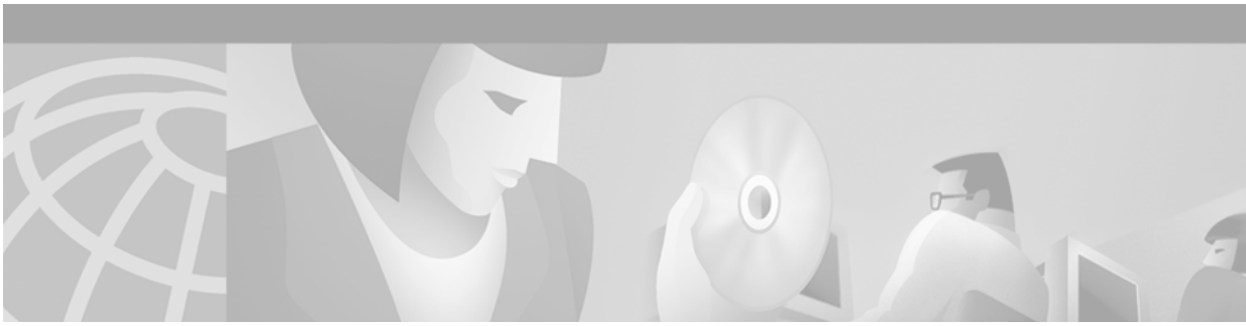

Configuring Interactive Voice Response for Cisco Access Platforms

Feature History

Release	Modification
11.(3)NA2	This feature was introduced.
12.0(7)T	This feature was implemented into the 12.0(7)T release.
12.1(5)XM2	Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.

This feature module describes both the original version of Interactive Voice Response (IVR) and the Tool Command Language (TCL) enhancements to IVR for Cisco IOS Release 12.0(7)T. The following sections include:

Feature Overview

Cisco Systems is building voice gateways to connect more traditional telephone networks to voice over IP (VoIP) networks. Customers who are installing VoIP networks often need a mechanism at the gateway to present a customized interface to the caller. The Interactive Voice Response (IVR) feature was first made available to customers in Cisco IOS Release 11.(3)NA2, with the Service Provider VoIP feature set. IVR, with the addition of scripts using Tool Command Language (TCL), is being introduced with Cisco IOS Release 12.0(7)T. These TCL IVR scripts are the default scripts that must be used with the IVR application in Cisco IOS Release 12.0(7)T.

IVR consists of simple voice prompting and digit collection to gather caller information for authenticating the user and identifying the destination. IVR provides the ability to:

- Play customized prompts

- Collect account numbers and PINs
- Collect destination phone numbers
- Perform Authentication, Authorization, and Accounting (AAA) tasks interacting with a variety of servers

See the "TCL IVR Scripts" section on page 4 for a description of the new scripts.

Note You can download TCL scripts that are not embedded in Cisco IOS software from the CCO Software Center at the following URL: http://www.cisco.com/kobayashi/sw-center/sw-access.shtml

Functional Description

The IVR feature allows the use of one of several interactive voice response scripts during the call processing. The TCL scripts are designed to interact with the IVR application software to perform the various functions. TCL scripts contain both executable files and audio files that interact with the system software. See the "TCL IVR Script Summaries" section on page 4.

The IVR application is used for information gathering and processing purposes, such as 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 PIN. After playing the voice prompt, the IVR application collects the predetermined number of touch tones (digit collection) and forwards the collected digits to a server for storage and retrieval. Call records can be kept and a variety of accounting functions performed.

IVR applications can be assigned to specific ports or invoked based on DNIS. (See the Glossary, page 57.) An Internet protocol (IP) public switched telephone network (PSTN) gateway can have several different IVR applications to accommodate many different gateway services, and you can customize the IVR applications to present different interfaces to the different callers.

Dial-Peer Application Field

Use the application field in the inbound dial peer to associate an application with an incoming call. This field is applicable only to the dial-peer of the POTS encapsulation type. The IVR scripts are examples of applications entered in this field. See the **call application voice** command in the "Command Reference" section.

When a call initially comes in, an application is selected by the Call Control Applications Programming Interface (CCAPI) based on the configuration in the inbound dial peer. (See Configuring the Inbound Dial-Peer with IVR, page 40.)

Script Descriptions

The following section describes both the Classic IVR scripts (which are the original IVR scripts in Cisco IOS Release 11.3(6)NA), and the new TCL IVR scripts, both in summary format and in detail.

Classic IVR Script Summaries

The classic IVR scripts audio files were introduced in Cisco IOS Release 11.3(6)NA2 and are described below. These audio files are provided by Cisco. However, we recommend that you record your own audio files to use with these scripts.

Note Use the **copy** command to copy your audio file (.au file) to your Flash memory, and the **audio-prompt load** command will read it into RAM.

To obtain a complete description of each IVR script, enter the **show call application voice** [<*application name*> | summary] command and insert the desired script name in the *application name* field. The screen output displays a description of each script. See the "Classic IVR Scripts in Detail" section on page 5.

- **fax_hop_on_1**—Collects digits from the redialer, such as account number and destination number. When placing the call 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 Dialed Number Identification Service (DNIS) numbers, collects the destination data, and makes the call.
- **clid_authen_npw**—Same 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, but if authentication fails, it collects the account and password.
- **clid_authen_col_npw**—Same as **clid_authen_collect**, but uses a null password and does not use or collect DNIS numbers.
- **clid_col_npw_3**—Same as **clid_authen_col_npw** except with that script if authentication with the digits collected (account and PIN number) fails, the script **clid_authen_col_npw** just plays a failure message (auth_failed.au) and then hangs up. This **clid_col_npw_3** script allows two failures, then plays the retry audio file (auth_retry.au) and collects the account and PIN numbers again.

  The caller can interrupt the message by entering digits for the account number, which triggers the prompt to tell the caller to enter the PIN number. If authentication fails the third time, the script plays the audio file **auth_fail_final.au**, and hangs up.

  The **clid_col_npw_3** script plays the following prompts:

Script Name	Action
flash:enter_account.au	Asks the caller to enter an account number the first time.
flash:auth_fail_retry.au	Played after two failures, asks the caller to reenter the account number.
flash:enter_pin.au	Asks the caller to enter a PIN number.
flash:enter_destination.au	Asks the caller to enter a destination phone number.
flash:auth_fail_final.au	Informs the caller that the authorization failed three times.

These are two new audio files:

Audio File Name	Action
auth_fail_retry.au	Informs the caller that authorization failed. Prompts the caller to please reenter the account number followed by the pound sign (#).
auth_fail_final.au	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.

  This IVR script plays the following audio files:

Audio File Name	Action
flash:enter_account.au	Asks the caller to enter the account number the first time.
flash:auth_fail_retry.au	Plays after first two failures, asks the caller to reenter the account number.
flash:enter_destination.au	Asks the caller to enter the destination phone number.
flash:auth_fail_final.au	Informs the caller that the authorization failed three times.

TCL IVR Scripts

The TCL IVR scripts are the default scripts for all Cisco voice features using IVR. All IVR scripts that were developed for releases before Cisco IOS Release 12.0(5)T are modified and secured with a proprietary Cisco locking mechanism using TCL. Only Cisco internal technical support personnel can open and modify these scripts. When the TCL script is activated, the system verifies the Cisco signature level. If the script is inconsistent with the authorized signature level, the script does not load, and the customer's console screen displays an error message.

Note You can download TCL scripts that are not embedded in Cisco IOS software from the CCO Software Center at the following URL: http://www.cisco.com/pcgi-bin/ibld/all.pl?i=support&c=3

TCL IVR Script Summaries

The following TCL IVR scripts available with Cisco IOS Release 12.0(3)T and 12.0(7)T are described:

Script Name	Description
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 3 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 fourteen 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 length of digits allowed for the account number and password are configurable through the command line interface (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 command line interface (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 command line interface (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 command line interface (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.

Classic IVR Scripts in Detail

The following IVR script descriptions are screen displays that are shown when you enter the **show call application voice <*script name*>** command. The Classic IVR scripts were introduced in Cisco IOS Release 12.0(3)T and earlier releases. These IVR scripts have been modified with TCL and introduced in Cisco IOS Release 12.0(7)T.

The following output displays the classic Classic IVR scripts in detail.

fax_hop_on_1

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

clid_authen

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

clid_authen_npw

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

end
```

clid_authen_collect

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

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

sblab115>show call application voice clid_authen_collect
Application clid_authen_collect has 10 states with 0 calls active
  State start has 1 actions and 5 events
    Do Action IVR_ACT_AUTHENTICATE. accountName=ani, pinName=dnis
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_CALL_SETUP_IND do action IVR_ACT_CALL_SETUP_ACK
          and goto state start
    If Event IVR_EV_AAA_SUCCESS goto state collect_dest
    If Event IVR_EV_AAA_FAIL goto state get_account
  State end has 1 actions and 3 events
    Do Action IVR_ACT_END.
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_CALL_DISCONNECT_DONE do action IVR_ACT_CALL_DESTROY
          and do nothing
  State get_account has 4 actions and 7 events
    Do Action IVR_ACT_PLAY.
            URL: flash:enter_account.au
            allowInt=1, pContent=0x60E4C564
    Do Action IVR_ACT_ABORT_KEY. abortKey=*
    Do Action IVR_ACT_TERMINATION_KEY. terminationKey=#
    Do Action IVR_ACT_COLLECT_PATTERN. Pattern account is .+
    If Event IVR_EV_DEFAULT goto state end
    If Event IVR_EV_CALL_DIGIT do nothing
    If Event IVR_EV_PAT_COL_SUCCESS goto state get_pin
            patName=account
    If Event IVR_EV_ABORT goto state get_account
    If Event IVR_EV_PLAY_COMPLETE do nothing
    If Event IVR_EV_TIMEOUT goto state get_account count=0
    If Event IVR_EV_PAT_COL_FAIL goto state get_account
  State get_pin has 4 actions and 7 events
    Do Action IVR_ACT_PLAY.
            URL: flash:enter_pin.au
            allowInt=1, pContent=0x0
```

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

clid_authen_col_npw

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

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

clid_col_npw_3

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

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

clid_col_npw_npw

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

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

TCL IVR Script in Detail

The following TCL IVR script description is shown in detail when you enter the **show call application voice <*script name*>** command.

Note You can download TCL scripts that are not embedded in Cisco IOS software from the CCO Software Center at the following URL: http://www.cisco.com/kobayashi/sw-center/sw-access.shtml

Caller ID Authenticate and Collect TCL Script

This TCL script is called *clid_authen_collect.tcl* and is shown in detail below. This is an example of a script that is used to interact with the Debit Card Feature in Cisco IOS Release 12.0(7)T.

```
Router# show call application voice clid
Idle call list has 1 calls on it.
Application clid
    The script is read from URL tftp://keyer/cchiu/devtest/clid_authen_collect.tcl
    The signature is invalid
    The uid-len is 10               (Default)
    The pin-len is 4                (Default)
    The warning-time is 60          (Default)
    The retry-count is 3            (Default)
    The language 1 is set to en
    The audio files for language en, all categories are read from
        URL tftp://keyer/cchiu/prompts/en/
    It has 0 calls active.

The TCL Script is:
------------------
# clid_authen_collect.tcl
#-----------------------------------------------------------------
# September 1998, Software Development
#
# Copyright (c) 1998, 1999 by Cisco Systems, Inc.
# All rights reserved.
#-----------------------------------------------------------------
# Mimic the clid_authen_collect script in the SP1.0 release.
#
# It authenticates using (ani, dnis) for (account, password). If
# that fails, it collects account and pin number, then authenticates
# using (account, pin).
#
# If authentication passes, it collects the destination number and
# places the call.
#
# The main routine is at the bottom. Start reading the script there.
#

proc do_get_account {} {
global state
global account
global fcnt

if { $fcnt == 0 } {
set prompt(url) flash:enter_account.au
} else {
set prompt(url) flash:auth_fail_retry_number.au
}
set prompt(interrupt) true
set prompt(abortKey) *
set prompt(terminationKey) #
set patterns(account) .+
set event [promptAndCollect prompt info patterns ]

if {$event == "collect success"} {
set state get_pin
set account $info(digits)
return 0
}

if {$event == "collect aborted"} {
```

```
set state get_account
return 0
}>
if {$event == "collect fail"} {
set state get_account
return 0
}
set state end
return 0
}

proc do_get_pin {} {
global state
global pin

set prompt(url) flash:enter_pin.au
set prompt(interrupt) true
set prompt(abortKey) *
set prompt(terminationKey) #
set patterns(account) .+
set event [promptAndCollect prompt info patterns ]

if {$event == "collect success"} {
set state authenticate
set pin $info(digits)
return 0
}

if {$event == "collect aborted"} {
set state get_account
return 0
}

if {$event == "collect fail"} {
# timeout
if {$info(code) == 102} {
set state get_pin
return 0
}

# invalid number
if {$info(code) == 28} {
set state get_pin
return 0
}
}

set state end
return 0
}

proc do_authenticate {} {
global state
global pin
global account
global fcnt

set event [authenticate $account $pin info]

if { $event == "authenticated" } {
set state authen_pass
return 0
}
```

```
if {$event == "authentication failed"} {
incr fcnt
if { $fcnt < 3 } {
set state get_account
} else {
set state authen_fail
}
return 0
}

set state end
return 0
}

proc do_get_dest {} {
global state
global destination>
set prompt(url) flash:enter_destination.au
set prompt(interrupt) true
set prompt(abortKey) *
set prompt(terminationKey) #
set prompt(dialPlan) true

set event [promptAndCollect prompt info ]

if {$event == "collect success"} {
set state place_call
set destination $info(digits)
return 0
}

if {$event == "collect aborted"} {
set state get_dest
return 0
}

if {$event == "collect fail"} {
set state get_dest
return 0
}
set state end
return 0
}

proc do_authen_pass {} {
global state
global destination

set dnislen [string len [dnis]]

if { [did] && $dnislen } {
set destination [dnis]
set state place_call
} else {
set state get_dest
}
return 0
}

proc do_place_call {} {
global state
global destination

set event [placeCall $destination callInfo info]
```

```
if {$event == "active"} {
set state active
return 0
}
if {$event == "call fail"} {
set state place_fail
return 0
}

set state end
return 0
}

proc do_active_notimer {} {
global state

set event [waitEvent]
while { $event == "digit" } {
set event [waitEvent]
}
set state end
return 0
}

proc do_active_last_timer {} {
global state

set event [startTimer [creditTimeLeft] info]
while { $event == "digit" } {
set event [startTimer $info(timeLeft) info]
}
if { $event == "timeout" } {
clearOutgoingLeg retInfo
set state out_of_time
} else {
set state end
}
return 0
}

proc do_active_timer {} {
global state

set delay [expr [creditTimeLeft] - 10]
set event [startTimer $delay info]
while { $event == "digit" } {
set event [startTimer $info(timeLeft) info]
}
if { $event == "timeout" } {
insertMessage flash:beep.au retInfo
do_active_last_timer
} else {
set state end
}

return 0
}

proc do_active {} {
global state

if { ( [creditTimeLeft] == "unlimited") ||
([creditTimeLeft] == "uninitialized") } {
```

```
do_active_notimer
} else {
do_active_timer
}
return 0
}
proc do_out_of_time {} {
global state

set prompt(url) flash:out_of_time.au
set prompt(playComplete) true
set event [promptAndCollect prompt info ]
set state end
return 0
}

proc do_authen_fail {} {
global state

#set prompt(url) tftp://keyer/echeng/bowie_audio/auth_failed.au
set prompt(url) flash:auth_fail_final.au
set prompt(playComplete) true
set event [promptAndCollect prompt info ]
set state end
return 0
}
proc do_place_fail {} {
global state

playFailureTone 5 retInfo
set state end
return 0
}
#---------------------------------------------------------
# And here is the main loop
#
acceptCall
set fcnt 0
set event [authenticate [ani] [dnis] info]
if {$event != "authenticated"} {
set state get_account
} else {
set state authen_pass
}
while {$state != "end"} {
puts "cid([callID]) running state $state"
if {$state == "get_account"} {
do_get_account
} elseif {$state == "get_pin"} {
do_get_pin
elseif {$state == "authenticate"} {
do_authenticate
} elseif {$state == "get_dest"} {
do_get_dest
} elseif {$state == "place_call"} {
do_place_call
} elseif {$state == "active"} {
do_active>    } elseif {$state == "authen_fail" } {
do_authen_fail
} elseif {$state == "authen_pass" } {
do_authen_pass
} elseif {$state == "place_fail"} {
do_place_fail>    } elseif {$state == "out_of_time"} {
do_out_of_time
```

```
} else {
break
}
}end
```

TCL Script Examples

The original versions of the IVR scripts (“classic”) have been modified using the TCL scripting language. A few examples of TCL scripts follow. These examples are shown for comparison purposes only.

Session.tcl

```
Router # show call application voice session
Application session
    The script is compiled into the image
    It has 0 calls active.

The TCL Script is:
------------------
# session.tcl
#------------------
# September 1998, Development Engineer name
#
# Copyright (c) 1998, 1999 by cisco Systems, Inc.
# All rights reserved.
#----------------------------------
#
# This tcl script mimics the default SESSION app
#
# If DID is configured, just place the call to the dnis
# Otherwise, output dial-tone and collect digits from the
# caller against the dial-plan.
#
# Then place the call. If successful, connect it up, otherwise
# the caller should hear a busy or congested signal.

# Don't accept until we can place outgoing side.
# acceptCall
#
# The main routine is at the bottom. Start reading there.

proc do_active_notimer {} {
    set event [waitEvent]
    while { $event == "digit" } {
        set event [waitEvent]
    }
proc do_active_last_timer {} {

    set event [startTimer [creditTimeLeft] info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
    if { $event == "timeout" } {
        clearOutgoingLeg retInfo
        do_out_of_time
    }
proc do_active_timer {} {

    if { [creditTimeLeft] < 10 } {
        do_active_last_timer
        return 0
    }
    set delay [expr [creditTimeLeft] - 10]
    set event [startTimer $delay info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
    if { $event == "timeout" } {
        insertMessage flash:beep.au retInfo
        do_active_last_timer
    }
proc do_out_of_time {} {
```

```
    set prompt(url) flash:out_of_time.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
# And here is the main routine
#
#if DID, don't collect digits, just place the call.

if {[did]} {
    set destination [dnis]
} else {
#   play a dial tone, and collect against the pattern setupAck
    playTone Dial
    set prompt(dialPlan) true
    set event [promptAndCollect prompt retInfo]
    if {$event != "collect success"} {
        puts "Call [callID] got event $event collecting destination"
        exit 0
    }
    set destination $retInfo(digits)
    callProceeding

# we have not accepted the call yet. placeCall will place the outbound
# call, conference the two call legs, and accept the inbound call leg.
#
set callInfo(destinationNum) $destination
set event [placeCall $destination callInfo info]

if {$event != "active"} {
    puts "Call [callID] got $event when placing call\n"
    exit 0
}
# The call is active, they are talking.
#
if { ( [creditTimeLeft] == "unlimited") ||
([creditTimeLeft] == "uninitialized") } {
    do_active_notimer
} else {
    do_active_timer
end
```

Clid_authen.tcl

```
Router # show call application voice clid_authen.tcl
Application clid_authen
    The script is compiled into the image
    It has 0 calls active.

The TCL Script is:
------------------
# clid_authen.tcl
#----------------------------------
# September 1998, Development Engineer name
 --More-
#
# Copyright (c) 1998, 1999 by cisco Systems, Inc.
# All rights reserved.
#----------------------------------
# Mimic the clid_authen script in the SP1.0 release.
#
# It authenticates using (ani, dnis) for (account, password). If
# that fails, play a message and end the call.
#
# If authentication passes, it collects the destination number and
# places the call.
#
# The main routine is at the bottom. Start reading the script there.
#

proc do_get_dest {} {
    global state
    global destination

    playTone Dial
    set prompt(dialPlan) True
    set prompt(terminationKey) #
set event [promptAndCollect prompt info ]

    if {$event == "collect success"} {
        set state place_call
        set destination $info(digits)
        return 0
    }
if {$event == "collect aborted"} {
        set state get_dest
        return 0
    }
set state get_fail
    return 0
}
proc do_authen_pass {} {
    global state
    global destination

    set dnislen [string len [dnis]]

if { [did] && $dnislen } {
        set destination [dnis]
        set state place_call
    } else {
        set state get_dest
    }
    return 0
}
```

```
proc do_place_call {} {
    global state
    global destination

    set event [placeCall $destination callInfo info]

    if {$event == "active"} {
        set state active
        return 0
    }
    if {$event == "call fail"} {
        set state place_fail
        return 0
    }
set state end
    return 0
}
proc do_active_notimer {} {
    global state

    set event [waitEvent]
    while { $event == "digit" } {
        set event [waitEvent]
    }
    set state end
    return 0
}
proc do_active_last_timer {} {
    global state

    set event [startTimer [creditTimeLeft] info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
if { $event == "timeout" } {
        clearOutgoingLeg retInfo
        set state out_of_time
    } else {
        set state end
    }

    return 0
}
proc do_active_timer {} {
    global state

    if { [creditTimeLeft] < 10 } {
        do_active_last_timer
        return 0
    }
    set delay [expr [creditTimeLeft] - 10]
    set event [startTimer $delay info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
    if { $event == "timeout" } {
insertMessage flash:beep.au retInfo
        do_active_last_timer
    } else {
        set state end
    }

    return 0
}
```

```
proc do_active {} {
    global state

    if { ( [creditTimeLeft] == "unlimited") ||
         ([creditTimeLeft] == "uninitialized") } {
             do_active_notimer
    } else {
             do_active_timer
    }
    return 0
}

proc do_out_of_time {} {
    global state
 --More--

    set prompt(url) flash:out_of_time.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}

proc do_authen_fail {} {
    global state

    set prompt(url) flash:auth_failed.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}

proc do_get_fail {} {
    global state


    playTone None
 --More--

    set prompt(url) flash:collect_failed.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}

proc do_place_fail {} {
    global state

    playFailureTone 5 retInfo
    set state end
    return 0
}
#---------------------------------------
# And here is the main loop
#

acceptCall

 --More--
set event [authenticate [ani] [dnis] info]
```

```
if {$event != "authenticated"} {
    set state authen_fail
} else {
    set state authen_pass
}

while {$state != "end"} {
    puts "cid([callID]) running state $state"
    if {$state == "get_dest"} {
        do_get_dest
    } elseif {$state == "place_call"} {
        do_place_call
    } elseif {$state == "active"} {
        do_active
    } elseif {$state == "authen_fail" } {
        do_authen_fail
    } elseif {$state == "authen_pass" } {
        do_authen_pass
    } elseif {$state == "place_fail"} {
        do_place_fail
    } elseif {$state == "get_fail"} {
 --More--
      do_get_fail
    } elseif {$state == "out_of_time"} {
        do_out_of_time
    } else {
        break
    }end
```

Clid_authen_collect.tcl

```
Router # show call application voice clid_authen_collect
Application clid_authen_collect
    The script is compiled into the image
    It has 0 calls active.

The TCL Script is:
------------------
# clid_authen_collect.tcl
#---------------------------------
# September 1998, Software Development
#
# Copyright (c) 1998, 1999 by Cisco Systems, Inc.
# All rights reserved.
#---------------------------------
# Mimic the clid_authen_collect script in the SP1.0 release.
#
# It authenticates using (ani, dnis) for (account, password). If
# that fails, it collects account and pin number, then authenticates
# using (account, pin).
#
# If authentication passes, it collects the destination number and
# places the call.
#
# The main routine is at the bottom. Start reading the script there.
#
proc do_get_account {} {
    global state
    global account

    set prompt(url) flash:enter_account.au
    set prompt(interrupt) true
    set prompt(abortKey) *
    set prompt(terminationKey) #
    set patterns(account) .+
    set event [promptAndCollect prompt info patterns ]

    if {$event == "collect success"} {
        set state get_pin
set account $info(digits)
        return 0
    }

    if {$event == "collect aborted"} {
        set state get_account
        return 0
    }

    if {$event == "collect fail"} {
        set state get_account
        return 0
    }
    set state end
    return 0
}
proc do_get_pin {} {
    global state
    global pin

    set prompt(url) flash:enter_pin.au
    set prompt(interrupt) true
 --More--
```

```
 set prompt(abortKey) *
    set prompt(terminationKey) #
    set patterns(account) .+
    set event [promptAndCollect prompt info patterns ]

    if {$event == "collect success"} {
        set state authenticate
        set pin $info(digits)
        return 0
    }
if {$event == "collect aborted"} {
        set state get_account
        return 0
    }
if {$event == "collect fail"} {
        # timeout
        if {$info(code) == 102} {
            set state get_pin
            return 0
        }
# invalid number
        if {$info(code) == 28} {
            set state get_pin
            return 0
        }
set state end
    return 0
}
proc do_authenticate {} {
    global state
    global pin
    global account

    set event [authenticate $account $pin info]

    if { $event == "authenticated" } {
        set state authen_pass
        return 0
    }

if {$event == "authentication failed"} {
        set state authen_fail
        return 0
    }
set state end
    return 0
}
proc do_get_dest {} {
    global state
    global destination

    set prompt(url) flash:enter_destination.au
    set prompt(interrupt) true
    set prompt(abortKey) *
    set prompt(terminationKey) #
    set prompt(dialPlan) true

    set event [promptAndCollect prompt info ]

    if {$event == "collect success"} {
        set state place_call
set destination $info(digits)
        return 0
    }
```

```
if {$event == "collect aborted"} {
        set state get_dest
        return 0
    }
if {$event == "collect fail"} {
        set state get_dest
        return 0
    }
    set state end
    return 0
}
proc do_authen_pass {} {
    global state
    global destination

    set dnislen [string len [dnis]]

if { [did] && $dnislen } {
        set destination [dnis]
        set state place_call
    } else {
        set state get_dest
    }
    return 0
}
proc do_place_call {} {
    global state
    global destination

    set event [placeCall $destination callInfo info]

    if {$event == "active"} {
        set state active
        return 0
    }
    if {$event == "call fail"} {
        set state place_fail
        return 0
    }
set state end
    return 0
}
proc do_active_notimer {} {
    global state

    set event [waitEvent]
    while { $event == "digit" } {
        set event [waitEvent]
    }
    set state end
    return 0
}
proc do_active_last_timer {} {
    global state

    set event [startTimer [creditTimeLeft] info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
if { $event == "timeout" } {
        clearOutgoingLeg retInfo
        set state out_of_time
    } else {
        set state end
```

```
    }
return 0
}
proc do_active_timer {} {
    global state

    if { [creditTimeLeft] < 10 } {
        do_active_last_timer
        return 0
    }
    set delay [expr [creditTimeLeft] - 10]
    set event [startTimer $delay info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
    if { $event == "timeout" } {
insertMessage flash:beep.au retInfo
        do_active_last_timer
    } else {
        set state end
    }
return 0
}

proc do_active {} {
    global state

    if { ( [creditTimeLeft] == "unlimited") ||
         ([creditTimeLeft] == "uninitialized") } {
            do_active_notimer
    } else {
            do_active_timer
    }
    return 0
}
proc do_out_of_time {} {
    global state

    set prompt(url) flash:out_of_time.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}
proc do_authen_fail {} {
    global state

    set prompt(url) flash:auth_failed.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}
proc do_place_fail {} {
    global state

    playFailureTone 5 retInfo
    set state end
return 0
}
#--------------------------------------
# And here is the main loop
#
acceptCall
```

```
set event [authenticate [ani] [dnis] info]

if {$event != "authenticated"} {
    set state get_account
} else {
    set state authen_pass
}
while {$state != "end"} {
    puts "cid([callID]) running state $state"
    if {$state == "get_account"} {
        do_get_account
    } elseif {$state == "get_pin"} {
do_get_pin
    } elseif {$state == "authenticate"} {
        do_authenticate
    } elseif {$state == "get_dest"} {
        do_get_dest
    } elseif {$state == "place_call"} {
        do_place_call
    } elseif {$state == "active"} {
        do_active
    } elseif {$state == "authen_fail" } {
        do_authen_fail
    } elseif {$state == "authen_pass" } {
        do_authen_pass
    } elseif {$state == "place_fail"} {
        do_place_fail
    } elseif {$state == "out_of_time"} {
        do_out_of_time
    } else {
        break
    }end
```

Clid_authen_npw.tcl

```
Router # show call application voice clid_authen_npw.tcl
Application clid_authen_npw
    The script is compiled into the image
    It has 0 calls active.

The TCL Script is:
------------------
# clid_authen_npw.tcl
#----------------------------------
# September 1998, Software Engineer
#
# Copyright (c) 1998, 1999 by Cisco Systems, Inc.
# All rights reserved.
#----------------------------------
# Mimic the clid_authen_npw script in the SP1.0 release.
#
# It authenticates using (ani, null) for (account, password). If
# that fails, play a failure message and exit.
#
# If authentication passes, it collects the destination number and
# places the call.
#
# The main routine is at the bottom. Start reading the script there.
#

proc do_get_dest {} {
    global state
    global destination

    set prompt(url) flash:enter_destination.au
    set prompt(interrupt) true
    set prompt(abortKey) *
    set prompt(terminationKey) #
    set prompt(dialPlan) true
    set event [promptAndCollect prompt info ]

    if {$event == "collect success"} {
        set state place_call
        set destination $info(digits)
        return 0
    }
if {$event == "collect aborted"} {
        set state get_dest
        return 0
    }
set state get_fail
    return 0
}
proc do_authen_pass {} {
    global state
    global destination

    set dnislen [string len [dnis]]

    if { [did] && $dnislen } {
        set destination [dnis]
        set state place_call
    } else {
        set state get_dest
    }
    return 0
```

```
}
proc do_place_call {} {
    global state
    global destination

set event [placeCall $destination callInfo info]

    if {$event == "active"} {
        set state active
        return 0
    }
    if {$event == "call fail"} {
        set state place_fail
        return 0
    }

    set state end
    return 0
}
proc do_active_notimer {} {
    global state

    set event [waitEvent]
    while { $event == "digit" } {
        set event [waitEvent]
    }
    set state end
return 0
}

proc do_active_last_timer {} {
    global state

    set event [startTimer [creditTimeLeft] info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
    if { $event == "timeout" } {
        clearOutgoingLeg retInfo
        set state out_of_time
    } else {
        set state end
    }

    return 0
}
proc do_active_timer {} {
    global state

 --More--
  if { [creditTimeLeft] < 10 } {
        do_active_last_timer
        return 0
    }
    set delay [expr [creditTimeLeft] - 10]
    set event [startTimer $delay info]
    while { $event == "digit" } {
        set event [startTimer $info(timeLeft) info]
    }
    if { $event == "timeout" } {
        insertMessage flash:beep.au retInfo
        do_active_last_timer
    } else {
        set state end
```

```
    }

    return 0
}
proc do_active {} {
    global state

    if { ( [creditTimeLeft] == "unlimited") ||
([creditTimeLeft] == "uninitialized") } {
            do_active_notimer
    } else {
            do_active_timer
    }
    return 0
}
proc do_out_of_time {} {
    global state

    set prompt(url) flash:out_of_time.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}
proc do_authenticate_fail {} {
    global state

    set prompt(url) flash:auth_failed.au
    set prompt(playComplete) true
 --More--
  set event [promptAndCollect prompt info ]
    set state end
    return 0
}

proc do_get_fail {} {
    global state

    set prompt(url) flash:collect_failed.au
    set prompt(playComplete) true
    set event [promptAndCollect prompt info ]
    set state end
    return 0
}

proc do_place_fail {} {
    global state

    playFailureTone 5 retInfo
    set state end
    return 0
}
#--------------------------------------
# And here is the main loop
#

acceptCall

set event [authenticate [ani] "" info]

puts "authenticate returned $event"

if {$event != "authenticated"} {
    set state authenticate_fail
```

```
} else {
    set state authen_pass
}
while {$state != "end"} {
    puts "cid([callID]) running state $state"
    if {$state == "get_dest"} {
        do_get_dest
    } elseif {$state == "authen_pass" } {
        do_authen_pass
} elseif {$state == "place_call"} {
        do_place_call
    } elseif {$state == "active"} {
        do_active
    } elseif {$state == "authenticate_fail" } {
        do_authenticate_fail
    } elseif {$state == "place_fail"} {
        do_place_fail
    } elseif {$state == "get_fail"} {
        do_get_fail
    } elseif {$state == "out_of_time"} {
        do_out_of_time
    } else {
        puts "breaking because state is $state\n"
        break
}
end
```

Benefits

- TCL IVR scripts are intended to meet the growing need of service providers to have voice interaction with the caller.
- Support up to half of the total call capacity on the individual platform on authentication and/or authorization processes, while the other half is already in progress.

Restrictions

Ensure that you have the version of VCWare loaded on your Cisco AS5300 that is compatible with the Cisco IOS Release that you are running.

Related Features and Technologies

- *Service Provider feature set for VoIP* uses the IVR for interaction with the caller; collects digits for account and billing purposes.
- *Authentication, Authorization, and Accounting* (AAA) feature is used in conjunction with IVR.
- *Settlement for Packet Telephony on Cisco Access Platforms* uses the TCL IVR scripts for the billing process.
- *Debit Card for Packet Telephony on Cisco Access Platforms* uses TCL IVR extensively for interoperability.

Related Documents

See Related Documents for *Configuring Interactive Voice Response for Cisco Access Platforms*.

Supported Platforms

This feature is supported on the following platforms:

- Cisco 2600 series routers
- Cisco 3600 series routers
- Cisco AS5300 universal access server
- Cisco AS5400 universal gateway

Supported Standards, MIBs, and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

No new or modified MIBs are supported by this feature.

To obtain lists of MIBs supported by platform and Cisco IOS release and to download MIB modules, go to the Cisco MIB web site on Cisco Connection Online (CCO) at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

The Cisco AS5350 and Cisco AS5400 do not support the Mica Modem Card, Microcom Modem Card, or VoIP Feature Card. Voice and modem functions are provided by the Universal Port Dial Feature card running SPE firmware. See the *Cisco AS5350 Universal Gateway Card Installation Guide* and the *Cisco AS5400 Universal Gateway Card Installation Guide* for more information. All references to the Cisco AS5300 in this document apply to the Cisco AS5350 and Cisco AS5400 platforms with the following exceptions:

- Use the Universal Port Dial Feature Card instead of the Mica or Microcom modem cards.
- Use SPE firmware instead of portware version 6.7.7.
- Run Cisco IOS Release 12.1(5)XM2 software for VoIP functionality.

Other Prerequisites

- Establish a working IP network. For more information about configuring IP, refer to the "IP Overview," "Configuring IP Addressing," and "Configuring IP Services" chapters in the Cisco IOS Release 12.0 *Network Protocols Configuration Guide, Part 1*.

- Configure Voice over IP. For more information about configuring Voice over IP, refer to the "*Voice over IP Software Configuration Guide"* for the appropriate access platform.
- 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.
- Program and configure the interface between the RADIUS server and the Cisco gateway to operate with Vendor Specific Attributes.
- Download the TCL scripts that are not embedded in Cisco IOS from the Cisco CCO software support URL:
http://www.cisco.com/pcgi-bin/ibld/all.pl?i=support&c=3
- The IVR prompts require audio file (.au) format of 8-bit, Mu-law, and 8KHz encoding.
Cisco recommends these two audio tools or others of comparable quality:
- Cool Edit, manufactured by Syntrillium Software Corporation
- AudioTool, manufactured by Sun Microsystems Inc.
- Ensure that your access platform has a minimum of 16 MB Flash and 64 MB of DRAM memory.

Configuration Tasks

Perform the following tasks to configure your Cisco access platform for Interactive Voice Response with TCL scripts:

- Be sure to download the appropriate TCL IVR scripts from the CCO Software Support Center.
- Configure the IVR inbound dial-peer by using the *application* field.

Configuring the Inbound Dial-Peer with IVR

To configure the inbound VoIP dial-peer for IVR, enter the following commands:

Note When configuring a voice port, use the following configuration designations:
For the Cisco AS5300 access server, port designation is *port*.
For the Cisco AS5400 access server, port designation is *slot/port*.
For the Cisco AS5800 access server, port designation is *shelf/slot/port*.

	Command	Purpose
Step 1	`Router # configure terminal`	Enters the global configuration mode.
Step 2	`Router(config) # dial-peer`	Enters the dial-peer configuration mode.
Step 3	`Router(config-dial-peer) # dial-peer voice number pots`	Enters the dial-peer configuration mode to configure a POTS dial peer. The *number* value of the **dial-peer voice pots** command is a tag that uniquely identifies the dial peer.

	Command	Purpose
Step 4	`Router(config-dial-peer)# call application voice` *application name*	Enters the command to initiate the IVR application and the selected TCL application name. Enter the application name and the location where the IVR TCL script is stored.
Step 5	`Router(config-dial-peer)# destination-pattern` *call-number*	Enters the number to be dialed when calls fail. For example, the operator number.
Step 6	`Router(config-dial-peer)# port` *port-number*	Configures the voice port associated with this dial peer.

Verifying IVR Configuration

Enter the **show running configuration** command to generate screen output showing the configured parameters. The following IVR configuration uses the **clid_col_dnis_3.tcl** TCL IVR script:

```
Router # show running-config
Building configuration...

Current configuration:
!
! Last configuration change at 17:57:03 pst Wed Feb 24 1999
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname sblab115
!
boot system tftp username/c3620-is56i-mz 172.29.248.12
boot system tftp username/c3620-is-mz 172.29.248.12
no logging buffered
aaa new-model
aaa authentication login no_rad local
aaa authentication login h323 group radius local
!
clock timezone pst -8
clock summer-time pdt recurring
ip subnet-zero
ip domain-name cisco.com
ip name-server 172.29.248.16
ip name-server 171.69.187.13
!
call application voice c4 tftp://santa/username/clid_4digits_npw_3.tcl
call application voice c5 tftp://santa/username/clid_col_dnis_3.tcl
!
voice-port 1/0/0
!
voice-port 1/0/1
!
voice-port 1/1/0
 shutdown
 pre-dial-delay 0
!
voice-port 1/1/1
 shutdown
 pre-dial-delay 0
!
dial-peer voice 997 voip
 destination-pattern 997
 session target loopback:rtp
!
dial-peer voice 1 pots
 application clid
!
dial-peer voice 2 pots
!
dial-peer voice 100 pots
 application c5
 answer-address 1234
 destination-pattern 100
```

```
 port 1/0/0
!
voice-port 1/1/0
 shutdown
 pre-dial-delay 0
!
voice-port 1/1/1
 shutdown
 pre-dial-delay 0
!
dial-peer voice 997 voip
 destination-pattern 997
 session target loopback:rtp
!
dial-peer voice 1 pots
 application clid
!
dial-peer voice 2 pots
!
dial-peer voice 100 pots
 application c5
 answer-address 1234
 destination-pattern 100
 port 1/0/0
!
dial-peer voice 110 pots
 application clid
 destination-pattern 110
 direct-inward-dial
 port 1/1/0
!
dial-peer voice 111 pots
 destination-pattern 111
 port 1/1/1
!
dial-peer voice 114 voip
 destination-pattern 114...
 session target dns:sblab114
!
dial-peer voice 991 pots
 destination-pattern 991
 port 1/0/0
 session target loopback:uncompressed
!
dial-peer voice 992 pots
 destination-pattern 992
 port 1/0/1
 session target loopback:uncompressed

Router #
```

Command Reference

This section only documents the following new or modified commands. All other commands used with this feature are documented in the Cisco IOS Release 12.1 command reference publications.

- **application**
- **audio-prompt load**
- **call application voice**
- **call application voice load**
- **show call application voice**

application

To configure the dial peers, use the **application** command. This command selects the session application for Interactive Voice Response.

application *name*

Syntax Description

name	Indicates the name of the IVR script application the call should be handed to

Defaults

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

Command Modes

Dial-peer configuration mode

Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.1(5)XM2	The command was introduced for the Cisco AS5350 and CiscoAS5400.

Usage Guidelines

Use this field when configuring the inbound dial-peer.

Examples

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

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

Related Commands

None

audio-prompt load

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

audio-prompt load *name*

Syntax Description

audio-prompt load	Initiates loading the selected audio file from Flash memory into RAM.
name	Indicates the location of the audio file that you want to be loaded from memory, Flash memory, or an FTP server. Presently, with Cisco IOS Release 11.3(6)NA2, the URL pointer refers to the directory where Flash memory is stored.

Defaults

None

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.1(5)XM2	The command was introduced for the Cisco AS5350 and CiscoAS5400.

Usage Guidelines

- The first time the IVR application plays a prompt, it reads it from the URL (or the specified location for the .au file, such as Flash or TFP) into RAM. Then it plays the script from RAM.
- The router is up and running, and uses the *clid_authen_collect* script. The script is playing the prompt from *flash:enter_destination.au*.
- An example of the sequence of events are :
  - When the first caller is asked to enter their account and PINs, the *enter_account.au* and *enter_pin.au* files are loaded into RAM from Flash memory.
  - When the next call comes in, these prompts are played from the RAM copy.
  - If all callers enter valid account numbers and PINs, then the *auth_failed.au* file is not loaded from Flash memory into RAM memory.

Examples

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

Related Commands

None

call application voice

To create and then call the application that will interact with the IVR feature, use the **call application voice** command. Enter the command and designate the name of the application that you created.

call application voice *name*

Syntax Description

name	Is used to specify which TCL application the system is to call, or use for the calls configured on the inbound dial-peer. Enter the name of the TCL script file and the pathname where the TCL scripts are stored. An abbreviated name can be configured to represent the full TCL application and pathname included in one word. For example, the application name "test" can be an alias for: **tftp://keyer/debit audio/**. Enter the pathname first, and then the script file name. Valid storage locations can be URL or TFTP server.

Defaults

None. The system will not call an IVR application with the dial-peer.

The system does not have a default behavior. Each variable must be defined for system interaction with the voice prompts.

Command Modes

Global configuration mode

Command History

Release	Modification
12.0(4)XH	This command was introduced.
12.1(5)XM2	The command was introduced for the Cisco AS5350 and CiscoAS5400.

Usage Guidelines

None

Examples

```
call application voice clid_4digits_npw_3_cli.tcl
tftp://keyer/username/clid_4digits_npw_3_cli.tcl
call application voice clid_4digits_npw_3_cli.tcl pin-len 4
call application voice clid_4digits_npw_3_cli.tcl retry-count 3
call application voice clid_4digits_npw_3_cli.tcl uid-len 10
```

Related Commands

Command	Description
call application voice load	Reloads the selected TCL script from the URL.
show call application voice	Dislpays a list of the voice applications that are configured.

call application voice load

To reload the selected TCL script from the URL, use the **call application voice load** command. The software checks the signature lock to ensure it is a Cisco supported TCL script.

Syntax Description

call application voice load <*name*>

name	Defines the TCL application to use for the call. Enter the name of the TCL application you want this dial peer to use.

Defaults

None. No TCL application is loaded.

Command Modes

Privileged EXEC command mode

Command History

Release	Modification
12.0(7)T	This command was introduced.
12.1(5)XM2	The command was introduced for the Cisco AS5350 and CiscoAS5400.

Usage Guidelines

Note If the TCL script does not have a valid Cisco supported signature, the software fails to load the script and generates the following error message:

```
00:02:54: %IVR-3-BAD_IVR_SIG: Script signature is invalid
```

Related Commands

Command	Description
call application voice	Creates and then calls the application that will interact with the IVR feature.
show call application voice	Dislpays a list of the voice applications that are configured.

show call application voice

To see a list of the voice applications that are configured, use the **show call application voice** command. A one-line summary of each application appears. The description defines the names of the audio files the script plays, the operation of the interrupt keys, what prompts are used, and the caller interaction. If this command is entered without entering the *summary* field, a detailed summary appears for the application named in the <*name*> field.

show call application voice *name* **[summary]**

Syntax Description

Field	Description
name	The name of the desired IVR application.
summary	Enter this field to display a one-line summary. A complete detailed description appears of the application if you enter the command without the **summary** keyword.

Defaults

None

Command Modes

Privileged EXEC (also called enable mode)

Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.1(5)XM2	The command was introduced for the Cisco AS5350 and CiscoAS5400.

Usage Guidelines

- If the name of a specific application is entered, the system supplies information about that application.
- If the *summary* field is included, a one-line summary appears about each application.
- If you enter the command without the **summary** field, a detailed description of the IVR application is displayed.

Example Detail Display

show call application voice clid_authen_collect

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

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

Related Commands

Command	Description
call application voice	Creates and then calls the application that will interact with the IVR feature.
call application voice load	Reloads the selected TCL script from the URL.

Debug Commands

This section describes new and modified **debug** commands associated with the Settlement feature. All other commands used with this feature are documented in the Cisco Release 12.1 command reference publications. All **debug** commands are EXEC commands.

- **debug voip ivr**

debug voip ivr

To debug the IVR application, use the **debug voip ivr** command. IVR debug messages appear when a call is actively handled by the IVR scripts. An **error** output only occurs if something is not working or an error condition has been raised. A **states** output supplies information about the current status of the IVR script and the different events that are occurring in that state.

debug voip ivr [states | error | all]

no debug voip ivr [states | error | all]

Syntax Description

states	(Optional) Displays verbose information about how IVR is handling each call.
error	(Optional) Only displays information if an error occurs.
all	(Optional) Displays both **states** and **error** messages.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.1(5)XM2	The command was introduced for the Cisco AS5350 and CiscoAS5400.

Usage Guidelines

- IVR debug messages appear when a call is actively handled by the IVR scripts.
- **Error** output only occurs if something is not working, or an error condition has been raised.
- **States** output supplies information about the current status of the IVR script and the different events which are occurring in that state.
- The keyword *settlement* was added for the "Settlement for Packet Telephony" feature in Cisco IOS Release 12.0(7)T. The activity of the settlement servers appear when you enter the *settlement* keyword.

Command Output

The display below shows output from Cisco IOS Release 12.0(7)T and interaction of IVR with the Settlement application. The TCL IVR application is using the TCL script *clid_authen_collect*.

```
01:00:44:App clid_authen_collect:Handling callID 10
01:00:44:callingNumber=4085210502, calledNumber=4085210304,
redirectNumber=
01:00:44:accountNumber=, finalDestFlag=0,
guid=abea.7a4d.4299.0017.0000.0000.0037.9c94
01:00:44:peer_tag=5
01:00:44:settlement_validate_token:cid(0xA), target=, tokenp=0x0
01:00:44:Accepting CallID=10
01:00:44:authenticate
01:00:44:    account=4085210502
01:00:44:    password=4085210304
01:00:44:cid( 10) running state get_account
01:00:44:ta_PromptCmd. CallID=10

01:00:44:pcapp CallID 10 got event CC_EV_CALL_HANDOFF
01:00:44:prompt and collect app got callID 10
01:00:44:    Playing prompt flash:enter_account.au
01:00:44:    Prompt interrupt enabled
01:00:44:    No return on play complete
01:00:44:    Not matching against dial plan
01:00:44:    Abort key is *    Termination key is #
01:00:44:    Matching against 1 patterns.
01:00:44:        Pattern .+
01:00:44:CallID 10 First Buf Play at 01:00:44.580 of
flash:enter_account.au
01:00:44:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:44:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:44:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=1
01:00:44:CallID 10 Play Stopped at 01:00:44.928
01:00:45:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:45:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:45:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=1
01:00:45:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:45:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:45:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=1
01:00:45:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:45:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:45:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=#
01:00:45:pcapp CallID 10 returning PCAPP_MATCHED. string=111
01:00:45:cid( 10) running state get_pin
01:00:45:ta_PromptCmd. CallID=10

01:00:45:pcapp CallID 10 got event CC_EV_CALL_HANDOFF
01:00:45:prompt and collect app got callID 10
01:00:45:    Playing prompt flash:enter_pin.au
01:00:45:    Prompt interrupt enabled
01:00:45:    No return on play complete
01:00:45:    Not matching against dial plan
01:00:45:    Abort key is *    Termination key is #
01:00:45:    Matching against 1 patterns.
01:00:45:        Pattern .+
01:00:45:CallID 10 First Buf Play at 01:00:45.356 of flash:enter_pin.au
01:00:48:CallID 10 Play Stopped at 01:00:48.920
01:00:50:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:50:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:50:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=2
01:00:50:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:50:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:50:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=2
```

```
01:00:50:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:50:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:50:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=2
01:00:51:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:51:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:51:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=#
01:00:51:pcapp CallID 10 returning PCAPP_MATCHED. string=222
01:00:51:cid( 10) running state authenticate
01:00:51:authenticate
01:00:51:    account=111
01:00:51:    password=222
01:00:51:cid( 10) running state authen_pass
01:00:51:cid( 10) running state get_dest
01:00:51:ta_PromptCmd. CallID=10

01:00:51:pcapp CallID 10 got event CC_EV_CALL_HANDOFF
01:00:51:prompt and collect app got callID 10
01:00:51:    Playing prompt flash:enter_destination.au
01:00:51:    Prompt interrupt enabled
01:00:51:    No return on play complete
01:00:51:    Matching against dial plan
01:00:51:    Abort key is *    Termination key is #
01:00:51:    Matching against 0 patterns.
01:00:51:CallID 10 First Buf Play at 01:00:51.040 of
flash:enter_destination.au
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:55:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=4
01:00:55:CallID 10 Play Stopped at 01:00:55.468
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:55:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=0
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:55:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=8
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:55:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:55:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=5
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:56:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=2
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:56:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=1
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:56:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=0
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:56:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=4
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:56:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=0
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT_BEGIN
01:00:56:pcapp CallID 10 event CC_EV_CALL_DIGIT_BEGIN ignored
01:00:56:pcapp CallID 10 got event CC_EV_CALL_DIGIT digit=8
01:00:56:pcapp CallID 10 returning PCAPP_MATCHED. string=4085210408
01:00:56:cid( 10) running state place_call
01:00:56:Placing call for callID 10 to destination=4085210408
01:00:56:placecall CallID 10 got event CC_EV_CALL_HANDOFF
01:00:56:placecall pc_setupPeer cid(10), destPat(4085210408),
matched(10), prefix(), peer(617BB5AC)
01:00:56:pcSettlementAuthorize:authorizing for callid=10 using
calling=4085210502, called=4085210408
```

```
01:00:56:pcSettlementAuthorize:sending authorize request type=1,
args=0x619B50D8, sct=0x619B62DC
01:00:56:placecall cid(10) state change PC_CS_INIT to
PC_CS_CALL_SETTLING
01:00:57:pcSettlementSetup:settlement_curr_dest=0, num_dest=1
01:00:57:pcSettlementGetDestination:callinfop=0x619B61A0, error=0,
credit_time=20
01:00:57:pcSettlementSetup:placing call through ip(129.5.21.1),
calling(4085210502), called(4085210408), digits(4085210408)
01:00:57:placecall cid(10) state change PC_CS_CALL_SETTLING to
PC_CS_CALL_SETTING
01:00:57:placecall CallID 11 got event CC_EV_CALL_ALERT
01:00:57:placecall cid(10) state change PC_CS_CALL_SETTING to
PC_CS_CONFERENCING_ALERT
01:00:57:placecall CallID 10 got event CC_EV_CONF_CREATE_DONE
01:00:57:placecall cid(10) state change PC_CS_CONFERENCING_ALERT to
PC_CS_CONFERENCED_ALERT
01:01:00:placecall CallID 11 got event CC_EV_CALL_CONNECTED
01:01:00:placecall CallID 10 returning PLACECALL_ACTIVE.
01:01:00:pCall(0x619A63EC), settlement_credit_time=20
01:01:00:cid( 10) running state active
01:01:00:Wait for 10 seconds
01:01:06:Wait for 3 secondsMedia Content:flash:beep.au
  URL:flash:beep.au
  fd=-1, sampleRate=8000, bitsPerSample=8, coding=5, dataLength=1901,
  numReaders=0, writePtr=0, bufSize=1905, refCount=1

01:01:09:mallocing 1905
01:01:09:Only read 1901 bytes
01:01:09:pcapp CallID 10 got event CC_EV_CALL_HANDOFF
01:01:09:prompt and collect app got callID 10
01:01:09:    Playing prompt flash:beep.au
01:01:09:    Prompt interrupt disabled
01:01:09:    Return on play complete
01:01:09:    Not matching against dial plan
01:01:09:    No abort key
01:01:09:    No termination key
01:01:09:    Matching against 0 patterns.
01:01:09:CallID 10 First Buf Play at 01:01:09.732 of flash:beep.au
01:01:10:CallID 10 Play Stopped at 01:01:10.976
01:01:10:pcapp CallID 10 returning N/A. string=
01:01:10:Wait for 10 secondsMedia Content:flash:out_of_time.au
  URL:flash:out_of_time.au
  fd=-1, sampleRate=8000, bitsPerSample=8, coding=5, dataLength=8002,
  numReaders=0, writePtr=0, bufSize=8006, refCount=1

01:01:20:cid(10) conference_cleanup:ignoring event CC_EV_CALL_DIGIT
01:01:20:cid( 10) running state out_of_time
01:01:20:ta_PromptCmd. CallID=10

01:01:20:mallocing 8006
01:01:20:Only read 8002 bytes
01:01:20:pcapp CallID 10 got event CC_EV_CALL_HANDOFF
01:01:20:prompt and collect app got callID 10
01:01:20:    Playing prompt flash:out_of_time.au
01:01:20:    Prompt interrupt disabled
01:01:20:    Return on play complete
01:01:20:    Not matching against dial plan
01:01:20:    No abort key
01:01:20:    No termination key
01:01:20:    Matching against 0 patterns.
01:01:20:CallID 10 First Buf Play at 01:01:20.996 of
flash:out_of_time.au
01:01:23:CallID 10 Play Stopped at 01:01:23.020
```

```
01:01:23:pcapp CallID 10 returning N/A. string=
01:01:23:TCL script eval for callID 10 completed. code=OK
```

Related Commands None

Glossary

AAA—Authentication, Authorization, and Accounting. AAA is a suite of network security services that provides the primary framework through which you can set up access control on your Cisco router or access server.

ANI—Automatic number identification. Same as calling party.

DNIS—Dialed number identification service. Same as the called number.

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

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

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

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

IFS—Cisco IOS File System.

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

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

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

TCL—Tool Command Language. TCL is an interpreted script language developed by Dr. John Ousterhout of the University of California, Berkeley, and is now developed and maintained by Sun Microsystems Laboratories.

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

Note For a list of other internetworking terms, see *Internetworking Terms and Acronyms* document that is available on the Documentation CD-ROM and Cisco Connection Online (CCO) at the following URL: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ita/index.htm.