Getting Started with CallSuite 4

This chapter introduces CallSuite and contains the following sections:

- Overview
- Tutorial 1: Generating a Starter Application in CallSuite
- Tutorial 2: Modifying Your CallSuite Starter Application

Overview

The CallSuite portion of CT ADE provides a specialized platform that lets you reduce the repetitive tasks associated with a traditional telephony application development, and enables you to stay focused on the essential and profitable aspects of developing your application.

Made up of a collection of ActiveX components that includes an application wizard as well as a wizard utility, CallSuite makes it easy for you to add CT functions to your existing systems or to create applications from scratch. The ActiveX components can be used in any Windows development environment that supports ActiveX components. The CallSuite Wizard utility provides a GUI environment where you can create new CT functions, and depending on how your application was developed, edit existing ones.

You develop applications in CallSuite one of two ways: by automatically generating an application using CallSuite Application Wizard, or by writing your application directly in the programming language of your choice and adding ActiveX components to it.

If you use the CallSuite Application Wizard to automatically generate an application for you, you can view all of the CT functions contained in your application from the CallSuite Wizard. From this main window you also can modify or remove functions as well as add new ones.

Alternatively, if you initially created your application using a language other than the Callsuite Application Wizard then you cannot use the CallSuite Wizard to add new CT functions to the project. You must add and edit functions directly in the code.

CallSuite Components

The CallSuite set of ActiveX components can integrate directly into visual programming environments such as Visual Basic, Visual C++, or Delphi, and provide CT-specific development functionality.

CallSuite components are invisible at runtime and do not have an interface, but you can easily create one using the conventional visual controls such as buttons, scroll bars, etc., so that your system administrator can more easily facilitate the application during runtime.

VoiceBocx

VoiceBocx gives you control of trunk and media hardware through Topaz Trunk Resources and Media Resources by handling call control, the playing and recording of sound files and tones, and getting DTMF digits from callers.

Trunk Resources are responsible for all call control, which includes dialing out, accepting an incoming call, and hanging up when a call is finished. Media Resources control all playing and recording of sound files and tones, as well as getting DTMF digits from callers.

FaxBocx

FaxBocx adds fax support to your applications by giving you control of fax hardware through Topaz Fax Resources. FaxBocx works closely with VoiceBocx and the other CallSuite controls. In your programs, you use VoiceBocx first to control call processing functions such as answering or placing phone calls. Next, you use FaxBocx to process the fax information. Finally, you use VoiceBocx again to finish processing the call.

ConfBocx

ConfBocx lets you manage conferencing capabilities in your applications. You also control conferencing hardware through Topaz Conference Resources. ConfBocx works closely with VoiceBocx and the other CallSuite controls. In your programs, you use VoiceBocx first to control call processing, such as answering or placing phone calls, playing menus to callers, and getting the digits they type in response. Next, you use ConfBocx to connect to a conference, and then to disconnect from the conference. Finally, you use VoiceBocx again to finish processing the call.

ChatterBocx

Chatterbox lets you control text-to-speech features in your applications through Topaz Text-To-Speech Resources. ChatterBocx works closely with VoiceBocx and the other CallSuite components. In your programs, you use VoiceBocx first to control call processing, such as answering or placing phone calls, playing menus to callers, and getting the digits they type in response. Next, you use ChatterBocx to speak text to the caller. Finally, you use VoiceBocx again to finish processing the call.

MatchBocx

MatchBocx lets you add speech recognition capabilities to your application through Topaz Voice Recognition Resources. MatchBocx works closely with the VoiceBocx control. In your programs, you use VoiceBocx first to control call processing, such as answering or placing phone calls, playing menus to callers, and getting the digits they type in response. Next, you use MatchBocx to recognize speech from the caller. Finally, you use VoiceBocx again to finish processing the call.

CallSuite Wizard

CallSuite Wizard is a visual application generator that writes, edits, and tests telephony application source code. The CallSuite Application Wizard is a series of dialogs that you use to create starter applications in one of several popular visual tools such as Visual Basic, Visual C++, Delphi, and PowerBuilder. So in order to create an application using CallSuite you use: the programming language of your choice, the CallSuite Application Wizard (this is optional), and the CallSuite Wizard. You start by opening the programming language in which you plan to create your application; you can choose from Visual Basic, Visual C++, or Delphi.

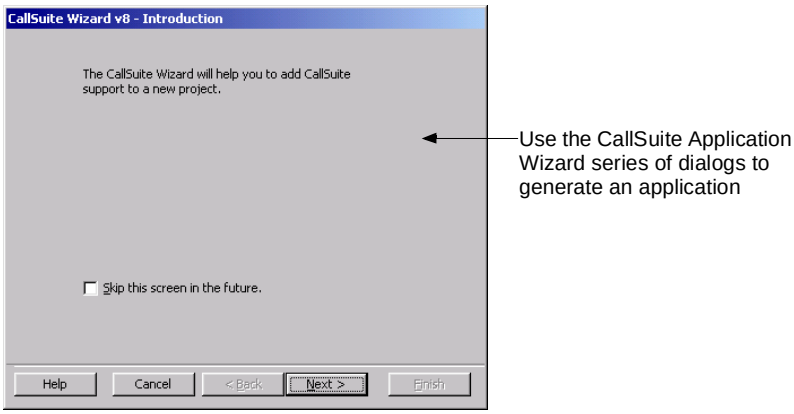

Figure 4-1 CallSuite Application Wizard—first dialog

Next, you launch the CallSuite Application Wizard from within the programming language. As you progress through the wizard dialogs, you specify how you want to set up your application and then the wizard generates a project containing your application. The telephony functions in applications are called Routines, which are series of source code statements you can invoke using a single name. Generated applications contain two sets of Routines: those that you chose in the Wizard, and those that are automatically generated (such as error message Routines and timeout Routines).

Finally, you use the CallSuite Wizard main window to change your application to suit your needs. For example, you can further define existing Routines, such as rerecording the default greeting or changing which dialed digits will generate an error message. You can also add new Routines to your application, such as a GetDigits Routine that captures the digits a caller dials.

Warning:

If you generate an application using CallSuite Application Wizard, you cannot edit the generated code directly in the program language because any edits you make manually are wiped out when you save your work in the CallSuite Wizard.

To view the CallSuite Wizard, choose Start | Programs | Parity Software | CallSuite | CallSuite Wizard.

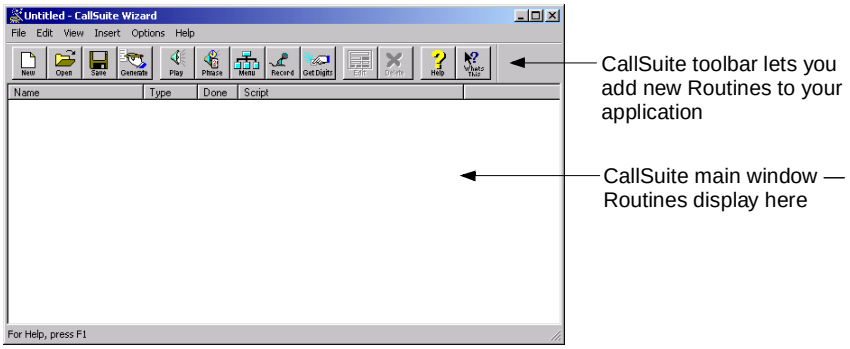

Figure 4-2 CallSuite Wizard—main window

For additional information, you can access the CallSuite Wizard help file from the CallSuite Wizard main menu.

This chapter includes two tutorials that use CallSuite Application Wizard and CallSuite Wizard. In Tutorial 1 you create an application in Visual Basic using CallSuite Application Wizard. In Tutorial 2 you make modifications to your application using the CallSuite Wizard.

Media Toolbox

Media Toolbox is a family of ActiveX components that you can use to further manipulate telephony sound files. After installing the utility you have access to a host of tools including a sound file converter component, a component that helps you create and edit Indexed Prompt Files (IPFs), a component that can query the capabilities of your installed Wave devices, and sample applications.

CallSuite Programming

As you develop and run applications using CallSuite, you use methods to send commands to Topaz, define properties to describe the attributes of the various objects you are using, and monitor events to find out what actions have occurred during development and runtime.

Methods

Methods are functions that implement most actions in your program. You send commands to Topaz requesting specific CT functions.

You can run CallSuite in synchronous or asynchronous mode. In synchronous mode, methods put the calling task or thread to sleep (this is called blocking) until the requested operation completes.

In contrast, when asynchronous mode is enabled, supported methods return immediately and indicate whether the operation started successfully while the operation continues in the background. Later, your application receives an event reporting that the operation has completed. Asynchronous mode is disabled by default. Many (but not all) CallSuite methods are affected by asynchronous mode; to find out which methods are affected, see the CallSuite online help.

Properties

A property is a data value similar to a variable and is generally used for attributes of an object that tend to remain fixed. In CallSuite for example, the ElapsedSecs property returns the number of seconds that elapsed during the last Play or Record. Properties are specified using the name of the object and the name of the property separated by a period. You set property values at runtime.

Events

Events are reports from the object to its client signifying that an action has occurred. Many events have corresponding methods that wait for a specified action to take place. For example, if an object displays a button on the screen, a typical event may report that the button was clicked. In CallSuite, you can process events by defining a function to be called when the event is triggered.

Tutorial 1: Generating a Starter Application in CallSuite

In this tutorial you use CallSuite Wizard and Visual Basic to create a simple outbound calling application that does the following:

- Makes an outbound call
- Plays a greeting when SimPhone answers the call, and gives this menu choice to the caller:

  - — Hear the company's stock price
  - — Contact Technical Support
- Performs the requested action or plays a message if an invalid option is chosen
- Lets the caller choose again until he or she hangs up or chooses to end the call
- Plays a goodbye message and hangs up

Before you Begin

Make sure you have CT ADE with CallSuite 8.0 (or higher), SimPhone, and Microsoft Visual Basic 6.0 installed on your system. If you have questions or want more information on CallSuite Application Wizard or the other wizard components, see the CallSuite Wizard online help. SimPhone is used in this tutorial to simulate a phone call to the voice card.

Step 1: Using CallSuite Application Wizard

In this section you use CallSuite Application Wizard to create a new application.

1. Start Visual Basic and close any open projects.
2. From the Add-Ins menu in Visual Basic, choose CallSuite Wizard v8.

   The CallSuite Wizard starts and displays the Introduction dialog.
3. Click Next to display the Step 1 dialog.
4. Select Outbound to create an outbound calling application. Then click Next to display the Step 2 dialog.
   a. In the Project Name field, type **VBSampleOutbound**.
   b. In the Project Path field, type the following path **C:\temp\VBSample**.
   c. Click Next to display the Step 3 dialog.

   Here we want to add a main menu in our application following the greeting. The starter application always begins with a greeting because most call processing applications begin with one such as:

   "Hello and thank you for calling..."

   —and follow with a main menu such as:

   "For this option, press 1, for this other option, press 2..."

   You can choose to add a main menu that has from two to nine options.

5. In the Step 3 dialog:
   a. Click Yes to add a menu to your application.
   b. Type 2 in the field below to specify two menu options.
   c. Click Next to display the Finished dialog.
6. Click Finish to create the project.

   CallSuite Wizard creates an outbound call processing application in the directory you specified in step 4. A message box tells you when CallSuite Wizard has created the Visual Basic project.
7. Click OK.

   Once the project is created, the Visual Basic Project Window shows the new form and module:

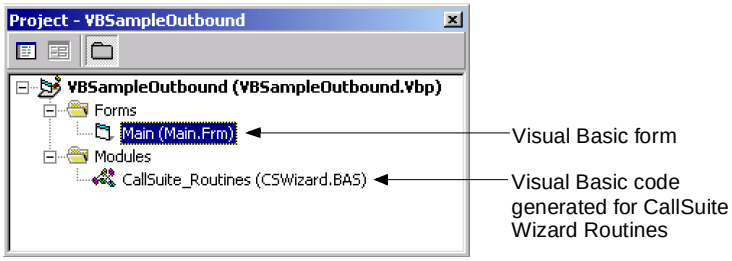

Figure 4-3 Project window in Visual Basic

Running the Application

Now you can run your new application.

1. Run the application from Visual Basic by choosing Run | Start.

   The Outbound with CPA dialog and the SimPhone main window appear.
2. From the Outbound Telephony Application dialog, click Dial.

   The Call Progress dialog displays. If you don't see this dialog, click anywhere on the SimPhone main window to bring it into focus on your desktop.
3. From the Call Progress dialog, click Connect, then click OK.

   The call is connected and a welcome message plays:

   "Hello, and thank you for calling Parity Software. Make your selection now, or press star to end this call."

Note that the Line State in the SimPhone main window, which is represented by a telephone, goes off hook when the call is answered.

4. From the SimPhone main window, press star to end the call.

   A goodbye message plays:

   "Goodbye, and thank you for calling."

   The application hangs up and the Line State in the SimPhone main window goes back on hook.

 *Caution:*

Dialing any digit other than the star key in step 4 results in an application error because a message hasn't been recorded for alternative choices yet.

---

5. From the Visual Basic main window, choose Run | End, and then click the close icon in the SimPhone main window.

Tutorial 2: Modifying Your CallSuite Starter Application

Now that you have created a project that contains a starter application, you can begin modifying existing routines and adding new ones. In this section you modify the Routines in your application and then test it.

 *Caution:*

You must use CallSuite Wizard to edit any Routine in an application generated using Callsuite Application Wizard. If you make changes directly to the code, they are erased the next time you open the project in CallSuite Wizard.

Before you Begin

In this tutorial you modify the application you created in Tutorial 1: Generating an Application in CallSuite. If you haven't completed the steps in Tutorial 1, go back and finish it before starting this tutorial. SimPhone is used in this tutorial to simulate a phone call to the voice card.

Step 1: Modifying the Main Menu Prompt Routine

In this section you modify the message in the Main Menu Routine.

1. Launch Visual Basic, and from the main window, open the VBSampleOutbound.Vbp project. Then choose Add-Ins | CallSuite Wizard v8.

   The Callsuite Wizard displays.

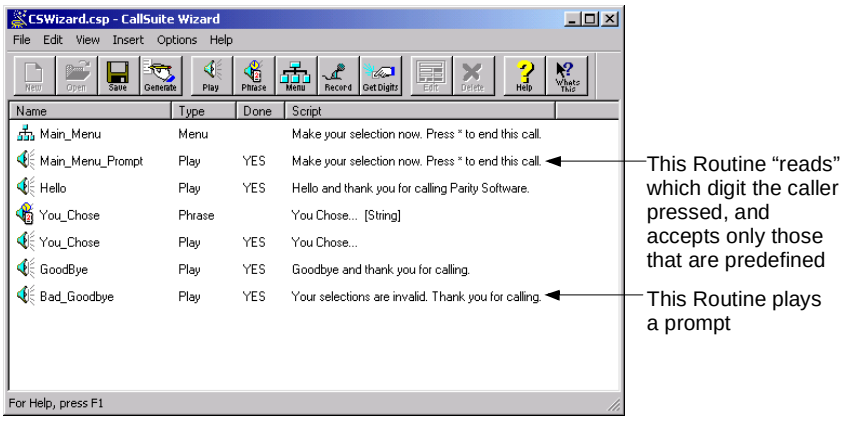

Figure 4-4 CallSuite Wizard—main window

Each item in the main window represents a Routine included in your VBSampleOutbound application. Let's view the properties of the Main Menu Routine.

2. From the CallSuite Wizard main window, double-click on Main_Menu to display the Menu Routine Editor.

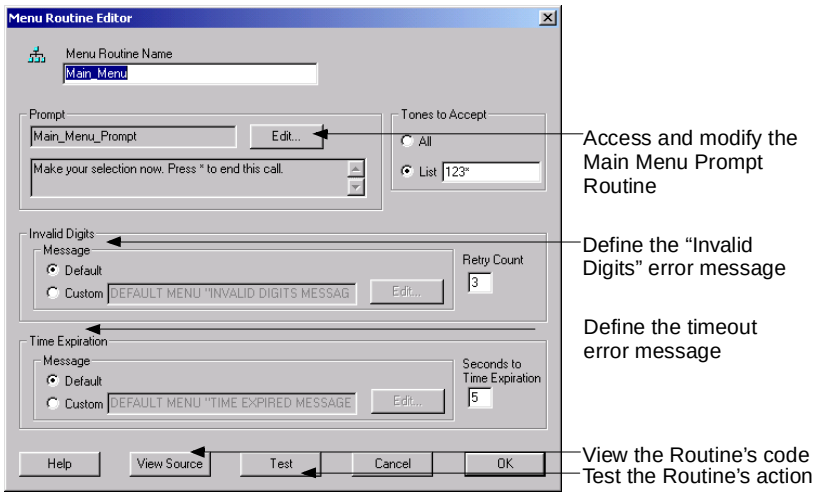

Figure 4-5 Menu Routine Editor dialog

From this dialog we can access the Main Menu Prompt Routine and modify it.

3. Click Edit.

The Play Routine Editor dialog displays. You can select another recording or create one of your own. We're going to create a new recording.

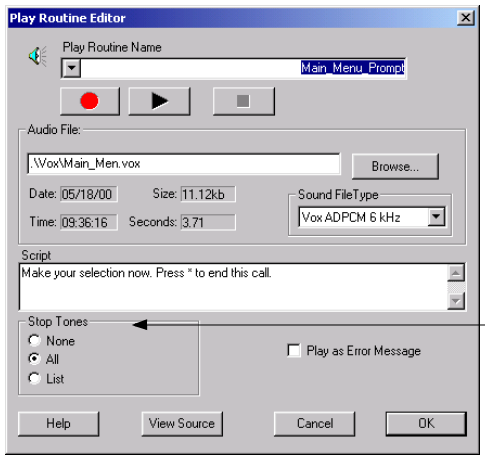

Figure 4-6 Play Routine Editor dialog

4. In the Script field, type **Press 1 to hear our stock price or press 2 to reach Technical Support. Press star to end this call.**

   SimPhone lets you record by using a microphone attached to your sound card.

5. Click Record (red dot) and record the message you typed. Then click Stop (black rectangle) when you are finished.

6. Review the message by clicking Play (the black triangle), and make new recordings until you are satisfied. When you are finished, click OK to close this dialog.

7. From the CallSuite Wizard main window, choose File | Save.

Step 2: Modifying the Hello Routine

Next, we want to change the Hello Routine so that it plays a different company name:

1. Double-click the Hello Routine to display the Play Routine Editor dialog.

2. In the Script field, type **Hello, and thank you for calling Precision Software.**

3. Click Record, say this message, and click Stop when you're finished. Review the message by clicking Play. If you're not happy with your message, make new recordings until you are satisfied.

4. Click View Source to see your changes reflected in the Routine's source code.

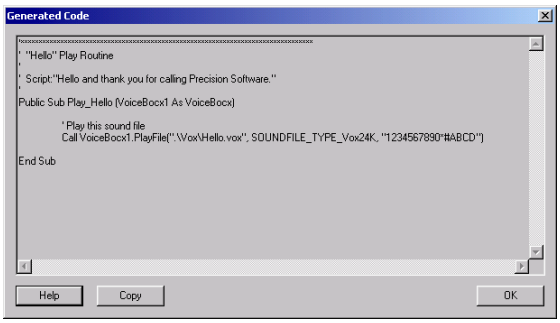

Figure 4-7 Generated Code dialog

5. Click OK to close the Generated Code dialog, and then click OK again to close the Play Routine Editor dialog.
6. From the CallSuite Wizard main window, choose File | Save.

Step 3: Editing and Creating Routines for the Menu choices

In Tutorial 1 we selected two menu choices for callers to choose from. Now we need to record messages for those choices. First, let's edit the Play Routine for the Technical Support menu option.

1. From the CallSuite Wizard main window, click the Play Routine to display the Play Routine Editor dialog.
   a. In the Play Routine Name field, type **Tech_Support**.
   b. In the Script field, type **You chose Technical Support.**
   c. Record this message, then click OK to save your changes and close the dialog.

Now let's create a Phrase Routine for the Stock Price. In order to play a message that contains variable data, like dates or dollar amounts, you need to build the elements of the phrase. When the caller chooses to listen to the stock price, a message like this plays:

"Our company's stock price was $256.84 May 16, 1998."

The elements in this phrase consist of: two Play Routine— "Our company's stock price is" and "on" , and two variables—a Date variable and a Money variable.

2. From the CallSuite Wizard main menu, double-click the "You Chose" Phrase Routine.

   The Phrase Routine Editor displays.

3. In the Phrase Routine Name field, type **Stock_Quote**.

Note, if you type a space in this field, CallSuite Wizard automatically inserts an underscore in its place.

4. In the Phrase Elements window, select the String element ABCD and click Remove Element to delete it.
5. In that same window, double-click the Play Routine called "You Chose."

   The Play Element Editor dialog displays.

   a. In the Play Routine Name field, type **On**.

   b. In the Script field, type **On**.

   c. Record this message, and then click OK to close the Play Element Editor.
6. From the Phrase Routine Editor, click Date, then double-click on the new Date element.

   The Insert Date Element dialog appears, showing this format: 20011010.

   a. Click Play to hear how the date is said during play time, then click OK to close this dialog.
7. Click Play Routine.

   The Play Element Editor dialog displays.

   a. In the Play Routine field, type **Our_Stock_Price**.

   b. In the Script field, type **our company's stock price is**.

   c. Record this message, then click OK to save your changes and return to the Phrase Editor.
8. Finally, click Money to insert a Money element.

   In the Script field at the bottom of the Phrase Routine Editor dialog, you can see the order in which the elements play during play.

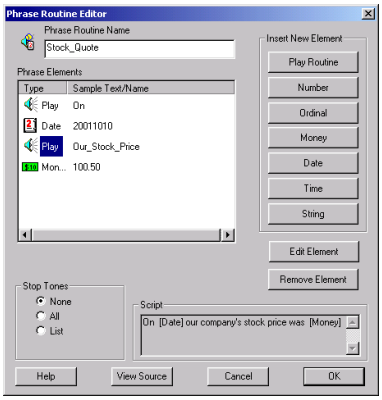

Figure 4-8 Phrase Routine Editor dialog

9. Click Play from the Play Routine Editor to hear the Phrase with a sample date and price.
10. Click OK to return to the main window in the CallSuite Wizard, and then choose File | Save.

Step 4: Editing the Goodbye Routine

Now we want to change the Goodbye message so that our company name is mentioned again.

1. From the CallSuite Wizard main window, double-click on the Goodbye Routine.

   The Play Routine Editor displays.

2. In the Script field, type **Goodbye, and thank you for calling Precision Software.**
3. Record the message you typed. When you are finished, close the Play Routine Editor.

Step 5: Changing the Bad Goodbye Routine

If the caller repeatedly presses the wrong digit or no digit at all, the Bad Goodbye message plays and then the application hangs up. We want to change the Bad Goodbye message so that it matches your voice like the other prompts.

1. Double-click on the Bad Goodbye Routine.

   The Play Routine Editor displays.

2. In the Script field, type **Your selections are invalid. Thank you for calling Precision Software.**
3. Record the message you typed. When you're finished, close the dialog.

4. From the Visual Basic main window, choose File | Save Project.

Step 6: Changing the Default Error Prompts

Callers will hear error messages if they press digits that are not predefined to perform a Routine or if they press no digits at all. If an invalid digit is pressed, this Menu Invalid Digit message plays:

"That is not a valid response."

If no digits are pressed at all, a Menu Time Expired message plays:

"No response received."

Both of these are "default error messages" that were automatically included in the generation of your application. We want to rerecord these messages so that they play in your voice.

We are going to rerecord the Menu Invalid Digit message first.

1. From the Visual Basic main window, choose Tools | CallSuite Wizard v8 to display the CallSuite Wizard.
2. From the CallSuite Wizard main window, choose Options | Default Messages | Menu Invalid Digit Message.

   The Routine Editor dialog for this Routine displays. We still want the script to say:

   "No response received."

   —so we do not change the script text, but we do want the message to play in a different voice.
3. Record the message in the script field, then click OK to save your changes and close the dialog.

Now we are going to rerecord the "No response received." message.

4. From the CallSuite Wizard main window, choose Options | Default Messages | Menu Time Expired Message.
5. The Routine Editor dialog for this Routine displays.
6. Rerecord the message in the Script field, then click OK to save your changes and close the dialog.
7. Choose File | Save to save your changes and then close the CallSuite Wizard main window.

Testing Your Application

Now it's time to test your application.

1. From the Visual Basic main window, choose Run | Start.

Visual Basic compiles the application, and then runs it. The SimPhone main window and the Outbound Telephony Application dialogs display.

2. Click Dial from the Outbound Telephony Application dialog.
3. On the Call Progress Analysis Results dialog, click Connect.

   You should hear your newly recorded welcome message and prompt asking you to make a selection.
4. Dial 1 from the SimPhone main window to hear the response in your voice.
5. This time, when you are prompted, dial 9 to hear the invalid response message in your voice. Then press star to end the call.
6. When you are finished, choose Run | End from the Visual Basic main window and close Visual Basic.