



## Configuring TCL IVR Applications

---

This chapter shows you how to configure Interactive Voice Response (IVR) using the Tool Command Language (TCL) scripts. This software release introduces TCL IVR Version 2.0 with several feature enhancements to the Cisco IVR functionality. This chapter contains the following sections:

- [TCL IVR Overview, page 457](#)
- [TCL IVR Enhancements, page 458](#)
- [TCL IVR Prerequisite Tasks, page 463](#)
- [TCL IVR Configuration Tasks List, page 464](#)
- [TCL IVR Configuration Examples, page 471](#)

For a complete description of the commands used in this chapter, refer to the *Cisco IOS Voice, Video, and Fax Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature in this chapter, use the [Feature Navigator](#) on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the “Identifying Supported Platforms” section in the “Using Cisco IOS Software” chapter.

## TCL IVR Overview

IVR consists of simple voice prompting and digit collection to gather caller information for authenticating the user and identifying the destination. IVR applications can be assigned to specific ports or invoked on the basis of DNIS. An IP public switched telephone network gateway can have several IVR applications to accommodate many different gateway services, and you can customize the IVR applications to present different interfaces to the various callers.

IVR systems provide information in the form of recorded messages over telephone lines in response to user input in the form of spoken words, or more commonly dual tone multifrequency (DTMF) signalling. For example, when a user makes a call with a debit card, an IVR application is used to prompt 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 and then places the call to the destination phone or system.

IVR uses TCL scripts gather information and to process accounting and billing. For example, a TCL IVR script plays when a caller receives a voice-prompt instruction to enter a specific type of information, such as a personal identification number (PIN). After playing the voice prompt, the TCL IVR application collects the predetermined number of touch tones and sends the collected information to an external server for user authentication and authorization.

# TCL IVR Enhancements

Since the introduction of the Cisco IVR technology, the software has undergone several enhancements. TCL IVR Version 2.0 is made up of separate components that are described individually in the sections that follow. The enhancements are as follows:

- Media Gateway Control Protocol (MGCP) scripting package implementation
- Real Time Streaming Protocol (RTSP) client implementation
- TCL IVR prompt playout and digit collection on IP call legs
- New TCL verbs to utilize RTSP and MGCP scripting features

The enhancements add scalability and enable the TCL IVR scripting functionality on VoIP legs. In addition, support for RTSP enables VoIP gateways to play messages from RTSP-compliant announcement servers. The addition of these enhancements also reduces the CPU load and saves memory on the gateway because no packetization is involved. Larger prompts can be played, and the use of an external audio server is allowed.

**Note**

---

TCL IVR 2.0 removed the signature locking mechanism requirement.

---

## MGCP Scripting

TCL IVR Version 2.0 infrastructure is greatly enhanced with the addition of support for MGCP using the application package model. MGCP defines application packages to run scripts on the media gateways. These application packages initiate scripts on the gateways and receive return values after execution completes. MGCP scripting allows external call agents (CAs) to instruct a media gateway to run an TCL IVR script in order to perform a specific task and return the end result. For example, you can request and collect the PIN and account number from a caller.

Two previously released Cisco VoIP features that can be implemented are the Debit Card for Packet Telephony and TCL IVR. Both features use the TCL scripting language. The TCL scripts that run with MGCP are written in TCL IVR API Version 2.0 and are able to receive calls through hand off. MGCP scripts can run any TCL command.

**Note**

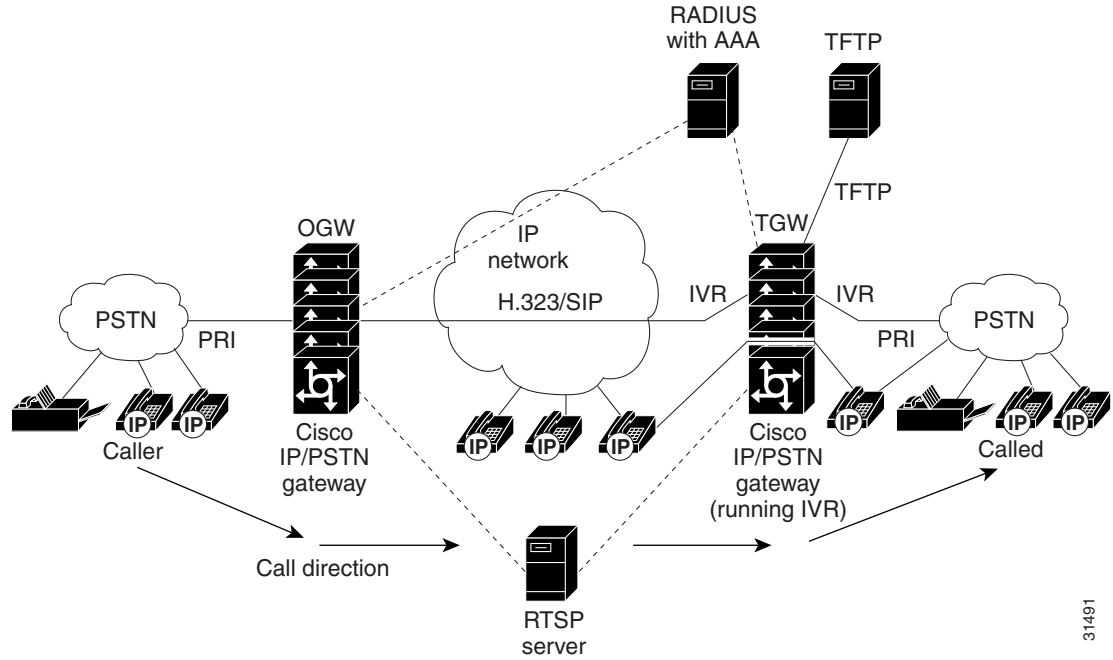
---

For more information about MGCP, see “Configuring Media Gateway Control Protocol and Related Protocols” chapter.

---

[Figure 91](#) displays the CA controlling the TCL IVR scripts. MGCP is the protocol that is running on the CA. The RTSP server is configured to interact with the gateways that have TCL IVR scripts installed and running. The RADIUS server running authentication, authorization, and accounting (AAA) also interacts with the gateways.

Figure 91 MGCP Control of TCL IVR Scripts



31491

## RTSP Client Implementation

RTSP is an application-level protocol used for control over the delivery of data that has real-time properties. Using RTSP also enables an external RTSP server to play announcements and interact with voice mail servers. It provides an extensive framework to enable control and to perform on-demand delivery of real-time data. For example, RTSP is used to control the delivery of audio streams from an audio server.

If you use an RTSP server in your network with VoIP gateways, a scripting application, (for example, an MGCP script) can run on the gateway and connect calls with audio streams from an external audio server. Using RTSP also has the following benefits:

- Reduces the CPU load
- Allows large prompts to be played that previously demanded high CPU usage from the gateway
- Saves memory on the gateway because no packetization is involved
- Allows use of an external audio server which removes the limitation on the number of prompts that can be played out and on the size of the prompt

## TCL IVR Prompts Played on IP Call Legs

TCL IVR Version 2.0 scripts can be configured for incoming plain old telephone service (POTS) or VoIP call legs to play announcements to the user or collect user input (digits). With TCL IVR Version 2.0 the prompts can be triggered from both the PSTN side of the call leg and the IP side of the call leg. This feature enables the audio files (or prompts) to be played out over the IP network.

TCL IVR scripts played toward a VoIP call leg are subject to the following conditions:

- G.711 mu-law encoding must be used when prompts are played.
- G.711 mu-law encoding must also be used for the duration of these calls, even after prompt ployout has completed.
- Digital signaling protocols (DSPs) can not be on the IP call leg so the script cannot initiate a tone.
- When an TCL IVR script is used to collect digits on a VoIP call leg, one of the following DTMF relay methods must be used.
  - For H.323 protocol configured on the call leg, use one of the following DTMF relay methods: Cisco proprietary RTP, H.245 Alphanumeric IE, or H.245 Signal IE
  - For SIP protocol configured on the call leg, use Cisco proprietary RTP

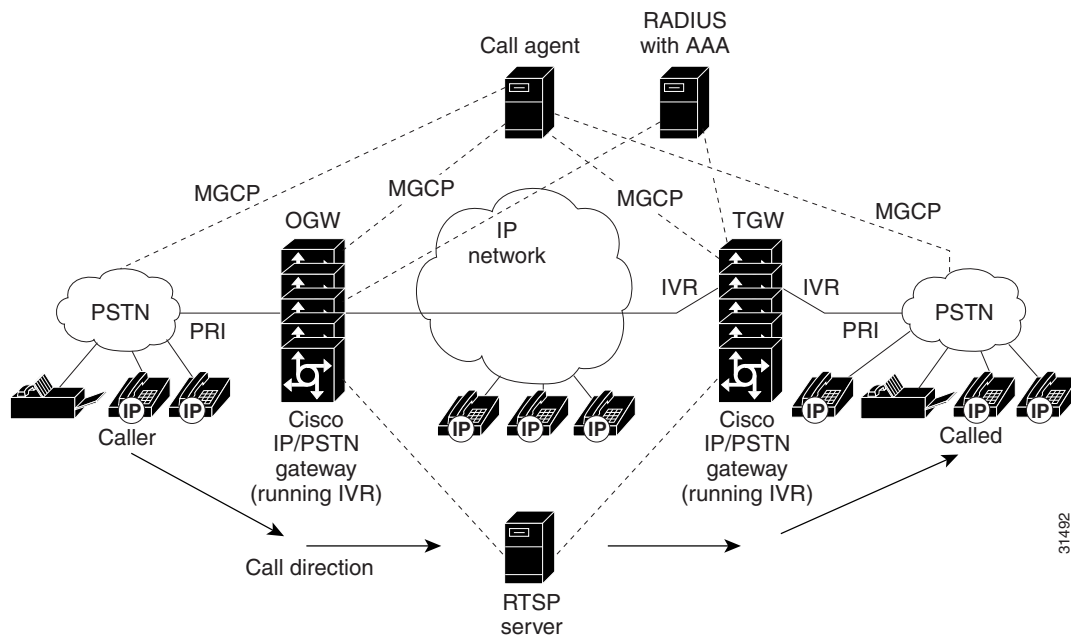


**Note**

For additional information about the **dtmf-relay** command, refer to the *Cisco IOS Voice, Video, and Fax Command Reference*.

IVR 2.0 enables the system to accept calls initiated from the IP side of the network using G.711, and terminate calls to the terminating gateway using the same codec. [Figure 92](#) displays the TCL IVR application on the gateways controlling the scripts. IP phones can also originate a call to a gateway running an TCL IVR script.

**Figure 92** IVR Control of Scripts on an IP Call Leg



## TCL Verbs

TCL IVR, Version 2.0, delivers a new set of TCL verbs and scripts that replace the previous TCL version. The new TCL verbs enable the user to:

- Utilize the RTSP audio servers
- Develop TCL scripts that interact with the IVR application
- Pass events to the Media Gateway Controller, which is a call agent

TCL IVR Version 2.0 is not backward compatible with the IVR 1.0 scripts. The MGCP scripting package can only be implemented using the new TCL verbs.

**Note**

For in-depth information about the TCL 2.0 verb set and how to develop scripts, refer to Cisco.com (Related Documentation index) and find the document, *TCL IVR API Version 2.0 Programmer's Guide*. The URL is:

[http://www.cisco.com/univercd/cc/td/doc/product/access/acs\\_serv/vapp\\_dev/index.htm](http://www.cisco.com/univercd/cc/td/doc/product/access/acs_serv/vapp_dev/index.htm).

The guide also contains an annotated example of a TCL IVR script and includes instructions for testing and loading TCL IVR scripts.

TCL IVR scripts use the TCL verbs to interact with the gateway during call processing in order to collect the required digits—for example, to request the PIN or account number for the caller. The TCL scripts are the default scripts for all Cisco voice features using IVR. TCL scripts are configured to control calls coming into or going out of the gateway.

**Note**

Ensure that you have loaded the version of TCL scripts that support IVR Version 2. These TCL scripts can be downloaded from the following Cisco.com URL:

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

The TCL IVR scripts shown below are listed as an example of the types of scripts available to be downloaded from the cisco.com Software Center. For a complete list of scripts, it is recommended that you check the Software Center.

Cisco provides the following IVR scripts:

- `fax_hop_on_1`—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_npw`—Performs as `clid_authen`, but uses a null password when authenticating, rather than DNIS numbers.
- `clid_authen_collect`—Authenticates the call with ANI and DNIS numbers and collects the destination data. If authentication fails, it collects the account and password.
- `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.

Table 32 lists the prompt audio files associated with the `clid_col_npw_3` script.

**Table 32** *clid\_col\_npw\_3 Script Prompt Audio Files*

Audio Filename	Action
<code>flash:enter_account.au</code>	Asks the caller to enter an account number. Played as the first request.
<code>flash:auth_fail_retry.au</code>	Asks the caller to reenter the account number. Plays after two failures.
<code>flash:enter_pin.au</code>	Asks the caller to enter a PIN.
<code>flash:enter_destination.au</code>	Asks the caller to enter a destination phone number.
<code>flash:auth_fail_final.au</code>	Informs the caller that the account number authorization has failed three times.

Table 33 lists additional audio files associated with the `clid_col_npw_3` script.

**Table 33** *Additional clid\_col\_npw\_3 Script Audio Files*

Audio Filename	Action
<code>auth_fail_retry.au</code>	Informs the caller that authorization failed. Prompts the caller to reenter the account number followed by the pound sign (#).
<code>auth_fail_final.au</code>	Informs the caller, “I’m sorry, your account number cannot be verified. Please hang up and try again.”

- `clid_col_npw_npw`—Tries to authenticate by using ANI, null as the user ID, user, and user password pair. If that fails, it collects an account number and authenticates with account and null. It allows three tries for the caller to enter the account number before ending the call with the authentication failed audio file. If authentication succeeds, it plays a prompt to enter the destination number.

Table 34 lists the audio files associated with the `clid_col_npw_npw` script.

**Table 34** *clid\_col\_npw\_npw Script Audio Files*

Audio Filename	Action
<code>flash:enter_account.au</code>	Asks the caller to enter the account number the first time.
<code>flash:auth_fail_retry.au</code>	Asks the caller to reenter the account number after first two failures.
<code>flash:enter_destination.au</code>	Asks the caller to enter the destination phone number.
<code>flash:auth_fail_final.au</code>	Informs the caller that the account number authorization has failed three times.

- `clid_col_dnis_3.tcl`—Authenticates the caller ID three times. First it authenticates the caller ID with DNIS. If that is not successful, it attempts to authenticate with the caller PIN up to three times.
- `clid_col_npw_3.tcl`—Authenticates with null. If authentication is not successful, it attempts to authenticate by using the caller PIN up to 3 times.
- `clid_4digits_npw_3.tcl`—Authenticates with null. If the authentication is not successful, it attempts to authenticate with the caller PIN up to 3 times using the 14-digit account number and password entered together.
- `clid_4digits_npw_3_cli.tcl`— Authenticates the account number and PIN respectively by using ANI and null. The number of digits allowed for the account number and password are configurable through the CLI. If the authentication fails, it allows the caller to retry. The retry number is also configured through the CLI.
- `clid_authen_col_npw_cli.tcl`—Authenticates the account number and PIN respectively using ANI and null. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.
- `clid_authen_collect_cli.tcl`—Authenticates the account number and PIN by using ANI and DNIS. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.
- `clid_col_npw_3_cli.tcl`—Authenticates by using ANI and null for account and PIN respectively. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI.
- `clid_col_npw_npw_cli.tcl`—Authenticates by using ANI and null for account and PIN respectively. If authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected together.

**Note**

To display the contents of the TCL IVR script, use the **show call application voice** command.

## TCL IVR Prerequisite Tasks

Before you configure your Cisco gateway to support TCL IVR, you must perform the following prerequisite tasks:

- Configure VoIP to support H.323-compliant gateways—meaning that in addition to the basic configuration tasks, such as configuring dial peers and voice ports, you must configure specific devices in your network to act as gateways.
- Configure a TFTP sever to perform storage and retrieval of the audio files, which are required by the Debit Card gateway or other features requiring TCL IVR scripts and audio files.
- Download the appropriate TCL IVR script from the Cisco.com. Use the **copy** command to copy your audio file (.au file) to your Flash memory, and the **audio-prompt load** command to read it into RAM. When you use TCL IVR applications, the gateway needs to know the URL where the TCL script can be found, as well as the URL of any audio file you want to use. Cisco IOS File System (IFS) is used to read the files, so any IFS-supported URLs can be used, which includes TFTP, FTP, or a pointer to a device on the router. During configuration of the application, you specify the URLs for the script and for the audio prompt. See the “Using URLs in IVR Scripts” chapter in the *TCL IVR API Version 2.0 Programmer's Guide* for more information.

- Make sure that your audio files are in the proper format. The TCL IVR prompts require audio file (.au) format of 8-bit, u-law, and 8-khz encoding. To encode your own audio files, we recommend that you use one of these two audio tools (or a tool of similar quality):
  - Cool Edit, manufactured by Syntrillium Software Corporation
  - AudioTool, manufactured by Sun Microsystems
- Make sure that your access platform has a minimum of 16 MB Flash and 128MB of DRAM memory.
- Install and configure the appropriate RADIUS security server in your network. The version of RADIUS that you are using must be able to support IETF-supported vendor specific attributes (VSAs), which are implemented by using IETF RADIUS attribute 26.

## TCL IVR Configuration Tasks List

Before starting the software configuration tasks for the TCL IVR Version 2.0 features, complete the following preinstallation tasks:

- Download the TCL scripts and audio files to be used with this feature from the Cisco.com.
- Store the TCL scripts and audio files on a TFTP server configured to interact with your gateway access server.
- Create the TCL IVR application script to use with the **call application voice** command when configuring IVR using TCL scripts. You create this application first and store it on a server or location where it can be retrieved by the access server.
- Define the call flow and pass the defined parameter values to the application. Depending on the TCL script you select, these values can include the language of the audio file and the location of the audio file. [Table 35](#) lists the TCL scripts and the parameter values they require.
- Associate the application to the incoming POTS or VoIP dial peer.

See the following sections for configuration tasks for the TCL IVR. Each task in the list is identified as either optional or required:

- [Configuring the Call Application for the Dial Peer](#) (Required)
- [Configuring TCL IVR on the Inbound POTS Dial Peer](#) or [Configuring TCL IVR on the Inbound VoIP Dial Peer](#) (Required)
- [Configuring MGCP Scripting](#) (Optional)



### Note

When an IVR script is used to detect a “long #” from a caller connected to the H.323 call leg, the DTMF method used must either be Cisco proprietary RTP or DTMF relay using H.245 signal IE. DTMF relay using H.245 alphanumeric IE does not report the actual duration of the digit, causing long pound (#) detection to fail.

## Configuring the Call Application for the Dial Peer

You must configure the application that interacts with the dial peer before you configure the dial peer. The dial peer collects digits from the caller and uses the application you have created. Use the **call application voice** command as shown in the table that follows. Each command line is optional depending on the type of action desired or the digits to be collected.

To configure the application, enter the following commands in global configuration mode:

	Command	Purpose
Step 1	Router(config)# <b>call application voice</b> <i>name url</i>	<p>Defines the name of the application to be used with your TCL IVR script. The <i>url</i> argument specifies the location of the file and the access protocol. An example is as follows:</p> <pre>flash:scripts/session.tcl tftp://dir/sarvi/scripts/session.tcl ftp://sarvi-ultra/scripts/session.tcl slot0:scripts/tcl/session..tcl</pre> <p><b>Note</b> You can only configure a url if the application named <i>name</i> has <i>not</i> been configured.</p>
Step 2	Router(config)# <b>call application voice</b> <i>name language digit language</i>	<p>Specifies the language used by the audio files. An example is: <code>call application voice test language 1 en</code>. The arguments are as follows:</p> <ul style="list-style-type: none"> <li><i>digit</i>—Specifies zero (0) through 9.</li> <li><i>language</i>—Specifies two characters that represent a language. For example, “en” for English, “sp” for Spanish, and “ch” for Mandarin. Enter <b>aa</b> to represent all.</li> </ul>
Step 3	Router(config)# <b>call application voice</b> <i>name pin-length number</i>	<p>Defines the number of characters in the PIN for the designated application. Values are from 0 through 10.</p>
Step 4	Router(config)# <b>call application voice</b> <i>name retry-count number</i>	<p>Defines the number of times a caller is permitted to reenter the PIN for the designated application. Values are from 1 through 5.</p>
Step 5	Router(config)# <b>call application voice</b> <i>name uid-length number</i>	<p>Defines the number of characters allowed to be entered for the user ID for the designated application. Values are from 1 through 20.</p>
Step 6	Router(config)# <b>call application voice</b> <i>name set-location language category location</i>	<p>Defines the location, language, and category of the audio files for the designated application. An example is: <b>set-location</b> en 1 tftp://server dir/audio filename.</p>

Table 35 lists TCL script names and the corresponding parameters that are required for each TCL scripts.

Table 35 TCL Scripts and Parameters

TCL Script Name	Description—Summary	Commands to Configure
clid_4digits_npw_3_cli.tcl	Authenticates the account number and PIN using ANI and null. The allowed length of digits is configurable through the CLI. If the authentication fails, it allows the caller to retry. The retry number is also configured through the CLI.	<b>call application voice uid-len</b> min = 1, max = 20, default = 10 <b>call application voice pin-len</b> min = 0, max = 10, default = 4 <b>call application voice retry-count</b> min = 1, max = 5, default = 3
clid_authen_col_npw_cli.tcl	Authenticates the account number and PIN using ANI and null. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.	<b>call application voice retry-count</b> min = 1, max = 5, default = 3
clid_authen_collect_cli.tcl	Authenticates the account number and PIN using ANI and DNIS. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected separately.	<b>call application voice retry-count</b> min = 1, max = 5, default = 3
clid_col_npw_3_cli.tcl	Authenticates using ANI and null for account and PIN. If the authentication fails, it allows the caller to retry. The retry number is configured through the CLI.	<b>call application voice retry-count</b> min = 1, max = 5, default = 3
clid_col_npw_npw_cli.tcl	Authenticates using ANI and null for account and PIN. If authentication fails, it allows the caller to retry. The retry number is configured through the CLI. The account number and PIN are collected together.	<b>call application voice retry-count</b> min = 1, max = 5, default = 3

## Configuring TCL IVR on the Inbound POTS Dial Peer

Configuring gw-accounting and AAA are not always required for POTS dial peer configuration. It is dependent upon the type of application that is being used with TCL IVR. For example, the Pre-Paid Calling Card feature requires accounting and the authentication caller ID application does not.

To configure the inbound POTS dial peer, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	Router(config)# <b>aaa new-model</b>	(Optional) Enables AAA security and accounting services.
Step 2	Router(config)# <b>gw-accounting h323</b>	(Optional) Enables gateway-specific H.323 accounting.
Step 3	Router(config)# <b>aaa authentication login h323 radius</b>	(Optional) Defines a method list called H.323 where RADIUS is defined as the only method of login authentication.

	Command	Purpose
Step 4	Router(config)# <b>aaa accounting connection h323 start-stop radius</b>	(Optional) Defines a method list called H.323 where RADIUS is used to perform connection accounting, providing start-stop records.
Step 5	Router(config)# <b>radius-server host ip-address auth-port number acct-port number</b>	Identifies the RADIUS server and the ports that will be used for authentication and accounting services.
Step 6	Router(config)# <b>radius-server key key</b>	Specifies the password used between the gateway and the RADIUS server.
Step 7	Router(config)# <b>dial-peer voice number pots</b>	Enters dial-peer configuration mode to configure the incoming POTS dial peer. The <i>number</i> argument is a tag that uniquely identifies the dial peer.
Step 8	Router(dial-peer)# <b>application name</b>	Associates the TCL IVR application with the incoming POTS dial peer. Enter the selected TCL IVR application name.
Step 9	Router(config-dial-peer)# <b>destination-pattern string</b>	<p>Enters the telephone number associated with this dial peer. The <i>pattern</i> argument is a series of digits that specify the E.164 or private dialing plan telephone number. Valid entries are numbers from zero (0) through nine and letters from A through D. The following special characters can be entered in the string:</p> <ul style="list-style-type: none"> <li>• Plus sign (+)—(Optional) Indicates an E.164 standard number. The plus sign (+) is not supported on the Cisco MC3810 multiservice concentrator.</li> <li>• <i>string</i>—Specifies the E.164 or private dialing plan telephone number. Valid entries are the digits 0 through 9, the letters A through D, and the following special characters: <ul style="list-style-type: none"> <li>– Asterisk (*) and pound sign (#) that appear on standard touch-tone dial pads.</li> <li>– Comma (,) inserts a pause between digits.</li> <li>– Period (.) matches any entered digit (this character is used as a wildcard).</li> </ul> </li> <li>• T—(Optional) Indicates that the destination-pattern value is a variable length dial-string.</li> </ul>
Step 10	Router(config-dial-peer)# <b>session target</b>	Specifies the session target IP address.

## Configuring TCL IVR on the Inbound VoIP Dial Peer

To configure the inbound VoIP dial peer, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	Router(config)# <b>dial-peer voice 4401 voip</b>	Enters the dial-peer configuration mode and identifies the call leg.
Step 2	Router(config-dial-peer)# <b>application application-name</b>	Specifies the name of the application and script to use.
Step 3	Router(config-dial-peer)# <b>destination-pattern pattern</b>	Enters the destination pattern.
Step 4	Router(config-dial-peer)# <b>session protocol sipv2</b>	Specifies the session protocol. The default session protocol is H.323. The <i>sipv2</i> argument enables SIP.
Step 5	Router(config-dial-peer)# <b>session target</b>	Specifies the session target IP address.
Step 6	Router(config-dial-peer)# <b>dtmf-relay cisco-rtsp</b>	Specifies the DTMF relay method. The keyword <b>cisco-rtsp</b> specifies H.323 and SIP. Other keywords that are available only for H.323 are <b>h245-alphanumeric</b> and <b>h245-signal</b> .  <b>Note</b> If digit collection from this VoIP call leg is required, the command <b>dtmf-relay</b> is required. The default is <b>no dtmf-relay</b> .
Step 7	Router(config-dial-peer)# <b>codec g711ulaw</b>	Specifies the voice codec.  <b>Note</b> If the configured application will be playing prompts to the VoIP call leg, the <b>g711ulaw</b> keyword is required.

## Configuring MGCP Scripting

To perform MGCP scripting, you must enable the MGCP script package. Enable the script in global configuration mode by entering the **mgcp package-capability script package** command. The example MGCP configuration shown in this section is for DS0s on T1 lines. The configuration tasks are as follows:

- Enabling the MGCP service on the DS0 groups
- Enabling the other MGCP packages
- Configuring the call agent address and other MGCP parameters

To configure MGCP scripting, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	Router(config)# <b>mgcp</b>	Starts the MGCP daemon.
Step 2	Router(config)# <b>mgcp request timeout timeout</b>	Specifies how long the gateway should wait for a response to a request.
Step 3	Router(config)# <b>mgcp request retries count</b>	Specifies the number of times to retry sending the <b>mgcp</b> command.
Step 4	Router(config)# <b>mgcp call-agent {ipaddr   hostname} [port]</b>	Configures the address of the call agent.
Step 5	Router(config)# <b>mgcp max-waiting-delay value</b>	Configures the maximum waiting delay to be used in a restart in progress (RSIP) message as restart instructions for the call agent.

	Command	Purpose
Step 6	Router(config)# <b>mgcp restart-delay</b> <i>value</i>	Configures the restart delay value to be used in an RSIP message as graceful teardown instructions for the gateway connection.
Step 7	Router(config)# <b>mgcp vad</b>	Configure voice activity detection.
Step 8	Router(config)# <b>mgcp package-capability</b> { <b>as-package</b>   <b>dtmf-package</b>   <b>gm-package</b>   <b>rtp-package</b>   <b>trunk-package</b> }	Specifies an MGCP package capability.
Step 9	Router(config)# <b>mgcp default-package</b> { <b>as-package</b>   <b>dtmf-package</b>   <b>gm-package</b>   <b>rtp-package</b>   <b>trunk-package</b> }	Configures the default package capability type.
Step 10	Router(config)# <b>mgcp quality-threshold</b> { <b>hwm-jitter-buffer</b> <i>value</i>   <b>hwm-latency</b> <i>value</i>   <b>hwm-packet-loss</b> <i>value</i>   <b>lwm-jitter-buffer</b> <i>value</i>   <b>lwm-latency</b> <i>value</i>   <b>lwm-packet-loss</b> <i>value</i> }	Configures the jitter buffer size, packet-loss threshold, and latency threshold.
Step 11	Router(config)# <b>mgcp playout</b> { <b>adaptive</b> <i>init-value</i> <i>min-value</i> <i>max-value</i> }   { <b>fixed</b> <i>init-value</i> }	Tunes the jitter buffer packet size used for MGCP connections.
Step 12	Router(config)# <b>mgcp codec</b> <i>type</i> [ <b>packetization-period</b> <i>value</i> ]	Configures the default codec type.
Step 13	Router(config)# <b>mgcp ip-tos</b> { <b>high-reliability</b>   <b>high-throughput</b>   <b>low-cost</b>   <b>low-delay</b>   <b>precedence</b> <i>value</i> }	Enables the IP type of service for MGCP connections.
Step 14	Router(config)# <b>controller t1 slot#</b>	Uses the controller configuration mode for the T1 controller in the specified slot.
Step 15	Router(config-controller)# <b>framing</b> <i>type</i>	Configures the framing type.
Step 16	Router(config-controller)# <b>clock source</b> <i>type</i>	Configures the clock source.
Step 17	Router(config-controller)# <b>linecode</b> <i>type</i>	Configures the line coding.
Step 18	Router(config-controller)# <b>ds0-group</b> <i>n</i> <b>timeslots</b> <i>range</i> <b>type</b> <i>signaling-type</i> <b>service</b> <b>mgcp</b>	Configures the DS0s to support MGCP.

## Verifying TCL IVR Configuration

You can verify TCL IVR configuration by performing the following tasks:

- To verify TCL IVR configuration parameters, use the **show running-config** command.
- To display a list of all voice applications, use the **show call application summary** command.
- To show the contents of the script configured, use the **show call application voice** command.
- To verify that the operational status of the dial peer, use the **show dial-peer voice** command.

To verify the TCL IVR configuration, perform the following steps:

- Step 1** Enter the **show call application voice summary** command to verify that the newly created applications are listed. The example output follows:

```
Router# show call application voice summary
```

```

name                description
-----
DEFAULT            NEW::Basic app to do DID, or supply dialtone.
fax_hop_on         Script to talk to a fax redialer
clid_authen        Authenticate with (ani, dnis)
clid_authen_collect Authenticate with (ani, dnis), collect if that fails
clid_authen_npw    Authenticate with (ani, NULL)
clid_authen_col_npw Authenticate with (ani, NULL), collect if that fails
clid_col_npw_3     Authenticate with (ani, NULL), and 3 tries collecting
clid_col_npw_npw   Authenticate with (ani, NULL) and 3 tries without pw
SESSION            Default system session application
hotwo              tftp://hostname/scripts/nb/nb_handoffTwoLegs.tcl
hoone              tftp://hostname/scripts/nb/nb_dohandoff.tcl
hodest             tftp://hostname/scripts/nb/nb_handoff.tcl
clid               tftp://hostname/scripts/tcl_ivr/clid_authen_collect.tcl
db102              tftp://hostname/scripts/1.02/debitcard.tcl
*hw                tftp://171.69.184.xxx/tr_hello.tcl
*hw1               tftp://san*tr_db
tftp://171.69.184.235/tr_debitcard.answer.tcl

TCL Script Version 2.0 supported.
TCL Script Version 1.1 supported.
```



**Note**

In the output shown, an asterisk (\*) in an application indicates that this application was not loaded successfully. Use the **show call application voice** command with the *name* argument to view information for a particular application.

- Step 2** Enter the **show dial-peer voice** command with the *peer tag* argument and verify that the application associated with the dial peer is correct.
- Step 3** Enter the **show running-config** command to display the entire configuration.

# TCL IVR Configuration Examples

Use the **show running-config** command to display the entire gateway configuration. [Figure 93](#) shows the type of topology used in the configuration for the example.

**Figure 93 Example Configuration Topology**



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

This section provides the following configuration examples:

- [TCL IVR for Gateway1 \(GW1\) Configuration Example, page 471](#)
- [TCL IVR for GW2 Configuration Example, page 474](#)
- [MGCP Scripting Configuration Example, page 476](#)

## TCL IVR for Gateway1 (GW1) Configuration Example

The following output is the result of using the **show running-config** command:

```

GW1
Router# show running-config

Building configuration...

Current configuration:

! Last configuration change at 08:39:29 PST Mon Jan 10 2000 by lab
!
version 12.2
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 baloo 1.14.124.xxx
ip host dirt 223.255.254.254

```

```

ip host rtspserver3 1.14.1xx.2
ip host rtspserver1 1.14.1xx.2
!
mgcp package-capability trunk-package
mgcp default-package trunk-package
isdn switch-type primary-net5
isdn voice-call-failure 0
!
tftp://dirt/hostname/WV/en_new/
call application voice debit_card tftp://dirt/Router/scripts.new/app_debitcard.tcl
call application voice debit_card uid-len 6
call application voice debit_card language 1 en
call application voice debit_card language 2 ch
call application voice debit_card set-location ch 0 tftp://dirt/hostname/WV/ch_new/
call application voice debit_card set-location en 0 tftp://dirt/hostname/WV/en_new/
call application voice debit_card_rtsp tftp://dirt/IVR 2.0/scripts.new/app_debitcard.tcl
call application voice debit_card_rtsp uid-len 6
call application voice debit_card_rtsp language 1 en
call application voice debit_card_rtsp language 2 ch
call application voice debit_card_rtsp set-location ch 0 rtsp://rtspserver1:554/
call application voice debit_card_rtsp set-location en 0 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 1.14.128.35 255.255.255.xxx
  no ip directed-broadcast
  h323-gateway voip interface
  h323-gateway voip id gk1 ipaddr 1.14.128.19 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 16.0.0.1 255.255.255.0
  no ip directed-broadcast
  duplex full
  speed auto
  no cdp enable
!
ip classless
ip route 0.0.0.0 0.0.0.0 1.14.128.33
ip route 1.14.xxx.0 255.xxx.255.xxx 16.0.0.2

```

```
ip route 1.14.xxx.16 255.xxx.255.240 1.14.xxx.33
no ip http server
!
radius-server host 1.14.132.2 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 ipv4:16.0.0.2
  dtmf-relay h245-alphanumeric
  codec g711ulaw
!
dial-peer voice 102 pots
  application 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
  application 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 1.14.42.23
end

GW1#
```

## TCL IVR for GW2 Configuration Example

The following output is the result of using the **show running-config** command:

```

GW2#
Router# show running-config

Building configuration...

Current configuration:
!
! Last configuration change at 08:41:12 PST Mon Jan 10 2000 by lab
!
version 12.2
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 1.14.132.2
ip host radiusserver1 1.14.138.11
ip host baloo 1.14.124.254
ip host rtspserver2 1.14.136.2
ip host dirt 223.255.254.254
ip host rtspserver3 1.14.126.2
!
mgcp package-capability trunk-package
mgcp default-package trunk-package
isdn switch-type primary-5ess
isdn voice-call-failure 0
!
call application voice clid_authen_sky
tftp://dirt/hostname/sky_scripts/clid_authen_collect_cli_sky.tcl

call application voice rtsp_demo tftp://dirt/hostname/sky_scripts/rtsp_demo.tcl
tftp://dirt/hostname/WV/en_new/
call application voice debit_card tftp://dirt/IVR 2.0/scripts.new/app_debitcard.tcl
call application voice debit_card uid-len 6
call application voice debit_card language 1 en
call application voice debit_card language 2 ch
call application voice debit_card set-location ch 0 tftp://dirt/hostname/WV/ch_new/
call application voice debit_card set-location en 0 tftp://dirt/hostname/WV/en_new/
call application voice clid_authen_rtsp tftp://dirt/IVR
2.0/scripts.new/app_clid_authen_collect_cli_rtsp.tcl

call application voice clid_authen_rtsp location rtsp://rtspserver2:554/

```

```
call application voice clid_authen1 tftp://dirt/IVR
2.0/scripts.new/app_clid_authen_collect_cli_rtsp.tcl
call application voice clid_authen1 location tftp://dirt/hostname/WV/en_new/
call application voice clid_authen1 uid-len 6
call application voice clid_authen1 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 1.14.xxx.4 255.255.xxx.240
  no ip directed-broadcast
  h323-gateway voip interface
  h323-gateway voip id gk2 ipaddr 1.14.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 16.0.0.2 255.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 1.14.xxx.5
ip route 1.14.xxx.32 255.255.xxx.240 16.0.0.1
no ip http server
!
radius-server host 1.14.132.2 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
  application 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
application debit_card
incoming called-number 34.....
shutdown
session protocol sipv2
session target ipv4:16.0.0.1
dtmf-relay cisco-rtp
codec g711ulaw
!
dial-peer voice 102 voip
incoming called-number 34.....
destination-pattern 53.....
session target ipv4:16.0.0.1
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 1.14.42.23
end

GW2#

```

## MGCP Scripting Configuration Example

The following example displays only the MGCP specific portion of the configuration:

```

!
mgcp
mgcp request timeout 10000
mgcp request retries 1
mgcp call-agent 1.14.138.11
mgcp restart-delay 10
mgcp codec g723ar63 packetization-period 30
mgcp vad
mgcp package-capability gm-package
mgcp package-capability dtmf-package
mgcp package-capability trunk-package
mgcp package-capability rtp-package
mgcp package-capability as-package
mgcp package-capability script-package
mgcp default-package trunk-package
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
  ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 1
  framing esf
  clock source line secondary 1
  linecode b8zs
  ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 2
  framing esf
  linecode b8zs
  ds0-group 0 timeslots 1-24 type none service mgcp
!
controller T1 3
  framing esf linecode b8zs
  ds0-group 0 timeslots 1-24 type none service mgcp
!
end
```

