CHAPTER 1

Overview

This chapter provides an overview of Interactive Voice Response (IVR), the Tool Command Language (Tcl), and version 2.0 of the Tcl IVR Application Programming Interface (API). This section includes the following topics:

IVR and Tcl

IVR is a term used to describe systems that collect user input in response to recorded messages over telephone lines. User input can take the form of spoken words or, more commonly, dual tone multifrequency (DTMF) signaling.

For example, when a user makes a call with a debit card, an IVR application is used to prompt the caller to enter a specific type of information, such as a PIN. After playing the voice prompt, the IVR application collects the predetermined number of touch tones (digit collection), forwards the collected digits to a server for storage and retrieval, and then places the call to the destination phone or system. Call records can be kept and a variety of accounting functions can be performed.

The IVR application (or script) is a voice application designed to handle calls on a *voice gateway*, which is a router equipped with voice features and capabilities.

The prompts used in an IVR script can be either static or dynamic:

- *Static prompts* are audio files referenced by a static URL. The name of the audio file and its location are specified in the Tcl script.
- *Dynamic prompts* are formed by the underlying system assembling smaller audio prompts and playing them out in sequence. The script uses an API command with a notation form (see the media play, page 3-48) to instruct the system what to play. The underlying system then assembles a sequence of URLs, based on the language selected and audio file locations configured, and plays them in sequence. This provides simple Text-to-Speech (TTS) operations.

  For example, dynamic prompts are used to inform the caller of how much time is left in their debit account, such as:

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

Note The above prompt is created using eight individual prompt files. They are: youhave.au, 15.au, minutes.au, and.au, 30.au, 2.au, seconds.au, and leftinyouraccount.au. These audio files are assembled dynamically by the underlying system and played as a prompt based on the selected language and prompt file locations.

The Cisco Interactive Voice Response (IVR) feature, available in Cisco IOS Release 12.0(6)T and later, provides IVR capabilities using Tcl 1.0 scripts. These scripts are signature locked, and can be modified only by Cisco. The IVR feature allows IVR scripts to be used during call processing. Cisco IOS software to perform various call-related functions. Starting with Cisco IOS Release 12.1(3), no longer is any Tcl script lock in place, so customers can create and change their own Tcl scripts.

Tcl is an interpreted scripting language. Because Tcl is an interpreted language, scripts written in Tcl do not have to be compiled before they are executed. Tcl provides a fundamental command set, which allows for standard functions such as flow control (if, then, else) and variable management. By design, this command set can be expanded by adding extensions to the language to perform specific operations.

Cisco created a set of extensions, called Tcl IVR commands, that allows users to create IVR scripts using Tcl. Unlike other Tcl scripts, which are invoked from a shell, Tcl IVR scripts are invoked when a call comes into the gateway.

The remainder of this document assumes that you are familiar with Tcl and how to create scripts using Tcl. If you are not, we recommend that you read *Tcl and the TK Toolkit,* by John Ousterhout (published by Addison Wesley Longman, Inc).

Tcl IVR API Version 2.0

This section describes the prerequisites, restrictions, benefits, features, and the developer support program for this application programming interface.

Prerequisites

In order to use the open Tcl IVR feature, you need the following:

- Cisco AS5300, Cisco AS5400, or Cisco AS5800
- Cisco IOS Release 12.1(3)T, or later

- Tcl Version 7.1 or later

Calls can come into a gateway using analog lines, ISDN lines, a VoIP link, or a Voice over Frame Relay (VoFR) link. Tcl IVR scripts can provide full functionality for calls received over analog or ISDN lines.

The functionality provided for calls received over VoIP or VoFR links varies depending on the release of Cisco IOS software being used. For example, if you are using Cisco IOS Release 12.0, you cannot play prompts or tones, and you cannot collect tones.

Note Tcl IVR API Version 2.0 is a separate product from Tcl IVR API Version 1.0.

Benefits

Tcl IVR API Version 2.0 has the following benefits:

- The scripts are event-driven and the flow of the call is controlled by a Finite State Machine (FSM), which is defined by the Tcl script.
- Prompts can be played over VoIP call legs.
- Digits can be collected over VoIP call legs.
- Real-Time Streaming Protocol (RTSP)-based prompts are supported (depending on the release of Cisco IOS software and the platform).
- Scripts can control more than two legs simultaneously.
- Call legs can be handed off between scripts.
- All verbs are nonblocking, meaning that they can execute without causing the script to wait, which allows the script to perform multiple tasks at once. See the following example code:

```
leg collect digits 1 callInfo
leg collect digits 2 callInfo
leg setup 295786 setupInfo $callID5
puts "\n This will be executed immediately i.e. before the collect digits or call
setup is actually complete"
```

  In the preceding script example, digit collection is initiated on legs 1 and 2 and a call setup process is started using the callID5 as the incoming leg. The script has issued each of the commands and will later receive events regarding their completion. None of these commands ever requires that any other command wait until it is finished processing.

Features Supported

Tcl IVR API Version 2.0 commands provide access to the following facilities and features:

- Call handling (setup, conferencing, disconnect, and so forth)
- Media playout and control (both memory-based and RTSP-based prompts)
- AAA authentication and authorization
- OSP settlements
- Call and leg timers
- Play tones
- Call handoff and return
- Digit collection

For more information, see Chapter 3, "Tcl IVR Commands."

Developer Support

Developers using this guide may be interested in joining the Cisco Developer Support Program. This new program has been developed to provide you with a consistent level of support that you can depend on while leveraging Cisco interfaces in your development projects.

A signed Developer Support Agreement is required to participate in this program. For more details, and access to this agreement, visit us at: http://www.cisco.com/warp/public/779/servpro/programs/ecosystem/devsup, or contact developer-support@cisco.com

Enhanced MultiLanguage Support

Beginning with Cisco IOS Release 12.2(2)T, a new feature has been introduced into Tcl IVR Version 2.0 that provides support for adding new languages and text-to-speech (TTS) notations to the core IVR infrastructure of the Cisco IOS gateway.

In the past, if you wanted an IVR application to do text-to-speech, you were limited to English, Spanish, and Chinese languages, and a fixed set of TTS notations. If an IVR application wanted to support more languages, it needed to do its own translation and include the language translation procedures with every Tcl IVR application that needed it.

With this new feature, you can make a new Tcl language module for any language and any set of TTS notations. You can test and deliver the module, and the audio files that go with it, as a language package, then document the language it delivers and the TTS notations it supports. When you configure this module on the gateway, any IVR application running on that gateway and using those TTS notations would work and speak that language.

For more information, refer to the *Enhanced Multi-Language Support for Cisco IOS Interactive Voice Response* document.

Note Tcl language modules are not Tcl IVR scripts. They are pure Tcl scripts that implement a specific Tcl language module interface (TLMI), so they must not use the Tcl IVR API extensions that are available for writing IVR scripts.

VoiceXML and IVR Applications

VoiceXML brings the advantages of web-based development and content to IVR applications. For more discussion on using VoiceXML with IVR applications, refer to the *Cisco IOS Tcl and VoiceXML Application Guide* and the *Cisco VoiceXML Programmer's Guide*.

Call Handoff in Tcl

Call handoff can best be understood when the concept of an application instance is first understood. In the Cisco IOS IVR infrastructure, an application instance is an entity that executes the application code and receives, creates, and manages one or more call legs to form a call, or to deliver a service to the user. The application instance owns and controls these call legs and receives all events associated with them. Although there can be exceptions, applications typically use a single application instance to deliver the services of a single call. Tcl IVR applications, when executing, act as one or more application instances.

Call Handoff is a term used to describe the act of transferring complete control of a call leg from one application instance to another. When handed off, all future events associated with that call leg will be received and handled by the target application instance.

Handoff can happen in several different ways, depending on whether the call leg needs to return to the source application instance of the handoff operation or not. A normal handoff application operation is similar to a *goto* event, with no automatic memory of a return address. The target cannot return the leg back to the source instance.

The *call app* operation is similar to a function call. The application instance performing the *call app* operation is saved on a stack and the target application instance can do a handoff return operation that returns the call leg to application instance on the top of the stack.

When doing a handoff of a call leg, any legs that are conferenced to that call leg are also handed off, even if they are not explicitly specified. When doing a handoff or a handoff return operation, an application instance can pass parameters as argument strings. Call handoff can take place between any combination of VoiceXML and Tcl IVR 2.0 applications.

The call handoff functionality allows a developer to write applications that may want to interact with each other for various purposes. This may be to use or leverage functionality in existing applications or to modularize a larger application into smaller application segments and use the handoff mechanism to coordinate and communicate between them. There may be times when the application developer need to leverage the functionality and features of both VoiceXML and Tcl IVR 2.0 in their applications. This may also be another application of the handoff operation.

Though handoff operations provide a certain amount of flexibility in achieving modularity and application interaction, they are limited when it comes to sharing control over a call leg. Only one application instance is in total control of the call leg and will receive events, which can prove to be limiting in certain scenarios. So, when considering a choice of mechanism for implementing applications involving both Tcl IVR 2.0 and VoiceXML, it is recommended that developers also consider *hybrid scripting* as an alternative.

Hybrid applications differ from call handoff operations. Hybrid applications are written using Tcl IVR scripts with VoiceXML dialogs either embedded or invoked in them. The Tcl IVR scripts are used for call control and the VoiceXML script is used for dialog management and they all run as part of one application instance allowing for a certain level of shared control of the call leg. Hybrid scripting is discussed in more detail in a later section.

Call Handoff in VoiceXML

The call handoff functionality in Cisco VoiceXML implementation is similar to the call handoff initiated by the *handoff appl* and *handoff callappl verbs* in Tcl IVR 2.0. For a discussion of call handoff in VoiceXML implementations, refer to the *Cisco VoiceXML Programmer Guide*.

Tcl/VoiceXML Hybrid Applications

Tcl IVR 2.0 and VoiceXML APIs each have their own strong points and some weak points. Tcl IVR 2.0 is very flexible when it comes to call control, able to describe multiple call legs, how they should be controlled, and how they should interwork. A weak point, however, is when it comes to user interface primitives being limited to **leg collectdigits** and **media play** commands.

VoiceXML on the other hand is both familiar and easy to use to design voice user interfaces, but is very limited in its call control capabilities. For example, VoiceXML dialog is good at IVR activities, such as collecting user input or playing prompts.

It would be advantageous, therefore, to use Tcl IVR 2.0 to describe the call legs , and the call flow and call control interactions between them, while using VoiceXML to describe user interface dialogs on one or more of the legs it is controlling.

Though it may be possible, to a limited extent, to use the handoff mechanism to have separate application instances in Tcl IVR 2.0 and use VoiceXML to deal with the call control and dialog aspects of the application, it is difficult to clearly partition, in time, the call control and dialog activities. This requires that the call control script and the dialog execution share control over the call leg, which is difficult to do in the handoff approach.

Cisco IOS Release 12.2(11)T introduces the ability for developers to use Tcl and VoiceXML scripts to develop hybrid applications. Tcl IVR 2.0 extensions allow Tcl applications to leverage support for ASR and TTS by invoking and managing VoiceXML dialogs from within Tcl IVR scripts. Hybrid applications can be developed using Tcl IVR for call control and VoiceXML for dialog management, allowing applications to use both Tcl IVR 2.0 and VoiceXML APIs, yet behave as a single application instance.

Hybrid scripting requires that some control sharing and precedence rules be established. In hybrid applications, the Tcl IVR 2.0 script is in control of the call and all of its call legs. It receives *ev_setup_indication* events for incoming call legs, and has the primitives to issue a *leg alert* or to accept the call leg through a **leg connect** command. It also has the primitives and event support to create outgoing call legs, bridging one or more call legs together, or other similar operations.

When the Tcl IVR script wants to communicate with the user on one of the call legs, it has two ways to do this. It can use the existing **leg collectdigits** and **media play** commands in native Tcl IVR 2.0 to play individual audio prompts and collect digits, or it can use the **leg vxmldialog** command to initiate the VoiceXML dialog operation on the leg. The **leg vxmldialog** command starts up a VoiceXML interpreter session on the call leg under the direct control of the Tcl IVR 2.0 script. The initial VoiceXML document that the session starts up could either be embedded in the Tcl IVR 2.0 script invoking it or it can simply refer to a VoiceXML document on a web server.

This VoiceXML session started on the leg is a normal VoiceXML session for the most part, but with the following exceptions:

- There are some synchronization primitives and mechanisms that have been added to allow information exchange between the VoiceXML dialog session and the Tcl IVR 2.0 call control script.
- VoiceXML supports some call control commands, such as the <*transfer*> and <*disconnect*> tags, which behave differently in this mode because the Tcl IVR 2.0 script should have complete control of all call control activities.

Communication Between VoiceXML and Tcl IVR 2.0 in Hybrid Applications.

When the Tcl IVR 2.0 script initiates a VoiceXML dialog on a call leg, it can pass an array of parameters to the **leg vxmldialog** command. These parameters becomes accessible from within the VoiceXML session through the *com.cisco.params.xxxxxx* variables. In the VoiceXML session, the *com.cisco.params* object gets populated with a information from the Tcl IVR array, where *xxxxx* is the index of the Tcl array.

When the VoiceXML dialog finishes, it can return some information back to the Tcl IVR script through the *namelist* attribute of the <*exit/*> tag. When the VoiceXML dialog finishes executing, the Tcl script receives the *ev_vxmldialog_done* event, which can carry with it the information returned in the exit tag. The event also carries with it a status code, which can be accessed through the *evt_status* information tag.

Apart from the start and end of a VoiceXML dialog, the Tcl script can send an intermediate message to a dialog in progress through the **leg vxmlsend** command. The event specified in the command is thrown inside VoiceXML interpreter and can be caught by a <*catch*> handler looking for that event. The command can also have a Tcl parameter array, whose information is accessible inside the VoiceXML catch handler through a scoped *_message.params.xxxxxx* variable, similar to *com.cisco.params.xxxx* described above.

Similarly, the VoiceXML interpreter environment or the executing document can send events to the Tcl script at various points. These events arrive at the Tcl script as *ev_vxmldialog_event* events. An executing VoiceXML document can use an <*object*> extension with *classid="builtin://com.cisco.ivrscript.sendevent"* to send an explicit message, with associated parameter information, to the parent Tcl script. If the VoiceXML document executes certain tags, such as <*disconnect*> or <*transfer*>*,* in the hybrid mode, that results in the Tcl script receiving an *ev_vxmldialog_event* event implicitly.

An *ev_vxmldialog_done* event or *ev_vxmldialog_event* event can come with two pieces of information:

- A VoiceXML-specific event name to differentiate the various reasons for the *ev_vxmldialog_done* or *ev_vxmldialog_event* event, which is accessible through the *evt_vxmlevent* information tag. This event name is a string in the form of *vxml.**. This indicates that the event name could be from the VoiceXML interpreter environment (vxml.session.*) or from the dialog executing in the VoiceXML interpreter (vxml.dialog.*). Examples of environment-level messages are *vxml.session.complete*, to indicate normal completion of a dialog, or *vxml.session.transfer,* to indicate that the document tried to execute a <*transfer*> tag, which is not supported in this mode of operation. If the document throws a *error.badfetch* message which is not caught and this causes the dialog to complete, or if the document uses the <*object*> send tag to send Tcl an explicit message, *evt_vxmlevent* will contain a *vxml.dailog.** string.
- A parameter array of information that is accessible through the *evt_vxmlevent_params* information tag.

Hybrid Mode and VoiceXML Call Control Tags

In the hybrid mode, the VoiceXML <*disconnect*> tag does not disconnect the call leg. Instead, a *vxml.session.disconnect* event is sent to the Tcl IVR script. From a VoiceXML execution perspective, a <disconnect> is emulated, throwing a disconnect event and then continuing execution. The dialog will not be able to play prompts or collect input from this point onwards.

When the user hangs up, a <disconnect> is again emulated, as above. But the leg is not disconnected yet. The Tcl script receives the *ev_disconnected* event as part of the control events, then has the responsibility of either terminating, or waiting for the termination of the dialog, and then disconnecting the leg.

When the document executes a <*transfer*> tag, this results in the following:

- A *vxml.session.transfer* event is sent by the VoiceXML environment to the Tcl script.
- The VoiceXML environment will throw an *error.unsupported.transfer* event at the VoiceXML session, which can be caught. If not caught, the default handler causes the termination of the dialog, resulting in an eventual *ev_vxmldialog_done* event to the Tcl script.

SendEvent Object

Recorded objects are represented as audio object variables in VoiceXML/JAVA scripting. In Tcl, which is totally text based, objects are represented as a *ram://XXXXX* URI. Tcl array elements that have a value of *ram://XXX* are available as audio variables or objects in VoiceXML. A similar reverse transformation happens when information is passed from VoiceXML to the Tcl script.

Tcl IVR Call Transfer Overview

Tcl IVR scripts can be used to provide blind- and consultation-transfer support for a variety of call transfer protocols. This section provides some background information about call-transfer terminology and usage scenarios as related to Tcl IVR applications. It also describes the call-transfer capabilities of each supported protocol and how these protocols can be interworked when the endpoints involved in the transfer use different signaling protocols.

Call Transfer Terminology

Transfer participants

A call transfer typically involves three participants:

- Transferor (XOR)—The endpoint that initiates the transfer.
- Transferee (XEE)—The endpoint that is transferred to different destination.
- Transfer target (XTO)—The endpoint that the transferee is transferred to.

Transfer Trigger

A call-transfer trigger is the mechanism a transferor endpoint uses to initiate a call-transfer procedure. This is normally a hookflash event for analog phones, or a button or softkey on an IP phone registered with the Cisco IOS voice gateway operating in Cisco CallTw55tieManager Express (CME) mode.

Transfer Commit

A transfer commit is the action a transferor endpoint takes when it wants to connect the transferee and transfer target endpoints, possibly after consulting with the transfer target endpoint. For analog phones and Cisco CME IP phones, the transfer commit is usually performed by hanging up the phone. When a Tcl IVR script receives a transfer-commit indication, it normally attempts to send a transfer request to the transferee call leg.

Supported Tcl IVR Call Transfer Script

Cisco provides an official Tcl IVR script that supports the H.450 call transfer scenarios discussed in the remainder of this section. This script is available in the Cisco CallManager Express (CME) zip files found at http://www.cisco.com/cgi-bin/tablebuild.pl/ip-key. The current version of the script is named *app_h450_transfer.2.0.0.3.tcl*. Refer to the README file associated with the script for more details.

Call Transfer Support in the Cisco IOS Default Session Application

Call transfer support has been added to the default voice session application in the 12.2(15)ZJ Cisco IOS release. The default application now provides H.450 and SIP transferee and transfer target functionality for blind and consultation transfers. It also provides H.450 and SIP blind and consultation transferor support for IP phones connected to the Cisco IOS gateway while operating in Cisco CallManager Express (CME) mode.

Note The enhanced default session application does not provide support for transfer initiation using an analog phone connected to the Cisco IOS gateway. This functionality is provided in the app_h450_transfer.2.0.0.3.tcl script mentioned above or can be implemented in a custom Tcl IVR application.

Custom Tcl IVR Call Transfer Scripts

The Cisco IOS default session application and app_h450_transfer.tcl script described above can be used to support many typical call transfer scenarios. In cases where a variant of this functionality is needed, a custom Tcl IVR script can be written. The call-transfer-sample.zip file on the Developer Support Central page contains sample Tcl IVR scripts and associated documentation that can be used as a guide in writing such a script.

Further assistance in Tcl IVR script writing can be obtained by joining the Cisco Developer Support program. This program provides a central resource for all development needs. Members of the program gain access to all available product and documentation downloads, bug reports, sample scripts, and frequently asked questions to facilitate development efforts.

The Developer Support engineers have subject matter expertise in Cisco interfaces and protocols. This team is dedicated to helping customers, and Cisco AVVID Partner Program and other ecosystem members, to use Cisco application programming interfaces (APIs) in their development projects. In addition to the benefits accessed from Cisco.com, the program provides an easy process to open, update, and track issues through Cisco.com. The Developer Support Agreement, which defines support commitments, fees, and available options, can be obtained from the Cisco Developer Support Web site at http://www.cisco.com/warp/public/570/.

Call Transfer Scenarios

There are many call transfer scenarios to consider when writing a Tcl IVR script. This subsection describes several such scenarios involving one, two, or three Cisco IOS voice gateways. To illustrate the call transfer scenarios, each description that follows includes the following diagrams:

- The first diagram shows the two-party call before the transfer.
- The second diagram shows a blind call transfer in progress.

- The third diagram shows a consultation transfer in progress.
- The fourth diagram shows the final call after a successful blind or consultation transfer.

Depending on the specific requirements, a script can be written to provide support for one or more of the scenarios that follow. In some cases, such as the consultation transfer scenario shown in Figure 1-7, two independent instances of the script may be active on the same gateway.

In the figures that follow, the labels XOR, XEE, and XTO designate the role each call leg plays in the call transfer. The IN and OUT labels track the incoming and outgoing call legs during a two-party call. This allows a script to keep track of the call leg topology and determine what action to take when an event is received.

In all scenarios described here, the original two-party call between phone A and phone B is already established. Phone A is the transferor endpoint (XOR), phone B is the transferee endpoint (XEE), and phone C is the transfer target endpoint (XTO). Transferor phone A is either an analog FXS phone or an IP phone registered with the Cisco IOS voice gateway operating in Cisco CallManager Express (CME) mode.

One Gateway Scenario with Analog Transferor

The first call transfer scenario is one in which phones A, B, and C are connected to the same gateway, as shown in Figure 1-1. In this case, all transferor, transferee, and transfer-target functionality is provided by a single instance of the Tcl IVR script.

Figure 1-1 Single GW: Analog XOR Before Transfer

To initiate a blind transfer, the analog phone user presses hookflash, enters the transfer destination, and then hangs up. The script then places a regular call to the transfer target, connects the transferee and transfer-target call legs, then disconnects the transferor call leg. See Figure 1-2.

Figure 1-2 Single GW: Analog XOR Blind Transfer

To initiate a consultation transfer, the analog phone user presses hookflash and enters the transfer destination number. The script then places a call to the transfer target so that phone A can consult with phone C. When the user commits the transfer (by hanging up), the script requests a consultation ID from the transfer target (phone C). Because phone C is a local analog phone, the gateway generates a local consultation ID and registers it to this script instance. The script then places the outbound transfer call to phone C that includes this consultation ID. Because the consultation ID is registered to this script instance, the transferee call leg is handed off to this same script. See Figure 1-3.

Figure 1-3 Single GW: Analog XOR Consultation Transfer

When the script receives the handoff event, it bridges the transferee and transfer-target legs and releases the transferor. See Figure 1-4.

Note In this single gateway scenario, it would be possible to simplify the call flow and avoid having the script hand off the transferee call leg to itself; however, using the handoff mechanism is the preferred approach as it also works in the multi-gateway scenarios described below.

Figure 1-4 Single GW: Analog XOR After Transfer

One Gateway Scenario with Cisco CME IP Phone Transferor

In this transfer scenario, phones A, B, and C are all connected to the same gateway. See Figure 1-5. In this case the transferor, transferee, and transfer-target functionality is provided by one or two instances of the Tcl IVR script.

Figure 1-5 Single GW: Cisco CME IP Phone XOR Before Transfer

To initiate a blind transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination. The script receives a transfer request event from the phone, places a regular call to the transfer target, and connects the transferee and transfer target call legs. It then disconnects the transferor call leg. See Figure 1-6.

Figure 1-6 Single GW: Cisco CME IP Phone XOR Blind Transfer

To initiate a consultation transfer, the IP phone user presses the transfer button on the phone and enters the transfer-destination number. The IP phone uses a separate line to place a call to the transfer target. This call is independently handled by a second instance of the Tcl IVR script running on the gateway, which treats the call as a normal two-party call, unaware it is a consultation call.

After consulting with the transfer target, the user commits the transfer by hanging up and the second script instance receives a consultation request from IP phone A. Because phone C is a local analog phone, the gateway generates a local consultation ID and registers it to this script instance. The script then sends a consultation response to IP phone A that includes this consultation ID. Next, the first script instance receives a transfer request from IP phone A that includes the consultation ID it received from the second script instance. See Figure 1-7.

Figure 1-7 Single GW: Cisco CME IP Phone XOR Consultation Transfer

This script instance then places the outbound transfer call to phone C that includes the consultation ID. Because the consultation ID is registered to the second script instance, the transferee call leg is handed off to the second script instance. The second script instance receives the handoff event and bridges the transferee and transfer-target legs. The first script instance releases the transferor call leg. See Figure 1-8.

Figure 1-8 Single GW: Cisco CME IP Phone XOR After Transfer

Two Gateway Scenarios with Analog Transferor

There are several call transfer scenarios that involve two Cisco IOS gateways and an analog transferor. Several of these are described in the following subsections.

XOR and XTO on Gateway 1 and XEE on Gateway 2

In the first scenario, the transferor (phone A) and transfer-target (phone C) endpoints are connected to Gateway 1. The transferee endpoint (phone B) is connected to Gateway 2. See Figure 1-9.

Figure 1-9 Two Gateways (XOR/XTO & XEE): Analog XOR Before Transfer

To initiate a blind transfer, the analog phone user presses hookflash, enters the transfer destination, then hangs up. The script on Gateway 1 sends a SIP or H.450 transfer request to phone B. The transfer request is received by the script handling the call between phone A and phone B on Gateway 2. This script places an outbound call to phone C and disconnects its transferor call leg when the call setup succeeds. See Figure 1-10.

Although phone C is also connected to Gateway 1, the incoming call from phone B to phone C is handled by a separate instance of the Tcl IVR script. This script simply places a normal call to phone C, without knowledge that this call was part of a call transfer.

Figure 1-10 Two Gateways (XOR/XTO & XEE): Analog XOR Blind Transfer

To initiate a consultation transfer, the analog phone user presses hookflash and enters the transfer destination number. The script then places a call to the transfer target so that phone A can consult with phone C. When the user commits the transfer (by hanging up), the script requests a consultation ID from the transfer target (phone C). Because phone C is a local analog phone, the gateway generates a local consultation ID and registers it to this script instance. The script then sends a SIP or H.450 transfer request to phone B that includes the consultation ID. The transfer request is received by the script handling the call on Gateway 2. This script places an outbound call to phone C and disconnects its transferor call leg when the call setup succeeds. See Figure 1-11.

Figure 1-11 Two Gateways (XOR/XTO & XEE): Analog XOR Consult Transfer

The setup request includes the consultation ID received in the transfer request. Unlike the blind transfer case above, the incoming setup request to phone C is handled by the same instance of the script that is handling the original call between phones A and B, and the consultation call between phones A and C. This script connects the incoming call to phone C and disconnects phone A. See Figure 1-12.

Figure 1-12 Two Gateways (XOR/XTO & XEE): Analog XOR After Transfer

XOR and XEE on Gateway 1 and XTO on Gateway 2

In this scenario, the transferor (phone A) and transferee (phone B) are connected to Gateway 1. The transfer target (phone C) is connected to Gateway 2. See Figure 1-13.

Figure 1-13 Two Gateways (XOR/XEE & XTO): Analog XOR Before Transfer

To initiate a blind transfer, the analog phone user presses hookflash, enters the transfer destination, and then hangs up. The script places a call to phone C by sending a SIP or H.323 setup request to Gateway 2. The script that handles this setup request on Gateway 2 places a normal call to phone C, unaware that this call was part of a call transfer. After a successful call setup, the script on Gateway 1 bridges phone B and phone C and releases the call from phone A. See Figure 1-14.

Figure 1-14 Two Gateways (XOR/XEE & XTO): Analog XOR Blind Transfer

To initiate a consultation transfer, the analog phone user presses hookflash and enters the transfer destination number. The script then places a call to the transfer target so that phone A is able to consult with phone C. When the user commits the transfer (by hanging up), the script requests a consultation ID from the transfer target (phone C).

For H.450 transfers, Gateway 1 sends an H.450 consultation request message to phone C. This request is received by the script instance on Gateway 2 that is handling the call between phones A and C. This script sends a consultation response that includes a consultation ID. See Figure 1-15.

For SIP, the consultation request is not sent to phone C. Instead, a consultation ID is generated locally by Gateway 1. In both cases, when the script on Gateway 1 receives the consultation response, it sends a SIP or H.450 setup request to Gateway 2 that includes this consultation ID. When the setup request arrives at Gateway 2, it is delivered to the same script instance that is handling the consultation call between phone A and phone C.

Figure 1-15 Two Gateways (XOR/XEE & XTO): Analog XOR Consultation Transfer

This script connects the incoming call to phone C and disconnects the consultation call from phone A. See Figure 1-16.

Figure 1-16 Two Gateways (XOR/XEE & XTO): Analog XOR After Transfer

XOR on Gateway 1 and XEE and XTO on Gateway 2

The third call transfer scenario involving two gateways is shown in Figure 1-17. The transferor (phone A) is connected to Gateway 1, and the transferee (phone B) and transfer target (phone C) are connected to Gateway 2.

Figure 1-17 Two gateways (XOR & XEE/XTO): Analog XOR Before Transfer

To initiate a blind transfer, the analog phone user presses hookflash, enters the transfer destination, then hangs up. The script on Gateway 1 sends a SIP or H.450 transfer request to phone B. The transfer request is received by the script handling the call between phone A and phone B on Gateway 2. This script places an outbound call to phone C. When the setup succeeds, this script connects phone B to phone C and disconnects the call from phone A. See Figure 1-18.

Figure 1-18 Two Gateways (XOR & XEE/XTO): Analog XOR Blind Transfer

To initiate a consultation transfer, the analog phone user presses hookflash and enters the transfer destination number. The script then places a call to the transfer target so that phone A can consult with phone C. The incoming call from phone A is handled by a different script instance on Gateway 2 than is handling the call between phones A and B. See Figure 1-19.

When the user commits the transfer (by hanging up), the script on Gateway 1 requests a consultation ID from the transfer target. For H.450 transfers, Gateway 1 sends an H.450 consultation request message to phone C. The request is received by the script instance on Gateway 2 that is handling the call between phones A and C. This script sends a consultation response that includes a consultation ID.

For SIP, the consultation request is not sent to phone C. Instead, a consultation ID is generated locally by Gateway 1. In both cases, when the script on Gateway 1 receives the consultation response, it sends a SIP or H.450 transfer request to Gateway 2 that includes this consultation ID.

Figure 1-19 Two Gateways (XOR & XEE/XTO): Analog XOR Consultation Transfer

The transfer request is received by the script instance handling the call between phones A and B on Gateway 2. This script places a call to phone C. The setup request includes the consultation ID received in the transfer request. Because the consultation ID included in the setup request matches the one sent to Gateway 1 in the consultation response, the call setup completes by handing off the incoming call to the second script instance. After the handoff, the original call from phone A to phone B is disconnected by the first script instance on Gateway 2 and the consultation call from phone A is disconnected by the second script instance. See Figure 1-20.

Figure 1-20 Two Gateways (XOR & XEE/XTO): Analog XOR After Transfer

Two Gateway Scenarios with Cisco CME IP Phone Transferor

There are several call transfer scenarios that involve two Cisco IOS gateways and a Cisco CallManager Express (CME) IP phone transferor. Several of these are described in the following subsections.

XOR and XTO on Gateway 1 and XEE on Gateway 2

The first scenario is shown in Figure 1-21. Here, the transferor (phone A) and transfer-target (phone C) endpoints are connected to Gateway 1. The transferee endpoint (phone B) is connected to Gateway 2.

Figure 1-21 Two Gateways (XOR/XTO & XEE): Cisco CME IP Phone XOR Before Transfer

To initiate a blind transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination. The script receives a transfer-request event from the phone and sends a SIP or H.450 transfer request to phone B. The transfer request is received by the script handling the call between phone A and phone B on Gateway 2. This script places an outbound call to phone C and disconnects its transferor call leg when the call setup succeeds. Although phone C is also connected to Gateway 1, the incoming call from phone B to phone C is handled by a separate instance of the Tcl IVR script. This script simply places a normal call to phone C without knowledge that this call was part of a call transfer. See Figure 1-22.

Figure 1-22 Two Gateways (XOR/XTO & XEE): Cisco CME IP Phone XOR Blind Transfer

To initiate a consultation transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination number. The IP phone uses a separate line to place a call to the transfer target. This call is independently handled by a second instance of the Tcl IVR script running on Gateway 1. The script instance treats the call as a normal two-party call, unaware that it is a consultation call. See Figure 1-23.

After consulting with the transfer target, the user commits the transfer by hanging up and the second script instance receives a consultation request from IP phone A. Because phone C is a local analog phone, the gateway generates a local consultation ID and registers it to this script instance. The script then sends a consultation response to IP phone A that includes this consultation ID.

Next, the first script instance receives a transfer request from IP phone A that includes the consultation ID it received from the second script instance. This script instance then sends a SIP or H.450 transfer request to phone B that includes the consultation ID. The transfer request is received by the script handling the call between phone A and phone B on Gateway 2. This script places an outbound call to phone C and disconnects its transferor call leg when the call setup succeeds. The setup request includes the consultation ID received in the transfer request.

Figure 1-23 Two Gateways (XOR/XTO & XEE): Cisco CME IP Phone XOR Consult Transfer

The incoming setup request is delivered to the script instance on Gateway 1 that is handling the consultation call between phone A and phone C. This script connects the incoming call to phone C and releases the call from phone A. See Figure 1-24.

Figure 1-24 Two Gateways (XOR/XTO & XEE): Cisco CME IP Phone XOR After Transfer

XOR and XEE on Gateway 1 and XTO on Gateway 2

The second scenario involving two gateways and an IP phone transferor. The transferor (phone A) and transferee (phone B) are connected to Gateway 1. The transfer target (phone C) is connected to Gateway 2. See Figure 1-25.

Figure 1-25 Two Gateways (XOR/XEE & XTO): Cisco CME IP Phone XOR Before Transfer

To initiate a blind transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination. The script receives a transfer request event from the phone A and places a call to phone C by sending a SIP or H.323 setup request to Gateway 2. The script that handles this setup request on Gateway 2 places a normal call to phone C, unaware that this call was part of a call transfer. After a successful call setup, the script on Gateway 1 bridges phone B and phone C and releases the call from phone A. See Figure 1-26.

Figure 1-26 Two Gateways (XOR/XEE & XTO): Cisco CME IP Phone XOR Blind Transfer

To initiate a consultation transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination number. The IP phone uses a separate line to place a call to the transfer target. This call is independently handled by a second instance of the Tcl IVR script running on Gateway 1. The script instance treats this as a normal two-party call and is not aware it is a consultation call.

After consulting with the transfer target, the user commits the transfer by hanging up and the second script instance receives a consultation request from IP phone A. For H.450 transfers, Gateway 1 relays this consultation request to phone C by sending an H.450 consultation request message to phone C. The request is received by the script instance on Gateway 2 handling the call between phones A and C. This script sends a consultation response that includes a consultation ID. See Figure 1-27.

For SIP, the consultation request is not relayed to phone C. Instead, a consultation ID is generated locally by Gateway 1. In both cases, when the script on Gateway 1 receives the consultation response, it relays it to IP phone A. In addition, due to the internal consultation ID management scheme in the Cisco IOS application framework, the consultation ID received from Gateway 2 is registered to this script instance (the second instance).

Note Because the script instance on Gateway 2 sent a consultation response to Gateway 1, it expects to receive an incoming call from the transferee. Because the transfer was handled locally on Gateway 1 through a handoff, Gateway 2 will not receive this incoming call. A guard timer in Cisco IOS eventually expires, and the script continues processing the call between Phone A and phone C as a normal two-party call.

Next, the first script instance receives a transfer request from IP phone A that includes the consultation ID from the second script instance. This script instance places the outbound call to phone C that includes the consultation ID.

Figure 1-27 Two Gateways (XOR/XEE & XTO): Cisco CME IP Phone XOR Consultation Transfer

Because the consultation ID is registered to the second script instance, the transferee call leg is handed off to the second script instance. This script instance receives the handoff event and bridges the transferee and transfer target legs. The first script instance releases the transferor call leg. See Figure 1-28.

Figure 1-28 Two Gateways (XOR/XEE & XTO): Cisco CME IP Phone XOR After Transfer

XOR on Gateway 1 and XEE and XTO on Gateway 2

The third call transfer scenario involving two gateways and an IP phone transferor is shown in Figure 1-29. The transferor (phone A) is connected to Gateway 1, and the transferee (phone B) and transfer target (phone C) are connected to Gateway 2.

Figure 1-29 Two Gateways (XOR & XEE/XTO): Cisco CME IP Phone XOR Before Transfer

To initiate a blind transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination. The script receives a transfer request event from the phone and sends a SIP or H.450 transfer request to phone B. The transfer request is received by the script handling the call between phone A and phone B on Gateway 2. This script places an outbound call to phone C. After a successful call setup, the script on Gateway 2 bridges phone B and phone C and releases the call from phone A. See Figure 1-30.

Figure 1-30 Two Gateways (XOR & XEE/XTO): Cisco CME IP Phone XOR Blind Transfer

To initiate a consultation transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination number. The IP phone uses a separate line to place a call to the transfer target. The call is independently handled by a second instance of the Tcl IVR script running on Gateway 1. This script instance treats the call as a normal two-party call and is not aware it is a consultation call.

After consulting with the transfer target, the user commits the transfer by hanging up and the second script instance receives a consultation request from IP phone A. For H.450 transfers, Gateway 1 relays this consultation request to phone C by sending an H.450 consultation request message to phone C. The request is received by the script instance on Gateway 2 that is handling the call between phones A and C. This script sends a consultation response that includes a consultation ID. For SIP, the consultation request is not relayed to phone C. Instead, a consultation ID is generated locally by Gateway 1. In both cases, when the script on Gateway 1 receives the consultation response, it relays it to IP phone A. In addition, due to the internal consultation ID management scheme in the Cisco IOS application framework, the consultation ID received from Gateway 2 is registered to this script instance (the second instance).

Next, the first script instance on Gateway 1 receives a transfer request from IP phone A that includes the consultation ID it received from the second script instance on Gateway 1. The script instance then sends a SIP or H.450 transfer request to phone B that includes this consultation ID. The transfer request is received by the script instance handling the call between phones A and B on Gateway 2. This script places a call to phone C. Because the consultation ID included in the setup request matches the one sent to Gateway 1 in the consultation response, the call setup is completed by handing off the incoming call to the second script instance. See Figure 1-31.

Figure 1-31 Two Gateways (XOR & XEE/XTO): Cisco CME IP Phone XOR Consultation Transfer

After the handoff, the original call from phone A to phone B is disconnected by the first script instance on Gateway 2 and the consultation call from phone A is disconnected by the second script instance. See Figure 1-32.

Figure 1-32 Two Gateways (XOR & XEE/XTO): Cisco CME IP Phone XOR After Transfer

Three Gateway Scenario with Analog Transferor

Figure 1-33 shows a scenario where three gateways are involved in the call transfer. Each call transfer participant is connected to a separate Cisco IOS gateway.

Figure 1-33 Three Gateways: Analog XOR Before Transfer

To initiate a blind transfer, the analog phone user presses hookflash, enters the transfer destination, then hangs up. The script on Gateway 1 sends a SIP or H.450 transfer request to phone B. The transfer request is received by the script handling the call on Gateway 2. This script places a regular outbound call to phone C. The script that receives the incoming call setup on Gateway 3 treats this as a normal two-party call. When the setup completes, the script on Gateway 2 sends a transfer response to phone A. The script on Gateway 1 receives the transfer response and releases the call from phone A. See Figure 1-34.

Figure 1-34 Three Gateways: Analog XOR Blind Transfer

To initiate a consultation transfer, the analog phone user presses hookflash and enters the transfer destination number. The script then places a call to the transfer target so that phone A can consult with phone C. When the user commits the transfer (by hanging up), the script requests a consultation ID from the transfer target (phone C). For H.450 call transfers, a consultation request protocol message is sent to phone C. This request is received by the script instance on Gateway 3 that is handling the call between phones A and C. The script sends a consultation response that includes a consultation ID. See Figure 1-35.

For SIP, the consultation request is not sent to phone C. Instead, a consultation ID is generated locally by Gateway 1. In both cases, when the script on Gateway 1 receives the consultation response, it sends a SIP or H.450 transfer request to phone B that includes the consultation ID.

This transfer request is received by the script handling the call between phones A and B on Gateway 2. This script places a call to phone C. The setup request includes the consultation ID received in the transfer request from phone A. When the incoming setup request from phone B arrives at Gateway 2, it is delivered to the script instance handling the call between phones A and C.

Figure 1-35 Three Gateways: Analog XOR Consultation Transfer

This script instance connects the incoming call to phone C and disconnects the call from phone A. See Figure 1-36.

Figure 1-36 Three Gateways: Analog XOR After Transfer

Three Gateway Scenario with Cisco CME IP Phone Transferor

Figure 1-37 shows a scenario where three gateways are involved in the call transfer. Each call transfer participant is connected to a separate Cisco IOS gateway.

Figure 1-37 Three Gateways: Cisco CME IP Phone XOR Before Transfer

To initiate a blind transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination. The script receives a transfer request event from the phone and sends a SIP or H.450 transfer request to phone B. The transfer request is received by the script handling the call on Gateway 2. This script places a regular outbound call to phone C. The script that receives the incoming call setup on Gateway 3 treats this as a normal two-party call. When the setup completes, the script on Gateway 2 sends a transfer response to phone A. The script on Gateway 1 receives the transfer response and releases the call from phone A. See Figure 1-38.

Figure 1-38 Three Gateways: Cisco CME IP Phone XOR Blind Transfer

To initiate a consultation transfer, the IP phone user presses the transfer button on the phone and enters the transfer destination number. The IP phone uses a separate line to place a call to the transfer target. This call is independently handled by a second instance of the Tcl IVR script running on Gateway 1. The script instance treats this call as a normal two-party call and is not aware it is a consultation call. See Figure 1-39.

After consulting with the transfer target, the user commits the transfer by hanging up and the second script instance receives a consultation request from IP phone A. For H.450 transfers, Gateway 1 relays this consultation request to phone C by sending an H.450 consultation request message to phone C. The request is received by the script instance on Gateway 3 that is handling the call between phones A and C. This script sends a consultation response that includes a consultation ID.

For SIP, the consultation request is not relayed to phone C. Instead, a consultation ID is generated locally by Gateway 1. In both cases, when the script on Gateway 1 receives the consultation response, it relays it to IP phone A. In addition, due to the internal consultation ID management scheme in the Cisco IOS application framework, the consultation ID received from Gateway 2 is registered to this script instance (the second instance).

Next, the first script instance on Gateway 1 receives a transfer request from IP phone A that includes the consultation ID it received from the second script instance. This script instance then sends a SIP or H.450 transfer request to phone B that includes this consultation ID.

The transfer request is received by the script instance handling the call between phones A and B on Gateway 2. This script places a call to phone C. The setup request includes the consultation ID received in the transfer request from phone A. When the incoming setup request from phone B arrives at Gateway 3, it is delivered to the script instance handling the call between phones A and C.

Figure 1-39 Three Gateways: Cisco CME IP Phone XOR Consultation Transfer

This script instance connects the incoming call to phone C and disconnects the call from phone A. See Figure 1-40.

Figure 1-40 Three Gateways: Cisco CME IP Phone XOR After Transfer

Call Transfer Protocol Support

The following subsection provides an overview of the call transfer protocols supported using Tcl IVR scripting on a Cisco IOS voice gateway. Refer to the appropriate section above for various scenarios that may use these protocols.

Analog Hookflash and T1 CAS Release Link Trunk (RLT) Transfers

Transferor Support

A script cannot initiate a hookflash transfer towards a T1 CAS or analog FXO endpoint. Instead, the script can place an outbound call to the transfer target and connect the transferee and transfer target call legs after the call is established.

Transferee Support

A Tcl IVR script can receive a hookflash transfer request from a T1 CAS or analog FXS endpoint connected to the gateway. The subscriber is able to initiate a blind or consultation call transfer using hookflash and DTMF digits.

When the script receives a hookflash transfer trigger, it can provide dialtone and collect the transfer target destination through DTMF.

When the script receives a transfer commit request, it can do one of the following:

- Interwork the transfer request by propagating it to the transferee call leg. This can be done if the transferee call leg supports one of the transfer protocols described in this document.
- Place an outbound call to the transfer target and connect the transferee and transfer target call legs after the call is established.

Transfer Target Support

A Tcl IVR script cannot receive a consultation request or setup indication containing a consultation ID from an analog endpoint.

ISDN Call Transfer

Transferor Support

A Tcl IVR script can send an ISDN Two B-Channel Transfer (TBCT) request to the transferee call leg when the transferee and transfer target are both part of the same TBCT group on the PBX connected to the gateway.

When the script initiates a TBCT request, the Cisco IOS software places a call to the transfer target. When the transfer target answers, the Cisco IOS software initiates the TBCT if both the transferee and transfer target are part of the same TBCT group configured on the PBX. If the transferee and transfer target are not part of the same TBCT group, the transferee and transfer target call legs are bridged by the script. If the call is successfully transferred to the PBX, the transferee and transfer target call legs are released and the script can close the call. In some cases, the script can re-connect the transferor and transferee call legs if the transfer attempt is unsuccessful.

The script can do the following to initiate a consultation transfer:

- Place a consultation call to the transfer target device and connect the transferor and transfer target call leg when the call is established.

- If the transferee and transfer target are part of the same TBCT group, the script can do the following when the transfer is committed:
  - Request a local TBCT consultation ID.
  - Send a TBCT request to the transferee call leg. The transfer request includes the consultation ID.
  - If the call is successfully transferred to the PBX, the transferee and transfer target call legs are released, and the script can close the call.
  - In some cases, the script may re-connect the transferor and transferee call legs if the transfer attempt is unsuccessful.
- If the transferee and transfer target are not part of the same TBCT group, the transferee and transfer target call legs can be bridged by the script when the transfer is committed.

Transferee Support

A Tcl IVR script does not support any network-side ISDN call transfer protocols and is not able to receive a call-transfer request from an ISDN device.

Note It is possible to allow an ISDN subscriber to initiate a blind transfer using DTMF input to trigger the transfer. This mechanism is similar to the analog FXS and T1 CAS transfer mechanisms described above and is not discussed further in this document.

Transfer Target Support

A Tcl IVR script cannot receive a consultation request or setup indication with a consultation ID from an ISDN endpoint.

SIP Call Transfer

Transferor Support

A Tcl IVR script can send a REFER transfer request to a remote transferee call leg. The script can also initiate a consultation call when performing a consultation transfer.

The script can initiate a blind transfer by sending a REFER message to the remote transferee. If the transfer is successful, the transferee places a call the transfer target. The call is established without involvement of this script and the script can close the call. In some cases, the script can re-connect the transferor and transferee call legs if the transfer attempt is unsuccessful.

The script can do the following to initiate a consultation transfer:

- Place a consultation call to the transfer target device, and connect the transferor and transfer target call leg when the call is established.
- When the transfer is committed, request a consultation ID.

Note Unlike H.450 transfers, the script handling the consultation call between the transferor and transfer target does not receive a consultation request from the transferor. Instead, the consultation ID is generated locally by the script handling the original call between the transferor and transferee.

- Send a REFER to the transferee call leg. This includes the consultation ID. The transferee device includes the consultation ID in the INVITE message it sends to the transfer target.

- If the transfer is successful, the transferee calls the transfer target. The call is established without involvement of this script and the script can close the call.
- In some cases, the script may re-connect the transferor and transferee call legs if the transfer attempt is unsuccessful.

Transferee Support

A Tcl IVR script can receive a SIP REFER or BYE/ALSO transfer request from a remote SIP transferor. When the script receives a transfer request, the script can do one of the following:

- Interwork the transfer request by propagating it to the transferee call leg. This can be done if the transferee call leg supports one of the transfer protocols described in this document.

Note It is not currently possible to interwork SIP and H.450 transfer requests.

- Place an outbound call to the transfer target and connect the transferee and transfer target call legs after the call is established.

Transfer Target Support

When the gateway receives an INVITE request from the remote transferee that includes a consultation ID, it is delivered to the script instance handling the consultation call to the transfer target. The script can then connect the transferee and transfer target call legs and disconnect the transferor call leg.

Note Unlike H.450 transfers, the script handling the consultation call between the transferor and transfer target does not receive a consultation request from the transferor. Instead, the consultation ID is generated locally by the script that is handling the original call between the transferor and transferee.

H.450 Call Transfer

Transferor Support

A Tcl IVR script can send a H450.2 transfer request to a transferee call leg. The script can also initiate a consultation call when performing a consultation transfer.

The script can initiate a blind transfer by sending an H450.2 transfer request to the remote transferee. If the transfer is successful, the transferee calls the transfer target. The call is established without involvement of this script and the script can close the call. In some cases, the script can re-connect the transferor and transferee call legs if the transfer attempt is unsuccessful.

The script can do the following to initiate a consultation transfer:

- Place a consultation call to the transfer target device, and connect the transferor and transfer target call leg when the call is established.
- When the transfer is committed, request a consultation ID from the transfer target.
- Send an H450.2 transfer request to the transferee call leg. This includes the consultation ID received in the consultation response from the transfer target device. The transferee includes the consultation ID in the SETUP request it sends to the transfer target.
- If the transfer is successful, the transferee calls the transfer target and the call is established without involvement of this script. The script can then close the call.
- In some cases, the script can re-connect the transferor and transferee call legs if the transfer attempt is unsuccessful.

Transferee Support

A Tcl IVR script can receive an H450.2 transfer request from a remote H.323 transferor. When the script receives a transfer request, it can do one of the following:

- Interwork the transfer request by propagating it to the transferee call leg. This can be done if the transferee call leg supports one of the transfer protocols described in this document.

Note It is not possible to interwork SIP and H.450 transfer requests.

- Place an outbound call to the transfer target and connect the transferee and transfer target call legs after the call is established.

Transfer Target Support

A Tcl IVR script can receive a consultation request from a remote H450 transferor and send a consultation response that includes the consultation ID and transfer destination. This transfer destination is the number the transferee should use when placing a call to the transfer target.

When the gateway receives a SETUP request from the remote transferee that includes an H450.2 consultation ID, it is delivered to the script instance handling the consultation call to the transfer target. The script can then connect the transferee and transfer target call legs and disconnect the transferor call leg.

Cisco CallManager Express Call Transfer

Transferor Support

A Tcl IVR script cannot send a call transfer request to a local IP phone registered with the Cisco IOS gateway operating in Cisco CallManager Express (CME) mode. Instead, the script can place an outbound call to the transfer target and connect the transferee and transfer target call legs after the call is established.

Transferee Support

A Tcl IVR script can receive a call transfer request from a local IP phone registered with the Cisco IOS gateway operating in Cisco CME mode. When the script receives a transfer request, it can do one of the following:

- Interwork the transfer request by propagating it to the transferee call leg. This can be done if the transferee call leg supports one of the transfer protocols described in this document.
- Place an outbound call to the transfer target and connect the transferee and transfer target call legs after the call is established.

Transfer Target Support

A Tcl IVR script can receive a consultation request from a local Cisco CME IP phone and do one of the following:

- Interwork the consultation request by relaying it to the other call leg. This can be done if the transferee call leg supports one of the transfer protocols described in this document.
- Send a local consultation response to the IP phone that includes a locally generated consultation ID and transfer destination. This transfer destination is the number the transferee should use when placing a call to the transfer target.

When the gateway receives a SETUP request from the remote transferee that includes a consultation ID, it is delivered to the script handling the consultation call to the transfer target. The script can then connect the transferee and transfer target call legs and disconnect the transferor call leg.

SIP Subscribe and Notify

Tcl IVR 2.0 scripts provide the ability to subscribe to a SIP subscribe server and receive notify events. Applications can be invoked when notification is received, which is useful when the subscribed event will probably take a long time to complete, say several minutes or hours. In this case, the application can choose to free its instance and cause the system to create another instance to handle the notification when it is received.

The application that handles the notification need not be the same one that made the subscription. This provides the flexibility to make separate applications for handling subscriptions and notifications.

The application that made the subscription can perform any of the following tasks:

- Keep alive and handle notifications from the server.
- Free its instance and cause another instance of the same application to generate on notification.
- Free its instance and cause a different application to generate on notification.
- Make another module to handle notification.

SIP Headers

Tcl IVR 2.0 scripts can specify headers to be sent in SIP invite or H.323 setup messages. The script writer can piggy-back the header-value pairs in the destination URI after the '?". For example:

```
set destination "sip:joe@big.com?Subject=Hotel Reservation&Priority=urgent&
X-ReferenceNumber=1234567890"
leg setup destination callInfo leg_incoming
```

In cases where the destination string is an E.164 number instead of a URI, where headers cannot be appended to the destination URI, the **set** command can be used to set the headers. For example:

```
set setupSignal(Subject) "Hotel Reservation"
set setupSignal(Priority) "urgent"
set setupSignal(X-ReferenceNumber) "1234567890"
set callInfo(protoHeaders) setupSignal
set destination "4085550100"
leg setup destination callInfo leg_incoming
```

A data-passing mechanism is provided to pass application-specified headers to the SPI for outbound calls. Tcl scripts can retrieve headers using the evt_proto_headers or leg_proto_headers information tags. As of Cisco IOS release 12.3(4)T, access to headers is limited to SIP invite, subscribe, and notify messages, and to H.323 setup messages. The following list of headers, however, cannot be overwritten:

- call-ID
- Supported
- Require
- Min-SE
- Session-Expires
- Max-Forwards
- CSeq

Note Each call leg is limited to a maximum of 20K memory allocation for header passing. Each header avpair is limited to 256 characters. The application throws an error if the Tcl script tries to pass a header avpair greater than 256 characters or if the 20K memory has been used up.

If a call is handed off to an outbound application, the outbound application can retrieve all headers handed off to it from the previous application, plus headers from the incoming call leg set by the SPI, through the evt_handoff proto_headers information tag.

Application Instances

A Tcl IVR 2.0 application that is configured on the 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, the application starts an instance, or session, of that application. The application instance executes the application script, and can place or transfer a call to a other applications. A call can initiate a single application instance or multiple application instances, depending on how the system is configured to handle the call. A single application session can manage multiple voice calls.

In Cisco IOS Release 12.3(x), 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 could pass status information to other applications that are handling incoming calls. This type of service application can run on the gateway without being triggered by a call.

An instance of a Tcl IVR 2.0 application can be started on the gateway by using the **call application session start** CLI command. An application instance can communicate with other sessions on the same gateway and calls can be bridged between different sessions.

The mod_all_handles information tag can be used to retrieve a list of all the instances currently running on the gateway.

Note Tcl IVR 2.0 limits the number of subscriptions per handler to 18. Because each script instance is a handler, an application instance can only handle a maximum of 18 subscriptions simultaneously.

Session Interaction

A session is an instance of a Tcl application and is independent from other sessions in that they do not share data directly. For example, a global Tcl variable in one session is not available to another session. However, application sessions can communicate with other sessions on the gateway for the purpose of sending and receiving messages, or to hand off calls between sessions.

A Tcl session can initiate multiple outgoing call legs, and can have incoming and outgoing call legs handed off to it. A Tcl session can be running with no call legs if it was started in the CLI from a sendmsg or a notify, or if it disconnects the legs it is handling.

Session Start and Stop

The most common way for a session to start is when a Tcl application handles an incoming voice call, however, a session can also run with no call legs if it is started in the CLI, from a sendmsg, from a notification, or if it disconnects the call legs it was handling.

When a Tcl instance starts up, it receives one of the following events, depending on how it got started:

- ev_msg_indication
- ev_notify
- ev_session_indication
- ev_call_indication

If a user stops a Tcl session using the **call app session stop** CLI command, the Tcl script receives an *ev_session_terminate* event. The Tcl script is expected to close. If the Tcl script does not close after 10 seconds, the session is shut down anyway and all call legs are disconnected. This gives the Tcl script time to clean up gracefully.

If the session returns after handling an event and there are no active timers, legs, registered services, or subscriptions, the session is closed.

Sending Messages

Messages are sent to other application instances using the **sendmsg** command. The sendmsg command is an asynchronous command that does not have to wait for the destination to act on the event. If an application name is provided, a new instance of that application is generated.

Receiving Messages

Applications are notified of incoming messages from other applications through the *ev_msg_indication* event. Any parameters passed with the message are then available to the application through the evt_msg information tag. The handle of the sending application is available through the evt_msg_source information tag.

Call Handoff

In addition to passing the name of an application, the **handoff** command allows the passing of a handle. For example, assume a Tcl script gathers the caller's account number, then receives a notification that the call is being handled by another instance. The script can hand the incoming call leg to the other application instance using the handoff command, providing information in the argument string. When the other application instance returns the call leg, this application receives an *ev_returned* event.

Handoff Return

Handoff returns of a set of separate call legs received from different sessions should be done with a separate **handoff return** commands for each leg. The command "handoff return leg_all" is undefined in this case. The entire set of legs should return to the return location for the first user-defined leg.

Handoff return of a set of conferenced legs returns both legs to the same session. For example, if a session has been handed leg1 from session1 and leg2 from session2, and it conferenced the two legs together. Then the command **handoff return** *$leg2* returns both legs, conferenced together, to session2.

Service Registry

The services registry is a database that keeps track of every Tcl IVR 2.0 application instance that registers as a service. Other Tcl applications can then find and communicate with any registered application.

A Tcl session is not registered as a service through Cisco IOS software. A running instance of a Tcl IVR 2.0 application registers itself as a service by using the Tcl **service** command. The handle of any registered service can be retrieved using the mod_handle_service information tag.