CHAPTER 3

Cisco VoiceXML Applications

Revised: June 20, 2007, OL-11175-01
For the latest version of the scripts described in this chapter, go to the developer support website at http://www.cisco.com/cgi-bin/dev_support/access_level/products.cgi?product=VOICE_XML_GATEWAY.

This chapter describes applications using Cisco VoiceXML features, and consists of the following sections:

Hybrid Application

The TCL and VoiceXML hybrid script in this example is a call screening application. This script is invoked, when the user calls into the gateway. The user is prompted to record their name which is recorded in a VoiceXML dialog and embedded in the TCL script. After the name is recorded, the destination number is collected and the script places a call. When the called party answers the call, the recorded name is played. The called party is prompted to enter the digit one to accept the call or two to reject it. If the called party chooses to accept the call, the incoming and outgoing legs are bridged. If the called party rejects the call, an audio prompt is played to the calling party indicating that the call has been rejected and the call is then disconnected.

Procedure init

#The init procedure defines the initial parameters of the digit collection. The first set of parameters, param1 allows the users to enter digits before a prompt playout is complete. It also defines the asterisk key to terminate the digit collection process and the pound key to indicate the end of digit collection. The collected digit is matched with a dialplan configured on a dial-peer. The second set of parameters, param2 has the same definition except that the collected digits are not matched with a dialplan.

```
#

proc init { } {
    global param1
    global param2

    set param1(interruptPrompt) true
```

```
    set param1(abortKey) *
    set param1(terminationKey) #
    set param1(dialPlan) true

    set param2(interruptPrompt) true
    set param2(abortKey) *
    set param2(terminationKey) #

}
```

Procedure init_perCallVars

In this procedure, the global variables are initialized.

```
proc init_perCallVars { } {
    global ani
    global dnis
    global dest

    set dest ""

    set ani [infotag get leg_ani]
    set dnis [infotag get leg_dnis]

}
```

Procedure act_RecordGreeting

This procedure is executed when the application receives an ev_setup_indication event. A setup acknowledgement and a signal level connect message is sent to the incoming call leg. The VoiceXML dialog is invoked and the VoiceXML dialog prompts the user to record the name. The recording codec is g729r8 and the recording terminates after 6 seconds or if the user enters a Pound key. The recorded audio is stored in the record field, myrecord.

```
#

proc act_RecordGreeting { } {
    global param1
    set baseURI http://WEB-SERVER/
    set vxmlScript {
                    <vxml version="2.0">
                      <form>
                        <catch event="error">
                          <audio src="technicalProblem.au"/>
                        </catch>

                        <var name="mydur"/>
                        <var name="mysiz"/>

                        <block>
                          <prompt><audio src="record.au"/></prompt>
                        </block>
                        <record name="myrecord" maxtime="6s" finalsilence="3s" beep="true"
type="audio/basic;codec=g729r8" dtmfterm="true">
                          <filled>
                            <assign name="mydur" expr="myrecord$.duration"/>
                            <assign name="mysiz" expr="myrecord$.size"/>

                            <log>DTMF Termintation key is <value
expr="myrecord$.termchar"/> </log>
```

```
                            <log>Duration of recording is <value
expr="myrecord$.duration"/> </log>
                            <log>Size of recorded audio file is <value
expr="myrecord$.size"/> </log>
                            <prompt><value expr="myrecord"/></prompt>

                            <exit namelist="myrecord"/>
                          </filled>
                        </record>
                      </form>
                    </vxml> }


    init_perCallVars

    leg setupack leg_incoming
    leg proceeding leg_incoming
    leg connect leg_incoming

    leg vxmldialog leg_incoming -u $baseURI -v $vxmlScript
}
```

Procedure act_GetDest

This procedure is called when the application receives the vxml_dialog_done event. It checks for the recorded audio file and plays it to the incoming leg. It then prompts and collects the destination number.

```
#

proc act_GetDest { } {
    global param1
    global param
    global recordedName

     # check the sub-event name
     set exp_ev vxml.session.complete
     set ev [infotag get evt_vxmlevent]
     if {$ev != $exp_ev} {
         puts "DEBUGGING: expected event $exp_ev, got $ev"
         puts "VXML Dialog not complete"
         call close
         return
     }

     # check the dialog status
     set status [infotag get evt_status]
     if {$status != "vd_000"} {
         puts "DEBUGGING: VXML Dialog status, expected vd_000, got $status"
         puts "VXML Dialog not successful"
         call close
         return
     }

     # get the audio variable
     infotag get evt_vxmlevent_params param
     if !{[info exists param(myrecord)]} {
         puts "DEBUGGING: myrecord does not exist"
         call close
         return
     }

     set recordedName $param(myrecord)
```

```
     media play leg_incoming $param(myrecord)

    puts "DEBUGGING: leaving act_GetDest"
    media play leg_incoming tftp://TFTP-SERVER/_enter_dest.au
    leg collectdigits leg_incoming param1

}
```

Procedure act_GetDestDone

This procedure is excecuted when the application receives a ev_collectdigits_done event. It then collects a password from the user.

```
#

proc act_GetDestDone { } {
    global dest
    global param1
    global param2
    global passwd

    set status [infotag get evt_status]
    puts "DEBUGGING : act_GetDestDOne status is $status"

    if {$status == "cd_004"} {
        set dest [infotag get evt_dcdigits]
        set pattern(passwd) .+

    leg collectdigits leg_incoming param2 pattern
    media play leg_incoming tftp://TFTP-SERVER_enter_pin.au
    } else {
        media play leg_incoming tftp://TFTP-SERVER/_invalid_dest.au
        leg collectdigits leg_incoming param1
        fsm setstate same_state
    }

}
```

Procedure act_GetPasswdDone

When the password is collected, the application places an outbound call with the destination number it has collected. The password information is set in the callInfo field.

```
#

proc act_GetPasswdDone { } {
    global passwd
    global dest

    set status [infotag get evt_status]
    puts "DEBUGGING : act_PasswdDone status is $status"

    if {$status == "cd_005"} {
        set passwd [infotag get evt_dcdigits]
        set callInfo(pinNum) $passwd

        leg setup $dest callInfo leg_incoming
    } else {
        media play leg_incoming tftp://TFTP-SERVER/_invalid_pin.au
        set pattern(passwd) .+
```

```
        leg collectdigits leg_incoming param2 pattern
        fsm setstate same_state
    }
}

proc act_CallSetupDone { } {
    global dest

    set status [infotag get evt_status]
    puts "DEBUGGING : act_CallSetupDone status is $status"
    if { [infotag get evt_status] == "ls_000" } {
        connection destroy con_all
    } else {
        # call setup fail
        puts "DEBUGGING : Call Setup Failed, incoming disconnected "
        call close
    }
}
```

Procedure act_Playout

When the connection between the incoming and outgoing call legs is destroyed. The recording of the calling party's name is played to the called party.

```
#

proc act_Playout {} {
    global param

    puts "DEBUGGING: In proc goto_Playout"
    media play leg_outgoing $param(myrecord)
    return
}
```

Procedure act_AnswerCall

The user is prompted to enter 1 to accept the call or 2 to reject the call.

```
#

proc act_AnswerCall { } {
    global param4

    set param4(maxDigits) 1
    leg collectdigits leg_outgoing param4
    media play leg_outgoing tftp://TFTP-SERVER/answercall.au
    set event [infotag get evt_event]
    puts "\n EVENT in act_AnswerCall is  $event"
    fsm setstate ACCEPTCALL

}
```

Procedure act_ApproveCall

This procedure handles the called party's decision to accept or reject the call. If the called party selects 1 (accepts the call), the state is set to CALLAPPROVED. If the called party selects 2 (reject the call), the script will play a prompt to the calling party.

```
#

proc act_ApproveCall { } {
```

```
     set event [infotag get evt_event]
     puts "\n EVENT in act_ApproveCall is  $event"
     set status [infotag get evt_status]

    #Collect Success ...matched pattern
    if {$status == "cd_005"} {
        set answer_call [infotag get evt_dcdigits]

        if {$answer_call == 1} {
            puts "USER ACCEPTS THE CALL"
            fsm setstate CALLAPPROVED
        } elseif {$answer_call == 2} {
            media play leg_incoming tftp://TFTP-SERVER/reject_call.au
            fsm setstate REJECTCALL
        } else {
            media play leg_outgoing tftp://TFTP-SERVER/wrong_selection.au
            fsm setstate WRONGSELECTION
        }
    }
}
```

Procedure act_Bridge

This procedure connects (bridges) the incoming and outgoing call legs if the called party accepts the call.

```
#
proc act_Bridge { } {

    puts "DEBUGGING: in act_Bridge"
    set ev [infotag get evt_event]
    puts "DEBUGGING: in act_Bridge event is $ev"

    set status [infotag get evt_status]
    puts "DEBUGGING: in act_Bridge status is $status"

    connection create leg_incoming leg_outgoing
    set connectionID [infotag get con_ofleg leg_outgoing]
    puts "\n  connectionID in act_Bridge:$connectionID"
    fsm setstate CONF_BRIDGING

}

proc act_Nothing { } {

}

proc act_Bridged { } {

}
```

Procedure act_UnBridge

This procedure destroys the connection between the incoming and outgoing call legs if the called party rejects the call.

```
proc act_UnBridge { } {
    connection destroy con_all
}

proc act_Ignore { } {
# Dummy
    puts "Event Capture"
```

```
}
```

proc act_Cleanup { }{

```
puts "DEBUGGING: in act_Cleanup"
call close
}

requiredversion 2.0
init


#---------------------------------
#   State Machine
#---------------------------------
  set fsm(any_state,ev_disconnected)           "act_Cleanup            same_state"
  set fsm(CALL_INIT,ev_setup_indication)       "act_RecordGreeting     RECORD"
  set fsm(RECORD,ev_vxmldialog_done)           "act_GetDest            GETDEST"
  set fsm(GETDEST,ev_collectdigits_done)       "act_GetDestDone       GETPASSWD"
  set fsm(GETDEST,ev_media_done)               "act_Nothing            same_state"
  set fsm(GETPASSWD,ev_media_done)             "act_Nothing            same_state"
  set fsm(GETPASSWD,ev_collectdigits_done)     "act_GetPasswdDone     PLACECALL"
  set fsm(PLACECALL,ev_media_done)             "act_Nothing            same_state"
  set fsm(PLACECALL,ev_setup_done)             "act_CallSetupDone      CONF_BREAKING"
  set fsm(CONF_BREAKING,ev_destroy_done)       "act_Playout            PLAYOUT"
  set fsm(PLAYOUT,ev_media_done)               "act_AnswerCall         COLLECTANSWER"
  set fsm(REJECTCALL,ev_media_done)            "act_Cleanup            same_state"
  set fsm(WRONGSELECTION,ev_media_done)        "act_AnswerCall         COLLECTANSWER"
  set fsm(ACCEPTCALL,ev_collectdigits_done)    "act_ApproveCall        CALLAPPROVED"
  set fsm(CALLAPPROVED,ev_media_done)          "act_Bridge             CONF_BRIDGING"
  set fsm(CONF_BRIDGING,ev_create_done)       "act_Bridged            CALLACTIVE"
  set fsm(CALLACTIVE,ev_disconnected)          "act_Cleanup            same_state"
  set fsm(CALLDISCONNECT,ev_media_done)        "act_Cleanup            same_state"
  set fsm(CALLDISCONNECT,ev_disconnect_done)   "act_Cleanup            same_state"
  set fsm(CALLDISCONNECT,ev_leg_timer)         "act_Cleanup            same_state"

  fsm define fsm CALL_INIT
```

Speech Enabled Banking Application

The speech enabled banking application demonstrates the ability to check balances, deposit, withdraw, and transfer funds using speech input. User authenticate, when you first launch the application you will hear a prompt that says, " please enter your account number." After it collects your spoken input, it asks for your pin number. After collecting your pin and account number, the application requests an authorization from the RADIUS server.

User Authentication

The userauthenticate.vxml script collects your account and pin numbers, and authorizes you to perform transactions.

```
<?xml version="1.0"?>
<vxml version="2.0" application="root.vxml">

<catch event="error.com.cisco.aaa.authorize.failure">
#Denotes an authorization failure.
<prompt> Radius Server error. </prompt>
```

```
</catch>

    <form id="userAuthentication" scope="document">
        <property name="timeout" value="10s"/>

    <field name="accountNumber" type="digits?length=6">
        <prompt> Please enter or say your account number </prompt>
        <filled>
              <log> Account is <value expr="accountNumber"/></log>
              <log>Confidence: <value expr="accountNumber$.confidence"/> </log>
              <log>Utterance: <value expr="accountNumber$.utterance"/> </log>
              <log>Inputmode: <value expr="accountNumber$.inputmode"/> </log>
            </filled>
    </field>

    <field name="pinNumber" type="digits?length=4">
        <prompt> Please enter or say your pin number </prompt>
          <filled>
                <log> Pin is <value expr="pinNumber"/></log>
          </filled>
    </field>

    <!--filled mode="all" namelist="accountNumber pinNumber ">
                <prompt> it works </prompt>
    </filled-->
<object
        name="authorize"
        classid="builtin://com.cisco.aaa.authorize">
#The aaa authorize command implemented through <object>, sends a RADIUS authentication or
authorization request, and allows the script to retrieve information that the RADIUS
server includes in its response.
        <param name="account" expr="'accountNumber'"/>
        <param name="password" expr="'pinNumber'"/>
</object>

      <filled>
        <log>DEBUG: got authorize result as <value expr="authorize.result"/>  </log>
        <if cond="authorize.result=='fail'">
            <log>DEBUG: Authentication failed due to <value
expr="authorize.attributes.h323_return_code"/></log>
            <prompt> Your account and pin did not match. </prompt>
            <clear namelist="accountNumber pinNumber"/>
            <goto nextitem="userAuthentication"/>
        <else/>
            <goto next="menu.vxml"/>
        </if>
      </filled>

    </form>
</vxml>
```

Menu

The menu.vxml script provides a menu of transaction options.

```
<?xml version="1.0"?>
<vxml version="2.0" application="root.vxml">

<catch count="1" event="nomatch">
    <prompt>Please speak clearly. Select Deposit, Withdrawal, Check Balance, Transfer
Funds or Repeat Menu </prompt>
</catch>
```

```
<catch count="1" event="noinput">
    <prompt>You must make a selection </prompt>
</catch>

<help> If you would like to deposit, say deposit. For withdrawal, say withdrawal. To
transfer funds, say transfer funds. To check balance, say check balance. </help>

<catch count="3" event="nomatch noinput">
<prompt> Sorry, please try again later </prompt>
<exit/>
</catch>

<var name="transaction"/>
    <menu id="transactionMenu" dtmf="true">
        <prompt bargein="true" timeout="10ms">
         Welcome to the AJAX Banking Application. Please choose from the following
choices. <enumerate/></prompt>

            <choice caching="safe"  next="deposit.vxml"> Deposit </choice>
            <choice caching="safe"  next="withdrawal.vxml"> Withdrawal </choice>
            <choice caching="safe"  next="checkBalance.vxml"> Check Balance </choice>
            <choice caching="safe"  next="transferFunds.vxml"> Transfer Funds </choice>
            <choice caching="safe"  next="menu.vxml"> Repeat Menu </choice>

    </menu>
</vxml>
```

Withdrawal

The withdrawal.vxml script selects the account from which you want to withdraw, and allows you to enter the withdrawal amount.

```
<?xml version="1.0"?>
<vxml version="2.0" application="root.vxml">

<form id="withdrawal_info" scope="document">
    <field name="acctType">
    <prompt>Please select the account you want to withdra from. Select one of the
following. <enumerate/> </prompt>
        <option value="SAVINGS">Savings</option>
        <option value="CHECKING">Checking</option>
    <filled>
        <log> ACCOUNT TO WITHDRAW FROM IS : <value expr="acctType"/></log>
            <goto nextitem="withdrawalAmt"/>
    </filled>
    </field>

    <field name="withdrawalAmt" type="currency" expr="0">
    <prompt> Please say or enter the amount you want to withdraw </prompt>
    <filled>
        <if cond="withdrawalAmt == "0">
                    <prompt>Withdrawal amount cannot be zero. Please enter the amount you
want to withdrawal</prompt>
                    <clear namelist="withdrawalAmt"/>
        <else/>
                <log> DEPOSIT AMOUNTT IS <value expr="withdrawalAmt"/></log>
  <prompt> withdrawal amount is <value expr="withdrawalAmt"/> </prompt>
                    <!--submit expr="webServer+'withdrawal'" method="get"
namelist="acctType withdrawalAmt"/-->
        </if>
    </filled>
```

```
    </field>
</form>
</vxml>
```

Deposit

The deposit.vxml script selects the account to which you want to deposit, and allows you to enter the deposit amount.

```
<?xml version="1.0"?>
<vxml version="2.0"  application="root.vxml">

<form id="deposit_info" scope="document">
    <field name="acctType">
        <prompt>Please select Savings or checking account <enumerate/></prompt>
    </prompt>
        <option value="SAVINGS">Savings</option>
        <option value="CHECKING">Checking</option>
    <filled>
            <log> ACCOUNT TO DEPOSIT TO IS <value expr="acctType"/></log>
       <goto nextitem="depositAmt"/>
    </filled>
    </field>

    <field name="depositAmt" type="number" expr="0">
    <prompt>Please enter the amount you want to deposit </prompt>
    <filled>
        <if cond="depositAmt == '0'">
                <prompt> Depositamount cannot be zero. Please enter the amount you want to
withdrawal</prompt>
                <clear namelist="depositAmt"/> <else/>
                <log> DEPOSIT AMOUNT is <value expr="depositAmt"/></log>
           <else/>
         <prompt> Deposit amount is <value expr="depositAmt"/> </prompt>
                <!--submit expr="webServer+'deposit'" method="get" namelist="acctType
depositAmt"/-->
       </if>
    </filled>
    </field>
</form>
</vxml>
```

Transfer of Funds

The transfer.vxml script allows you to transfer funds between accounts.

```
<?xml version="1.0"?>
<vxml version="2.0" application="root.vxml">

<form id="transfer_info" scope="document">
    <field name="transType">
        <prompt> Please say one of the following options  <enumerate/><prompt>
        <option value="SAVINGS_TO_CHECKING">Savings To Checking</option>
        <option value="CHECKING_TO_SAVINGS">Checking To Savings</option>
    <filled>
                <log> TRANSFER FUNDS FROM : <value expr="transType"/></log>
            <goto nextitem="transferAmt"/>
    </filled>
    </field>

    <field name="transferAmt" type="number" expr="0">
```

```
        <prompt> Please say or enter the amount you want to deposit</ prompt>
    <filled>
          <if cond="transferAmt == '0'">
                   <prommpt>Transfer amount cannot be zero. Please enter the amount you
want to transfer</prompt>
                        <clear namelist="transferAmt"/>
                  <else/>
                        <log> TRANSFER AMOUNTT IS <value expr="transferAmt"/></log>
                        <!--submit expr="webServer+'transfer'" method="get"
namelist="transType transferAmt"/-->
                        <prompt> Transfer amount is <value expr="transferAmt"/> </prompt>
          </if>
    </filled>
    </field>
</form>
</vxml>
```

Checking Balances

The checkbalance.vxml script allows you to check your account balances.

```
<?xml version="1.0"?>
<vxml version="2.0" application="root.vxml">

<form id="check_balance_info" scope="document">
    <field name="acctType">
    <prompt> Please select Savings account or Checking account </prompt>
        <option value="SAVINGS">Savings</option>
        <option value="CHECKING">Checking</option>
    <filled>
        <log> ACCOUNT TYPE IS <value expr="acctType"/></log>
                    <submit expr="webServer+'balance'" method="get" namelist="acctType"/>
    </filled>
    </field>
</form>
</vxml>
```

Root Document

The root.vxml script defines default application properties, error handlers, and link elements.

```
<?xml version="1.0"?>
<vxml version="2.0">
<meta http-equiv="Expires" content="Monday ,Dec 31  2002"/>

    <property name="caching" value="safe"/>
    <property name="fetchtimeout" value="5s"/>
    <property name="timeout" value="3s"/>
    <property name="interdigittimeout" value="2s"/>
    <property name="termchar" value="#"/>
    <property name="bargein" value="true"/>
    <!--property name="inputmodes" value="dtmf"/-->

    <var name="webServer" expr="'http://171.71.16.68:8081/vxml/servlet/'"/>

    <catch event="error">
        <prompt>We are having technical difficulties. </prompt> <reprompt/>
    </catch>

    <catch event="nomatch ">
        <prompt>No Match  </prompt><reprompt/>
```

```
    </catch>

    <catch event ="noinput">
        <prompt>No Input  </prompt><reprompt/>
    </catch>

   <catch event ="noinput nomatch error" count="3">
    <goto next="exit.vxml#finalExit"/>
   </catch>

    <link next="menu.vxml" dtmf="9"></link>

</vxml>
```

ASR and TTS Application

The script below is a sample application that illustrates automatic speech recognition (ASR) , text-to-speech (TTS) synthesis, and an outbound call using the the <transfer> element. It also verifies configurations for the ASR and TTS servers and, inbound and outbound dial-peers.

When you dial into a Cisco AS5300 access server, a menu prompt plays, asking you to select 1 for the phone demonstration and 2 for the music demonstration. You can make your selection by entering a DTMF or voice input.

The phone demonstration prompts you to speak a person's name. After matching your input to the grammar specified in the form, the application dials a phone number associated with the matched name.

The music demonstration prompts you to speak an artist's name. After matching your input, the application plays an audio file.

To run this application, follow these steps:

Note For information on Cisco IOS configuration required for this application, refer to the *Cisco IOS Tcl IVR and VoiceXML Application Guide*.

1. Load the VoiceXML script, sample.vxml into flash on the Cisco AS5300 access server.
2. Configure the Cisco AS5300.
3. Configure the ASR and TTS servers.
4. Configure the application.
5. Configure the inbound and outbound dial-peers for the call to be placed.

The application is triggered when you dial the access number to the Cisco AS5300.

```
<?xml version="1.0"?>
<vxml version="2.0">

    <catch event="error">
      <prompt> We are having technical difficulties, please try again later </prompt>
    </catch>

    <catch event="nomatch ">
```

```
      <prompt> That is an invalid selection. Please try again </prompt>
    </catch>

    <catch event ="noinput">
      <prompt> We did not get your input. Please try again.</prompt>
    </catch>

   <catch event ="noinput nomatch error" count="3">
      <prompt> Sorry, please try again later</prompt>
        <exit/>
   </catch>

  <menu id="demo" dtmf="true">
       <prompt> Please select 1 for Phone demo, select 2 for Music demo. </prompt>
            <choice caching="safe" next="#phone"> one </choice>
            <choice caching="safe" next="#music"> two </choice>
  </menu>


   <form id="phone">
     <var name="phoneNum"/>
     <var name="mydur"/>

    <catch event="nomatch ">
      <prompt> Please say another name</prompt>
    </catch>

    <catch event ="noinput">
      <prompt> We did not get your input. Please say the name of the person you wish to
call</prompt>
    </catch>

     <field name="callee">
       <prompt bargein="true"> Please say the name of the person you wish to call, Peter,
Jane or Jacob </prompt>

       <grammar version="1.0" xml:lang="en-us" root="place">
         <rule id="place" scope="public">
          <one-of>
            <item> peter </item>
            <item> jane </item>
            <item> jacob  </item>
          </one-of>
        </rule>
      </grammar>

      <filled>
             <if cond="callee=='peter'">
                   <assign name="phoneNum" expr="'5241111'"/>
             <elseif cond="callee=='jane'"/>
                   <assign name="phoneNum" expr="'5243333'"/>
             <elseif cond="callee=='jacob'"/>
                   <assign name="phoneNum" expr="'5245555'"/>
             </if>
             <prompt> Calling <value expr="callee"/>now</prompt>
      </filled>
    </field>


    <transfer name="mycall"  destexpr="'phone://'+ phoneNum" bridge="true"
connecttimeout="15s" maxtime="180s" cisco-longpound ="true">
      <filled>
        <assign name="mydur" expr="mycall$.duration"/>
           <if cond = "mycall == 'busy'">
```

```
              <prompt> Called party is busy </prompt>
              <log>Status of transfer is BUSY</log>
            <elseif cond = "mycall == 'noanswer'"/>
              <prompt> Called party is is not answering </prompt>
              <log>Status of transfer is NO_ANSWER</log>
            <elseif cond = "mycall == 'near_end_disconnect'"/>
              <log>Status of transfer is NEAR_END_DISCONNECT</log>
            <elseif cond = "mycall == 'far_end_disconnect'"/>
              <prompt> Called party disconnected </prompt>
              <log>Status of transfer is FAR_END_DISCONNECT</log>
            <elseif cond = "mycall == 'unknown'"/>
              <prompt> Called transfer failed </prompt>
            </if>

            <log>The value in mycall is <value expr="mycall"/></log>
            <log>Duration of call is <value expr="mydur"/></log>
      </filled>
    </transfer>
  </form>

    <form id="music">
    <catch event="nomatch ">
      <prompt> Please say another artist's name</prompt>
    </catch>

    <catch event ="noinput">
      <prompt> We did not get your input. Please say the name of the artist </prompt>
    </catch>
       <field name="performer">
         <prompt bargein="false">Please say the name of the artist Madonna, Bon Jovi or
Aqua </prompt>

         <grammar version="1.0" xml:lang="en-us" root="place">
           <rule id="place" scope="public">
             <one-of>
               <item> madonna </item>
               <item> aqua </item>
               <item> bon jovi</item>
             </one-of>
           </rule>
         </grammar>

         <filled>
             <if cond="performer=='madonna'">
               <prompt><audio src="audio/crazy.au"/></prompt>
             <elseif cond="performer=='aqua'"/>
               <prompt><audio src="audio/barbie.au"/></prompt>
             <elseif cond="performer=='bon jovi'"/>
               <prompt><audio src="audio/itsMyLife.au"/></prompt>
             </if>
         </filled>
      </field>
    </form>
</vxml>
```

Multi-Language Application

The following multi-language sample scripts are French and Mandarin.

French:

```
<?xml version="1.0"?>
<vxml version="2.0" xml:lang="fr" base="http://nuance-asr/recognizer">
    <var name="number"/>
    <form id="places">
    <nomatch> Did not match </nomatch>
        <property name="bargein" value="true"/>
        <property name="timeout" value="10s"/>

        <field name="name">
     <prompt xml:lang="en-US"> Good Morning, Do you speak English? </prompt>
      <prompt xml:lang="fr"> Bonjour, S'il vous plaît choisissez anglais ou
français</prompt>
      <grammar version="1.0" root="name-choice" xml:lang="fr">
                <rule id="name-choice" scope="public">
          <one-of>
               <item xml:lang="fr">bonjour</item>
               <item xml:lang="en-US">welcome</item>
          </one-of>
        </rule>

      </grammar>
            <filled>

                <prompt> You chose <value expr="name"/></prompt>
                <if cond="name=='bonjour'">
                    <prompt xml:lang="fr"> Merci</prompt>
                <elseif cond="name=='welcome'"/>
                   <prompt xml:lang="fr"> vous avez choisi l' anglais </prompt>

                </if>
            </filled>
        </field>
    </form>
</vxml>
```

Mandarin

```
<?xml version="1.0"?>
<vxml version="2.0" xml:lang="en-US" base="http://nuance-asr/recognizer">
    <var name="number"/>
    <form id="places">
        <nomatch> Did not match </nomatch>
        <property name="bargein" value="true"/>
        <property name="timeout" value="10s"/>
        <field name="name">
            <grammar version="1.0" root="name-choice" xml:lang="ch">
                 <rule id="name-choice" scope="public">
                    <one-of>
                        <item xml:lang="ch">nv3 xue2</item>
                        <item xml:lang="ch">jue2 yuan2</item>
                    </one-of>
                 </rule>
            </grammar>
             <filled>
                 <prompt> You chose <value expr="name"/>
                 </prompt>
```

```
            </filled>
        </field>
    </form>
</vxml>
```

Recording Application

The recording application demonstrates recording of audio messages to different destination servers.

Note

- When a user hangs up during recording, the recording terminates and a telephone.disconnect.hangup event is thrown. The user must catch the disconnect event to continue playing prompts and submitting recordings.
- For a RAM recording, the recording is accessible through the recording variable.
- For a streamed recording (such as HTTP, RTSP, SMTP) the recording is streamed directly to the external server until the user hangs up. The recording terminates and a telephone.disconnect.hangup event is thrown.

RAM Recording

The ram_rec.vxml prompts the user to record an audio message. The message is recorded to RAM on the Cisco AS5300 universal access server. The recorded audio is played back and the user is prompted to record another message.

```
<?xml version="1.0"?>
<vxml version="1.0">

   <form id="record_to_ram">
      <record name="myrec"
          beep="true"
          maxtime="10s"
          dtmfterm="true"
          finalsilence="10ms"
          type="audio/basic;codec=g711ulaw">
          <prompt><audio src="record.au"/></prompt>

        <filled namelist="myrec">
            <prompt><value expr="myrec"/></prompt>
            <clear namelist="myrec"/>
        </filled>
      </record>
   </form>
</vxml>
```

HTTP Recording

The http_rec.vxml script prompts the user to record an audio message to an HTTP server. To process the recorded audio, the server-side script is specified in the cisco-dest attribute.

Modify the following to point to your server-side script:

cisco-dest= “http://myServer/record.php”

To playback the recorded audio, use the <audio> tag to specify the location of the recorded audio.

```
<?xml version="1.0"?>
<vxml version="1.0">

    <form id="record_to_http">

      <record name="myrec"
          beep="true"
          maxtime="15s"
          finalsilence="10s"
          dtmfterm="true"
          type="audio/basic;codec=g711ulaw"
              cisco-dest="http://myServer/saveRecording.php">
        <prompt> <audio src="record.au"/></prompt>
      </record>

      <block>
        <log> DURATION: <value expr="myrec$.duration"/></log>
        <log> SIZE: <value expr="myrec$.size"/></log>
        <log> CHAR: <value expr="myrec$.termchar"/></log>
      </block>
   </form>
</vxml>
```

SMTP Recording

The smtp_rec.vxml script prompts the user to record to an SMTP server. The location of the SMTP server is specified in the cisco-dest attribute.

Modify the following to point to your SMTP server:

cisco-dest= “mailto:test@myserver”

The recorded audio is stored on the SMTP server.

Note Configure the mail transfer agent (MTA) prior to using this script. For information on configuring the MTA, refer to the *Cisco IOS Tcl IVR and VoiceXML Application Guide*.

```
<?xml version="1.0"?>
<vxml version="1.0">

   <form id="record_to_smtp">

      <record name="myrec"
          beep="true"
          maxtime="15s"
          dtmfterm="true"
          type="audio/basic;
          codec=g711ulaw"
          cisco-dest="mailto:test@myserver">
#cisco-dest points to a URL specifying a recording destination.
        <prompt><audio src="record.au"/></prompt>
      </record>

      <block>
    <log> DURATION: <value expr="myrec$.duration"/> </log>
    <log> SIZE: <value expr="myrec$.size"/> </log>
    <log> CHAR: <value expr="myrec$.termchar"/> </log>
     </block>
   </form>
</vxml>
```