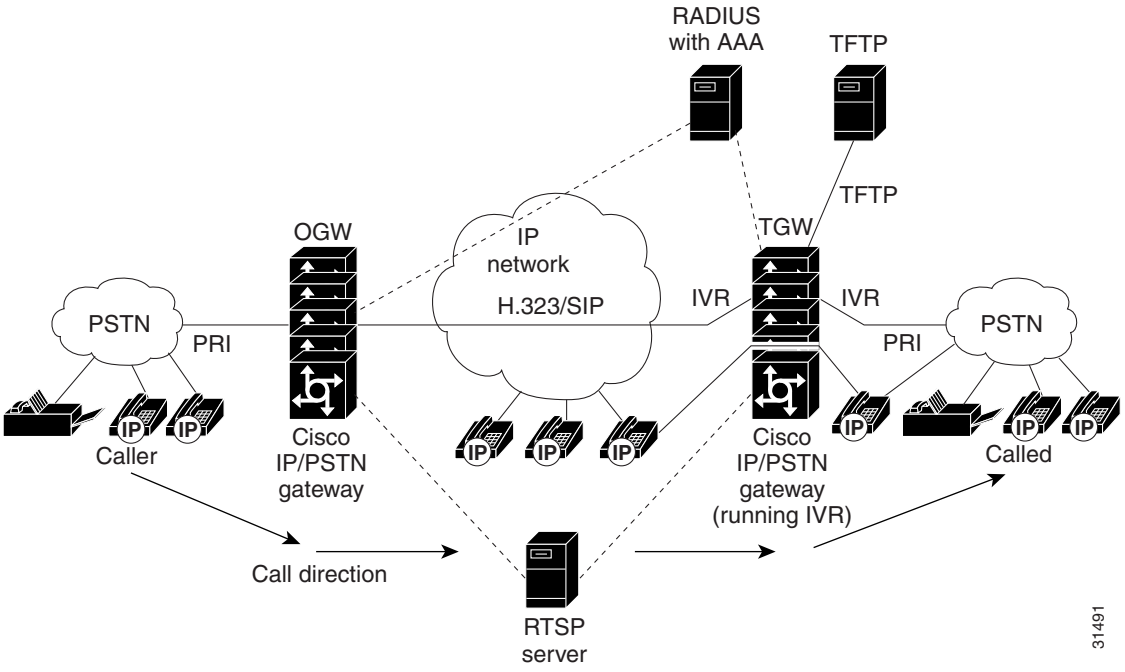
The loop count includes both before <disconnect> and after <disconnect> events. After <disconnect>, the VoiceXML document is mostly unrestricted if there is no user interaction.

# 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***



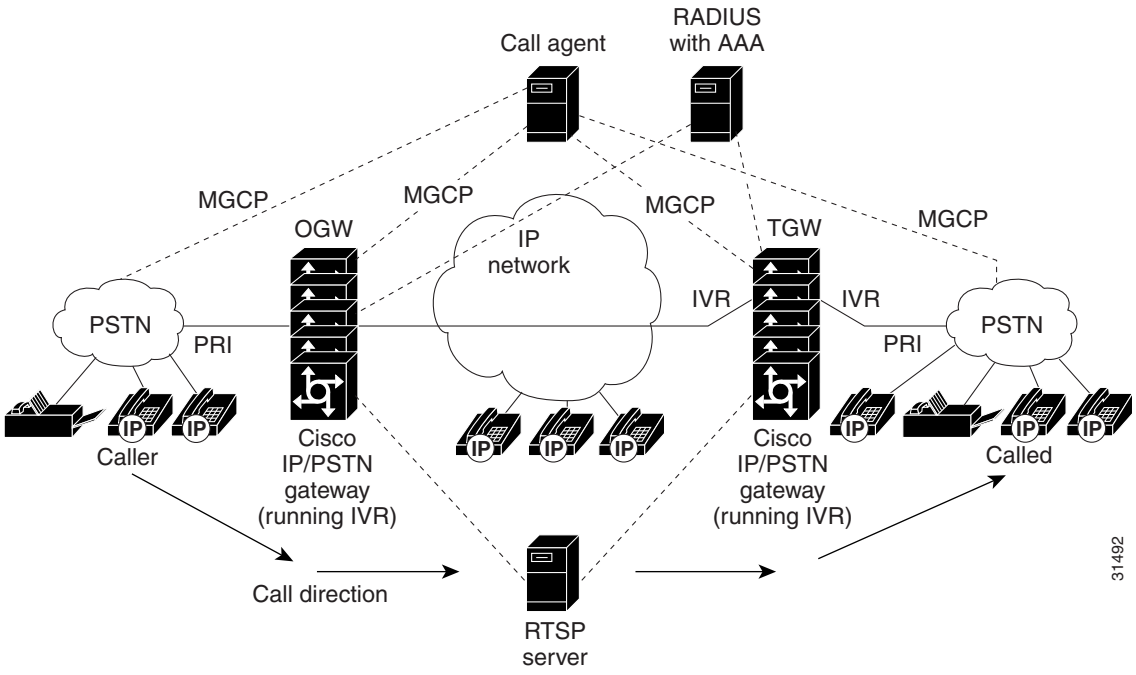

For information on developing Tcl scripts for voice applications, see the *Tcl IVR API Version 2.0 Programmer's Guide.*

# 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** | Global configuration |
| | **service** | Application configuration |
| | **package** | Application configuration |
| | **param** | 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**<br>or<br>**paramspace appcommon security** | Application parameter configuration |
| call application voice set-location | **paramspace language location** | Application configuration |
| call application voice transfer mode | **paramspace callsetup mode**<br>or<br>**param mode** | Application configuration |
| call application voice transfer reroute-mode | **paramspace callsetup reroutemode**<br>or<br>**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

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

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

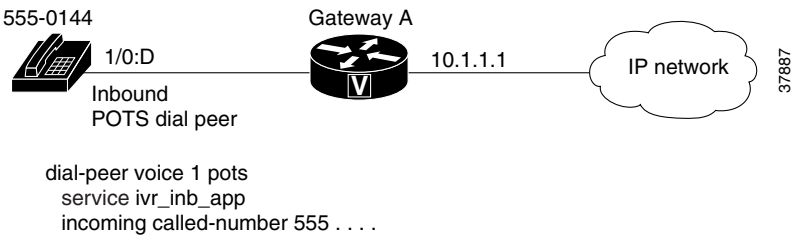
If dial peer parameter group configuration is not successful, verify the following:

- 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

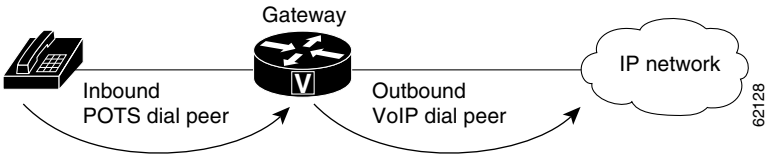
Dial peers are central to the configuration of Cisco voice gateways. They act as the focal point where associated elements come together, including:

- 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.

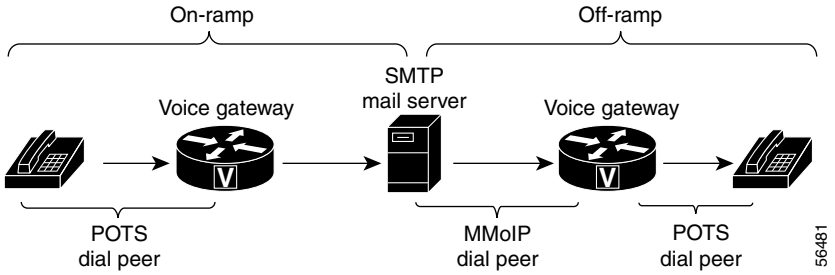
*Figure 3-5 Voice Dial Peers*




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.

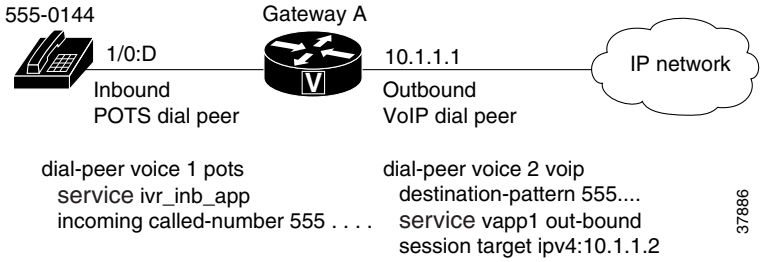
## Call Scenario for an Outbound Application

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:

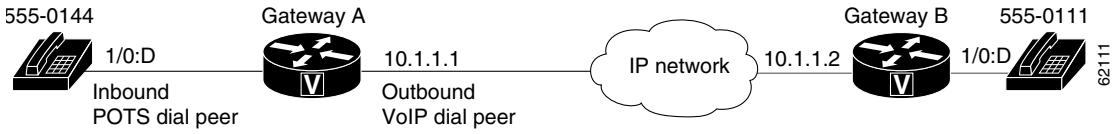
```
session protocol {cisco | sipv2}
```

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

**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)*

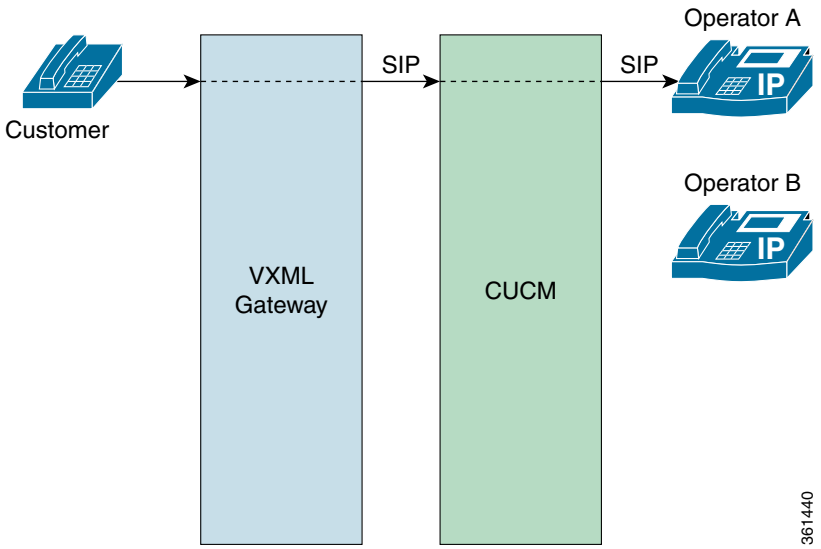
| Possible Causes | Suggested Actions |
|---|---|
| Gateway cannot find the specified application | Verify that the application is configured on the gateway. See the "Verifying Loading of Service" section on page 26. |
| Inbound dial peer does not have the correct application configured | Verify that 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.

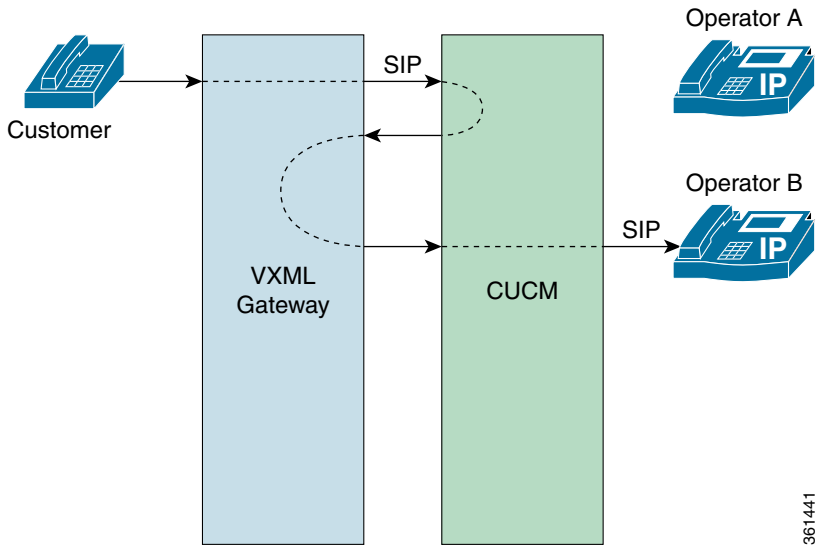
Call loop occurs when an operator transfers a call through the VoiceXML application to another operator, as shown in Figure 3-9.

*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 supplementary-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

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

```
!
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.