Configuring Tcl IVR 2.0 Session Interaction

The Tcl IVR 2.0 Session Interaction feature allows different instances of Tcl IVR applications (sessions) to communicate with other sessions on the same gateway and for applications to dynamically bridge call legs between different sessions. This enables different callers on the same gateway to be notified of each others' presence and to interact. You can also start a session without an active call leg so that a session can act as an application service for other sessions. This feature is useful for implementing a call-monitoring "server" application that is responsible for monitoring incoming calls and dynamically connecting selected callers.

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

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

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

Feature History for Tcl IVR 2.0 Session Interaction

Release	Modification
12.3(4)T	This feature was introduced.
12.3(14)T	A new command-line interface structure for configuring Tcl and IVR applications was introduced and affected the commands for configuring this feature.

Contents

Prerequisites for Session Interaction

- Install Cisco IOS Release 12.3(4)T or later on the voice gateway.
- Configure basic Tcl IVR 2.0 functionality on the gateway. For information, see "Configuring Basic Functionality for Tcl IVR and VoiceXML Applications" on page 1.
- Write a Tcl IVR 2.0 script that implements the desired feature. For information, refer to the *Tcl IVR API Version 2.0 Programmer's Guide*.

Restrictions for Session Interaction

- Application instances can be started for Tcl IVR 2.0 applications only. Session interaction is not supported for VoiceXML applications or for Tcl 1.0 applications.
- A single Tcl IVR 2.0 session can handle a maximum of 12 objects at one time. Objects include a call leg, a digit collection operation, or a conference. If a session is full, it cannot set up a new call, initiate digit collection on a leg, set up a conference, or receive the handoff of a call leg. For example, if a session is handling eight call legs that are in four conferences, the session is full and another leg-setup command fails. Or, if a session is handling six call legs that are all running digit collection, the session is full.

Information About Session Interaction

To use the session interaction feature on Cisco gateways, you should understand the following concepts:

Tcl IVR 2.0 Session Interaction and Service Registry

A Tcl IVR 2.0 or VoiceXML application that is configured on a Cisco voice gateway is typically triggered by an incoming call. The application then delivers IVR services to the caller and can create and control one or more call legs. When a voice call invokes an application, it starts an instance, or session, of that application. The application instance executes the code of the application script and can place or transfer a call to a different application. A call can initiate one or more application instances, depending on how your system is configured. A single application instance can manage multiple voice calls.

In Cisco IOS Release 12.3(x)T, you can manually start an instance of a Tcl IVR 2.0 application on the gateway, without a call leg. This enables you to launch an application session on the gateway without requiring an incoming call. For example, you might write an application that monitors the status of a server group, to provide a keep-alive service. An instance of this application interacts with other application sessions by passing status information to other applications that are handling incoming calls. This type of service application runs on the gateway without being triggered by a call.

You can start an instance of a Tcl IVR 2.0 application on the gateway by using the application session configuration submode, or the **call application session start** command in privileged EXEC mode. For configuration information, see the "Starting a New Tcl IVR 2.0 Application Instance (Session)" section on page 3.

Note It is not possible to start an instance of a VoiceXML application because VoiceXML applications using Cisco IOS software require an active call leg to run.

Session interaction enables an application instance to communicate with other sessions on the same gateway and for calls to be bridged between different sessions. Multiple call legs and the ability of sessions to interact with each other fit the Tcl structure because of its strong call-control capabilities. Session interaction is not supported between VoiceXML sessions. For information about Tcl sessions interacting with VoiceXML sessions, refer to the "Hybrid Applications" section in the *Cisco VoiceXML Programmer's Guide*.

Tcl IVR 2.0 application instances can also register as a service. A services registry is essentially a database that keeps track of every application instance that registers as a service. Any other Tcl application can then find and communicate with any registered Tcl application. Registering a session as a service is done from within the Tcl script using a new Tcl verb. For information, refer to the *Tcl IVR API Version 2.0 Programmer's Guide*. You can display a list of application sessions that have registered as services by using the **show call application services registry** command.

Benefits of Session Interaction

- Supports presence-based applications by enabling Tcl IVR 2.0 application sessions to communicate with other sessions and to bridge calls between sessions on the same gateway. This allows different callers on the same gateway to be notified of each others' presence, and for those callers to interact.
- Allows a single application to act as a service for sessions that are handling incoming calls. A service can track which callers are active and which external servers are active, and maintain statistics for the application.

How to Configure Session Interaction

This section contains the following procedures:

- Starting a New Tcl IVR 2.0 Application Instance (Session), page 3 (required)
- Verifying That an Application Instance is Running, page 5 (optional)
- Stopping an Application Instance, page 7 (optional)

Starting a New Tcl IVR 2.0 Application Instance (Session)

There are two ways to start an application instance: from global configuration mode and from privileged EXEC mode. Starting the application session from global configuration mode starts the session and also adds the configuration to the gateway. Starting a session from privileged EXEC mode starts the session until the gateway reboots, or the session is stopped by the **call application session stop** command. The application session restarts at system reboot only if the **start** command is used in application session configuration submode.

Choose one of the following tasks, depending on whether you want to start the session as part of the gateway configuration or you want to start the instance temporarily, for instance during testing.

- Starting an Application Instance in the Configuration, page 4
- Starting an Application Instance in Privileged EXEC Mode, page 5

Starting an Application Instance in the Configuration

This section describes how to start a new instance of a Tcl IVR 2.0 application in the gateway configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** *application-name location*
5. **session**
6. **start** *instance-name application-name*
7. **end**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

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

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router(config)# configure terminal
```

Step 3 Load a Tcl script and specify its application name:

```
    application
      service application-name location

Example:
Router(config)# application
Router(config-app)# service my_app flash:demo1.vxml
```

Step 4 Start an instance of a Tcl application:

```
    application
      session
        start instance-name application-name
```

Use the name that was assigned to the application in Step 3.

```
Router(config)# application
Router(config-app)# session
Router(config-app-session)# start my_instance my_app
```

Step 5 Exit to privileged EXEC mode:

```
    end
```

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

Starting an Application Instance in Privileged EXEC Mode

This section describes how to start an application instance in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **service** *application-name location*
5. **exit**
6. **call application session start** *instance-name* [*application-name*]

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

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

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router(config)# configure terminal
```

Step 3 Load a Tcl script and specify its application name:

```
    application
    service application-name location
Router(config)# application
Router(config-app)# service my_app flash:demo1.vxml
```

Step 4 Exit global configuration mode and return to privileged EXEC mode:

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

Step 5 Start an instance of a Tcl application:

```
    call application session start instance-name [application-name]
```

You do not have to enter the application name if the application instance was previously started and stopped.

```
Example: Router# call application session start my_instance my_app
```

Verifying That an Application Instance is Running

SUMMARY STEPS

1. **show running-config**
2. **show call application sessions** [**callid call-id** | **id session-id** | **name instance-name**]

3. Follow the steps in the "Verifying Loading of Service" section on page 26.

DETAILED STEPS

Step 1 Use the **show running-config** command to verify that the application is configured on the gateway with the application configuration submodes, for example:

```
!
application
 service my_app tftp://10.10.1.1/sample.tcl
!
 session
  start my_instance my_app
!
!
```

Step 2 Use the **show call application sessions** command to verify that the application is running on the gateway. The following example shows output for the application named *serv1*:

```
Router# show call application sessions name serv1

Session named serv1 is in the start list in state running
  It is configured to start on GW reboot
  The application it runs is sample_service
  Handle is TCL_HAND*1653710732*0*3193204
TCL Session ID B
                 App: sample_service
                 URL: tftp://dev/demo/scripts/sample_service.tcl
        Session name: serv1
      Session handle: TCL_HAND*1653710732*0*3193204
           FSM State: start_state
    ID for 'show call active voice id' display: 0
                Legs:
            Services: data_service
```

Tip If the output shows that the application is in the "stopped" state, the script might have a syntax error that prevents it from running. The **show call application sessions** command does not display a stopped session if the script was started by using the **call application session start** command in privileged EXEC mode instead of global configuration mode.

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

Troubleshooting Tips

If the application instance does not successfully start on the gateway, see Table 10-1 for some possible causes and actions.

Table 10-1 Application Instance Does Not Start Running on Gateway

Possible Causes	Suggested Actions
Tcl script contains an error that prevents the script from running.	• Enable the **debug voip application error** and **debug voip application script** commands and then retry the **call application session start** command. These debug commands provide information about why the script cannot run. For example, an error message is displayed if the script contains bad syntax. • An error in the script can also prevent it from loading onto the gateway. With the above debug commands enabled, try reloading the script by using the **call application voice load** command. To verify the contents of the application script or document, use the **show call application voice** command.
Cisco gateway cannot access the external server to download the Tcl script.	Ping the corresponding server to make sure that the gateway has connectivity.
An application instance by the same name is already running.	Use the **show call application session** command to verify whether another instance with the same name is running. If you try to start a new session using the same name as a session that is currently running, the gateway displays the error message, "Cannot restart session instance *name*, it is running." You can start another instance for the same application by using a different name.

Stopping an Application Instance

There are two ways to stop an applications instance: from global configuration mode and from privileged EXEC mode. Stopping the application session from global configuration mode stops the session and removes the **call application session start** command from the gateway's configuration. Stopping a session from privileged EXEC mode stops the session until the gateway reboots, if the session is configured to start by the **call application session start** command in global configuration mode.

Choose one of the following tasks, depending on whether you want to stop this instance temporarily, for instance during testing, or you want to remove the session configuration from the gateway.

- Stopping an Application Instance in the Configuration, page 7
- Stopping an Application Instance in Privileged EXEC Mode, page 8

Stopping an Application Instance in the Configuration

This section describes how to stop an application instance and remove it from the gateway configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **application**
4. **session**
5. **no start** *instance-name*
6. **exit**

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

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

Enter your password if prompted.

Step 2 Enter global configuration mode:

```
    configure terminal
Example: Router(config)# configure terminal
```

Step 3 Stop an instance of a Tcl IVR application:

```
    application
      session
        no start instance-name
```

Note This command stops the instance of the application and removes the configuration from the gateway. VoiceXML sessions cannot be stopped with the **no session start** command because VoiceXML sessions cannot be started with Cisco IOS commands.

Tip Use the **show call application sessions** command to see a list of currently running instances.

Step 4 Exit global configuration mode and return to privileged EXEC mode:

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

Stopping an Application Instance in Privileged EXEC Mode

This section describes how to stop an application instance in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **call application session stop** {**callid** *call-id* | **handle** *handle* | **id** *session-id* | **name** *instance-name*}

DETAILED STEPS

Step 1 Enable privileged EXEC mode:

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

Enter your password if prompted.

Step 2 Stop an instance of a VoiceXML or Tcl IVR application:

```
    call application session stop {callid call-id | handle handle | id session-id | name
    instance-name}
```

Configuration Examples for Session Interaction

This section provides the following configuration example:

- Tcl IVR Application Sessions Example, page 9

Tcl IVR Application Sessions Example

In the following example, the Tcl IVR 2.0 script sample_service.tcl is configured to run as a session on the gateway without a call leg. The **call application voice** command loads the script onto the gateway and assigns it the application name sample_service. The **call application session start** command starts a legless session of the application on the gateway. The script registers itself as a service by using the Tcl service register command. As a service, the script simply keeps track of how many calls come into the gateway. When a call comes in, the script service_user.tcl, which is configured in the dial peer to handle the call, finds and exchanges a message with the service. The caller hears nothing. You can use the **debug voip ivr scripts** command to display output showing the process.

Note To see the contents of the Tcl scripts used in this example, refer to the *Tcl IVR API Version 2.0 Programmer's Guide*.

Gateway Configuration

```
!
version 12.2
service timestamps debug datetime msec localtime
service timestamps log datetime msec localtime
no service password-encryption
service internal
!
hostname sblab160
!
logging buffered 2000000 debugging
aaa new-model
!
!
aaa authentication login h323 local
aaa session-id common
enable password lab
!
username admin nopassword
username cisco nopassword
username 100 password 101
username 123 password 321
!
!
resource-pool disable
clock timezone pst -8
clock summer-time pdt recurring
!
ip subnet-zero
ip domain-name cisco.com
ip host rtsp-ws.cisco.com 1.7.153.4
ip host dirt 223.255.254.254
ip host px1-sun.cisco.com 1.7.100.1
ip host ts 1.7.100.1
ip host CALLGEN-SECURITY-V2 48.92.30.60 79.63.0.0
ip name-server 172.29.248.16
ip name-server 171.69.187.13
ip name-server 1.7.100.1
!
!
isdn switch-type primary-net5
!
!
!
!
!
no voice hpi capture buffer
no voice hpi capture destination
!
ivr prompt memory 16000
ivr prompt streamed http
http client cache refresh 2
fax interface-type modem
mta receive maximum-recipients 0
!
controller T1 0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 1
 framing esf
 clock source line primary
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 2
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
controller T1 3
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
!
!
interface Ethernet0
 ip address 1.7.102.39 255.255.0.0
 ip helper-address 223.255.254.254
 no ip redirects
 no ip route-cache
 no ip mroute-cache
 load-interval 30
!
interface Serial0:23
 no ip address
 no logging event link-status
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 no cdp enable
!
interface Serial1:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn protocol-emulate network
 isdn incoming-voice modem
 no isdn T309-enable
 isdn disconnect-cause 1
 no cdp enable
!
interface Serial2:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn protocol-emulate network
 isdn incoming-voice modem
 no isdn T309-enable
 isdn disconnect-cause 1
 no cdp enable
!
interface Serial3:23
 no ip address
 no logging event link-status
 isdn switch-type primary-ni
 isdn protocol-emulate network
 isdn incoming-voice modem
 no isdn T309-enable
 isdn disconnect-cause 1
 no cdp enable
!
interface FastEthernet0
 no ip address
 no ip route-cache
 no ip mroute-cache
 shutdown
 duplex auto
 speed auto
!
ip default-gateway 1.7.0.1
ip classless
ip route 223.255.254.0 255.255.255.0 1.7.0.1
no ip http server
ip pim bidir-enable
!
!
access-list 2 deny   1.1.1.1
access-list 2 permit any
!
!
call rsvp-sync
!
application
 service service_user tftp://dev/demo/scripts/service_user.tcl
!
 service sample_service tftp://dev/demo/scripts/sample_service.tcl
```

```
!
!
 session
  start serv1 sample_service
!
voice-port 0:D
!
voice-port 1:D
!
voice-port 2:D
!
voice-port 3:D
!
!
mgcp profile default
!
dial-peer cor custom
!
!
!
dial-peer voice 1 pots
 service service_user
 incoming called-number 52944
!
dial-peer voice 2 pots
service rate
service sessvars out-bound
 destination-pattern 12.
 port 2:D
!
dial-peer voice 190 voip
service k00
 destination-pattern 190
 session target ipv4:1.7.102.38
 incoming called-number 123
 dtmf-relay cisco-rtp
 codec g711ulaw
 no vad
!
dial-peer voice 3 pots
service rate
 shutdown
 destination-pattern 12.
 port 1:D
!
dial-peer voice 4 pots
service rate
 shutdown
 destination-pattern 12.
 port 3:D
!
dial-peer voice 7 voip
 destination-pattern .......
 session target ipv4:1.7.104.85
!
!
line con 0
 exec-timeout 0 0
 privilege level 15
 transport preferred none
line aux 0
 exec-timeout 0 0
 privilege level 15
line vty 0
```

```
 exec-timeout 0 0
 password 7 09404F0B
 notify
line vty 1 4
 exec-timeout 0 0
 privilege level 15
!
no scheduler max-task-time
end
```

Cisco IOS Command Output

Session Start Output

The following examples show output when the session is started:

```
Router# show call application sessions

TCL Sessions
     SID  Name        Called     Calling                App Name              Legs
       B  serv1                                         sample_service

VXML Sessions
     SID  Called                 Calling                App Name              Legs
    No running VXML sessions

Router# show call application session name serv1

Session named serv1 is in the start list in state running
  It is configured to start on GW reboot
  The application it runs is sample_service
  Handle is TCL_HAND*1653710732*0*3193204
TCL Session ID B
               App: sample_service
               URL: tftp://dev/demo/scripts/sample_service.tcl
      Session name: serv1
    Session handle: TCL_HAND*1653710732*0*3193204
         FSM State: start_state
    ID for 'show call active voice id' display: 0
              Legs:
          Services: data_service
```

Incoming Call Output

The following example shows output from the **show call application sessions** command listing the application service_user as active when a call comes into the gateway:

```
Router# show call application sessions

TCL Sessions
     SID  Name        Called     Calling                App Name              Legs
       B  serv1                                         sample_service
       C                  52944  8009673822             service_user         8

VXML Sessions
     SID  Called                 Calling                App Name              Legs
    No running VXML sessions
```

Tcl Puts Output

The following example shows output from the **debug voip application script** command, displaying the Tcl puts commands imbedded in the script, when a call comes into the gateway:

```
Router# debug voip application script
```

```
ivr script debugging is on

*May 28 13:17:04.779: //-1//TCL2:HN0030B974:/tcl_PutsCmd: Timer went off. Get data, and
reset the timer.

*May 28 13:17:04.779:
*May 28 13:17:08.139: //8//TCL2:/tcl_PutsCmd: sample_service_user.tcl is handling callid
8
*May 28 13:17:08.139:
*May 28 13:17:08.139: //-1//TCL2:HN0030B974:/tcl_PutsCmd: sample_service.tcl got a
message. Respond with the data.
*May 28 13:17:08.139:
*May 28 13:17:08.139: //-1//TCL2:HN0030B974:/tcl_PutsCmd: Send of data to
TCL_HAND*1672011776*0*3526560 returned success
*May 28 13:17:08.139:
*May 28 13:17:08.139: //-1//TCL2:HN0035CFA0:/tcl_PutsCmd: sample_service_user.tcl for
callid  8 got a response:
*May 28 13:17:08.143:
*May 28 13:17:08.143: //-1//TCL2:HN0035CFA0:/tcl_PutsCmd:      data(call_count)=1
*May 28 13:17:08.143:
*May 28 13:17:08.143: //-1//TCL2:HN0035CFA0:/tcl_PutsCmd:      data(run_time)=330
*May 28 13:17:08.143:

*May 28 13:17:34.779: //-1//TCL2:HN0030B974:/tcl_PutsCmd: Timer went off. Get data, and
reset the timer.

*May 28 13:17:34.779:
*May 28 13:17:36.227: //8//TCL2:/tcl_PutsCmd:
      Script for callids < 8> got event ev_disconnected
              Closing up now
```

Where to Go Next

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

Additional References

- , page 1—Describes how to access Cisco Feature Navigator; also lists and describes, by Cisco IOS release, Tcl IVR and VoiceXML features for that release
- Overview of Cisco IOS Tcl IVR and VoiceXML Applications, page 1—Describes underlying Cisco IOS Tcl IVR and VoiceXML technology; also lists related documents, standards, MIBs, RFCs, and how to obtain technical assistance