

# CONTAX IVR Technical Guide

User's Manual  
First Edition

(C) 2002-2004 Eletech Enterprise Co., Ltd.  
All Rights Reserved.



# Contents

<b>Unit 1: Introduction .....</b>	<b>4</b>
<b>Chapter 1: What is Contax IVR?.....</b>	<b>5</b>
1.1 What is an IVR? .....	5
1.2 What makes Contax IVR stand out from other similar products? .....	5
<b>Chapter 2: Installing Contax IVR.....</b>	<b>7</b>
2.1 System requirements .....	7
2.1.1 <i>Installation Steps</i> .....	7
2.1.2 <i>First Impressions: CONTAX IVR</i> .....	10
<b>Unit II – Contax IVR Flow Designer .....</b>	<b>13</b>
<b>Chapter 3: Main Window .....</b>	<b>13</b>
3.1 (renumber) Cells, Controls, Properties and Variables .....	14
3.2 The Side Toolbar.....	15
3.2.1 <i>Flow Control Cells</i> .....	16
3.2.2 <i>Media Control cells</i> .....	18
3.2.3 Top Toolbar .....	19
3.3 Getting the Hang of It: Exploring Flow Designer.....	20
3.3.1 <i>Creating Cell icons</i> .....	20
3.3.2 <i>Select and Deselect Cell Icons</i> .....	21
3.3.3 <i>Deleting cell icons</i> .....	22
3.3.4 <i>Place Link Trace between 2 Cell Icons</i> .....	22
3.3.5 <i>Delete a Trace Line between 2 Cell Icons</i> .....	23
3.3.6 <i>Moving Selected Icons</i> .....	23
3.3.7 <i>Cut/Copy Cell Icons and Link Traces</i> .....	24
3.3.8 <i>Paste Cell Icons and Link Traces</i> .....	24
3.4 Property Table Pane .....	25
3.4.1 <i>Drop-Down Cell List</i> .....	27
3.5 Variable Table Pane.....	27
<b>Unit III-Basic Application Development .....</b>	<b>28</b>
<b>Chapter 4: Your First Application .....</b>	<b>28</b>
4.1 Break it Down .....	28

4.1 Taking the Plunge .....	29
4.1.1 Application No.1: Greeting .....	29
4.1.2 Saving the Finished Application .....	32
4.1.3 Proof of the Pudding: Simulating Your Call Flow .....	32
4.1.4 Recording Your Message .....	34
4.1.5 Using Text-To-Speech .....	35
<b>Chapter 5: Choice and Consequence .....</b>	<b>37</b>
5.1 Conditions and Loops .....	37
5.1.1 Application No. 2: Automatic Call Transfer .....	37
5.1.2 Application No. 3: Doing The Same Thing Differently .....	41
5.1.3 GetKeys versus GetDTMF .....	42
<b>Unit IV – Advanced Call Flow Design .....</b>	<b>50</b>
<b>Chapter 6: Subflow: Keeping Complex Things Simple .....</b>	<b>51</b>
6.1. What is a Subflow? .....	51
6.1.1 Application No. 4-Automatic Transfer with Voice Mail .....	51
6.1.2 Saving and Naming Subflows .....	53
6.1.3 Building a Library of Subflows .....	54
<b>Chapter 7: Variables and Expressions .....</b>	<b>56</b>
7.1 Variable Table Pane .....	56
7.1.1 Adding Variables to The Variable Table .....	56
7.1.2 Assigning a Variable .....	57
7.1.3 Using the Expression Builder .....	57
7.1.4 Local and Global Variables .....	60
7.1.5 Application No. 5-Another Variant of AutoTransfer .....	60
<b>Chapter 8: Working with Databases .....</b>	<b>65</b>
8.1 What is a Database? .....	65
8.2 Creating a Database .....	66
8.3 Harnessing the Power of a Database .....	68
<b>Chapter 9: Working with Fax, E-mail and SMS .....</b>	<b>71</b>
9.1 Sending Faxes .....	71
9.2 Application No: Fax on Demand .....	72
9.3 Receiving a Fax .....	74
9.4 Working with E-Mail .....	75

9.4.1 Application No. 7: Send Mail.....	76
9.4.2 Receiving Email .....	77
9.4.3 Application No. 8: Receive Email.....	78
9.5 Sending Information using SMS (Short Messaging System).....	80
9.6 Web Integration.....	82
9.7 Other Control Cells.....	83
9.7.1 Conference.....	83
9.7.2 Switch.....	84
<b>Unit V-Program Deployment and Monitoring .....</b>	<b>85</b>
<b>Chapter 10: Program Deployment .....</b>	<b>85</b>

# Unit I-Introduction

## Guide to Developing Applications for Contax IVR

Welcome to **CONTAX IVR**, the next-generation Interactive Voice Response (IVR) system that allows you to create customer service-oriented applications combining voice, fax, email and SMS. This guide lets you harness the power of **CONTAX IVR** even if you don't have any previous programming experience. On the other hand, seasoned programmers will also find this guide useful in helping them speed up their understanding and familiarity of **CONTAX IVR** including the elements and principles needed to design, test, and deploy **CONTAX IVR** applications.

This guide follows the natural skill-learning process that you experienced in learning most of what you know now, from learning how to read, how to ride a bike, how to drive, or how to swim. You have learned these skills because somebody helped you do so. Somebody read the letters and the words for you and told you what the words meant--- and then you repeated until you yourself can pronounce the words. Somebody helped you ride a bike and pushed and told you to pedal, until you can keep it running straight and without falling.

Similarly, this guide walks you through the paces of creating simple applications, using the basic programming elements in **CONTAX IVR**. In the early chapters of this guide, step-by-step instructions of how to create **CONTAX IVR** applications are presented. However, the instructions would become less detailed as you become more familiar with the elements and procedures involved.

You would run the applications, test them to find out what works and what makes them fail to do so. After getting to try all the examples provided, you would have acquired the basic skills and knowledge to design **CONTAX IVR** applications on your own.

# Chapter 1: What is Contax IVR?

## 2.1 What is an IVR?

IVR stands for Interactive Voice Response; an automated telephone information system that speaks to the caller using a combination of fixed voice menus and real-time data from databases. The caller responds by pressing digits or keys on the telephone. More advanced IVR systems also allow the user to interact using words or short phrases through a technology known as Automatic Speech Recognition (ASR).

IVR systems allow callers to get the information they need 24 hours a day, 7 days a week. They are also used to save costs on personnel by eliminating the need for people to constantly answer simple, repetitive questions.

Most IVR systems are run in personal computers (PCs) equipped with special voice boards used to process voice signals from its telephone interfaces. Voice boards also contain DSP (digital sound processor) chips. These specialized processors connect to the telephone system, which actually switches the calls. IVR systems may also be networked on LANs (Local Area Networks) and Was (Wide Area Networks).

Usually a PC-based IVR system is connected to one or more analog extensions on the corporate phone system private branch exchange (PBX). A Web-based IVR system may also connected to the Internet. A voice board can also interact with a PBX by; for example, sending flash hooks and dials digits, detecting hang-up signal, and notifying termination of a call. It can also answer, initiate, and terminate a call.

## 2.2 What makes Contax IVR stand out from other similar products?

The vast majority of IVR systems on the market today support only voice. Some of the more advanced systems may support email as well. However, **CONTAX IVR** *is truly the next-generation IVR, allowing users to create applications that combine the power of voice, fax, email, SMS and seamless database access in innovative and cost-effective ways.* **CONTAX IVR** also provides text-to-speech capabilities, allowing your IVR applications to “read out loud” documents to callers.

\*\*\*\*\* Show comparison by picture of Contax ivr vs other ivrs

In addition, **CONTAX IVR** provides an easy-to-use visual tool for designers to create, and for administrators to monitor and manage IVR applications. If you are a developer, you can simply click and drop cell icons from the toolbar into the design pane and click on the pin of cell icons to connect them to form desired graphical call flow charts. After this simple process, you can run simulation to verify, trace, debug and/or edit the call flows you have designed.

\*\*\*\*\* Show picture of drag/drop simplicity

You can easily configure **CONTAX IVR** according to your needs, whether application-based, web-based, or both. The **CONTAX IVR** Flow Designer is where you can design, edit, and test IVR applications on stand-alone computers before copying finished applications to servers that would run it.

**CONTAX IVR** supports voice boards that are TAPI (Telephony API) compliant. TAPI is an application-programming interface developed by Microsoft and Intel to allow for computer telephony integration. **CONTAX IVR** can also be configured to work with PBX systems to provide transfer and conferencing capabilities.

## Chapter 2: Installing Contax IVR

### 2.1 System requirements

The following are needed to run **CONTAX IVR**:

- Intel Pentium III 450 MHz/128 MB RAM/10 MB HD Space
- Microsoft Windows 2000 (Professional or Server) and IIS 5.0 or higher
- Analog Voice boards with TAPI driver

Fax modems with COM port driver – for fax applications

Microsoft SAPI 5.0 (standard in Windows XP, provided with **CONTAX IVR** CD)

PBX system – for call transferring and conferencing features

#### **2.1.1: Installation Steps**

Insert **CONTAX IVR** CD in your CD-ROM drive.

Double-click **setup.exe** file to start the installation process. **CONTAX IVR** v2.1 install wizard will then appear. Follow the instructions to complete your installation.

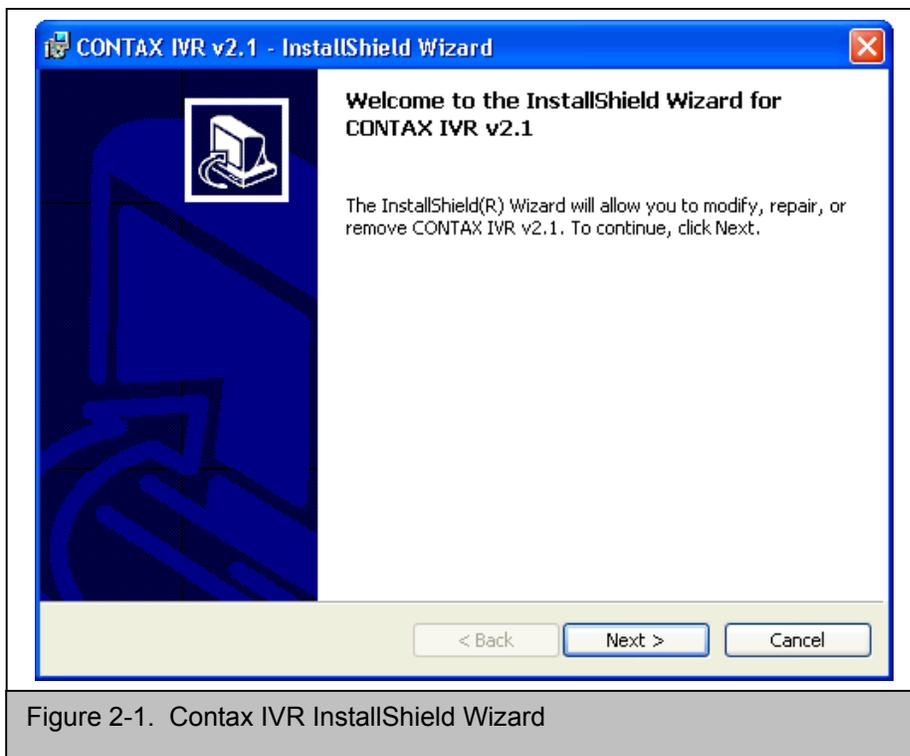


Figure 2-1. Contax IVR InstallShield Wizard

*After you click Next, you will see the following:*

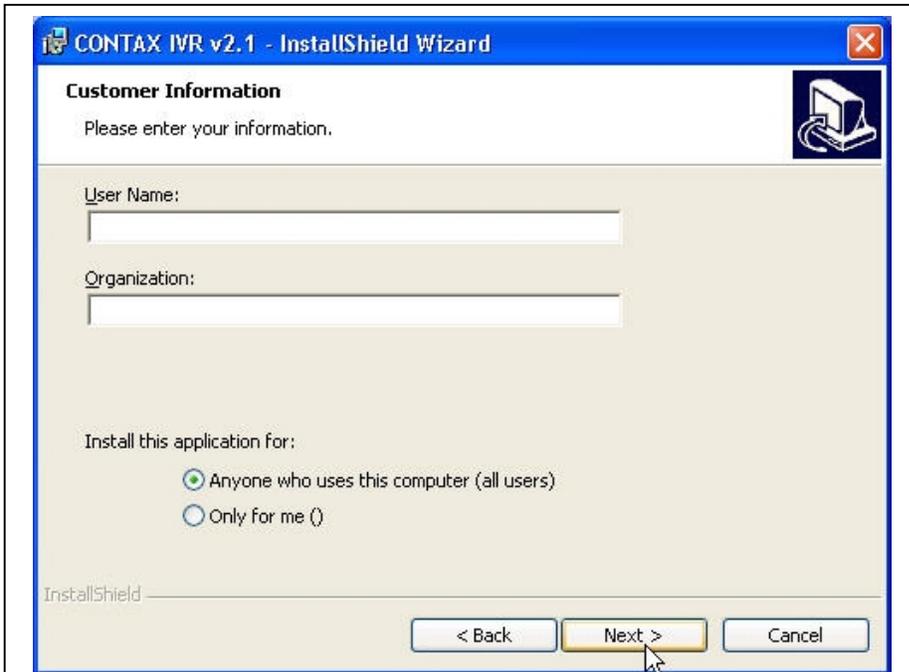


Figure 2-2. Contax IVR InstallShield Wizard

Type in the corresponding blank spaces your name and your organization or business. Next, you will be presented with a choice on where to install **CONTAX IVR**. The default directory is "C:\Program Files\Contax\IVR 2.1\" but you may install it in a directory of your choice by clicking the "Change" button.

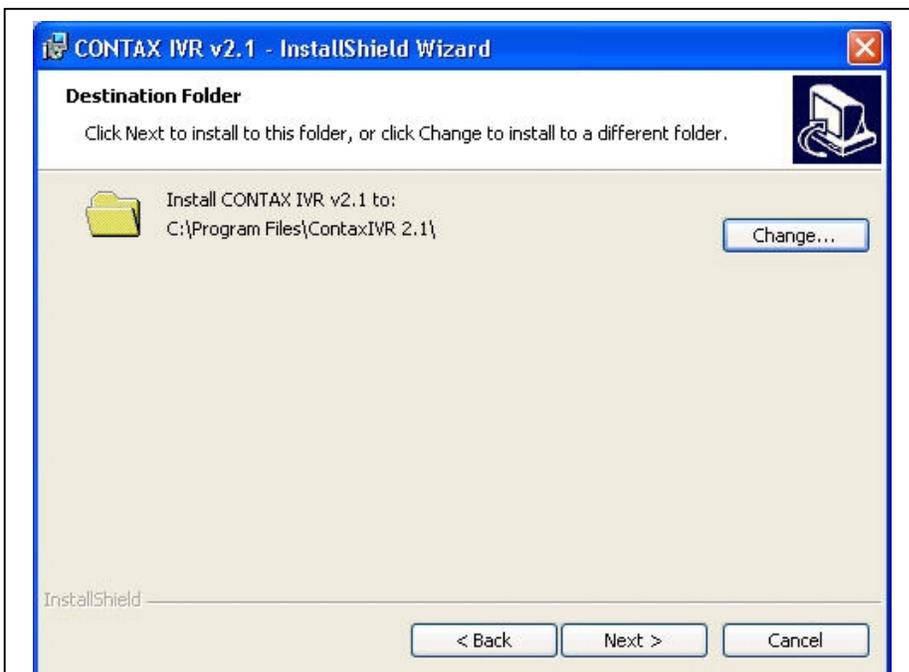


Figure 2-3. Contax IVR InstallShield Wizard

After you have decided on the installation directory of **CONTAX IVR** click the “Next” button and the program will install, as indicated in the box below:

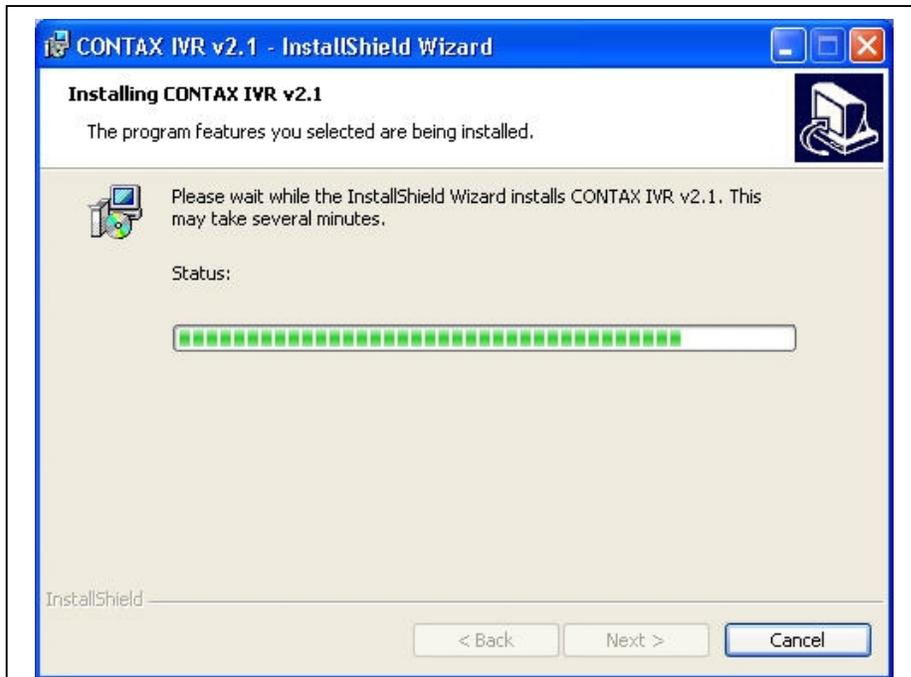


Figure 2-4. Contax IVR InstallShield Wizard

*When the installation is complete, you will see the following:*



Figure 2-5. Contax IVR InstallShield Wizard

## 2.1.2 First Impressions: CONTAX IVR

To enable you to get a better idea of how **CONTAX IVR** works, why not try one of the sample applications included in the installation CD.

First, run **CONTAX IVR** Flow Designer. Click **Start >Programs>ContaxIVR2.1>Flow Designer**. Inside the Flow Designer click **File | Open** from the drop-down menu. When the dialogue box opens, browse into the installation directory of **CONTAX IVR** Click on the **Transfer** folder and then click **Open**. You will see a file named **Transfer.IVX**. Click on this file and then click **Open**.

When the Tranfer.IVX call flow opens, you will see this:

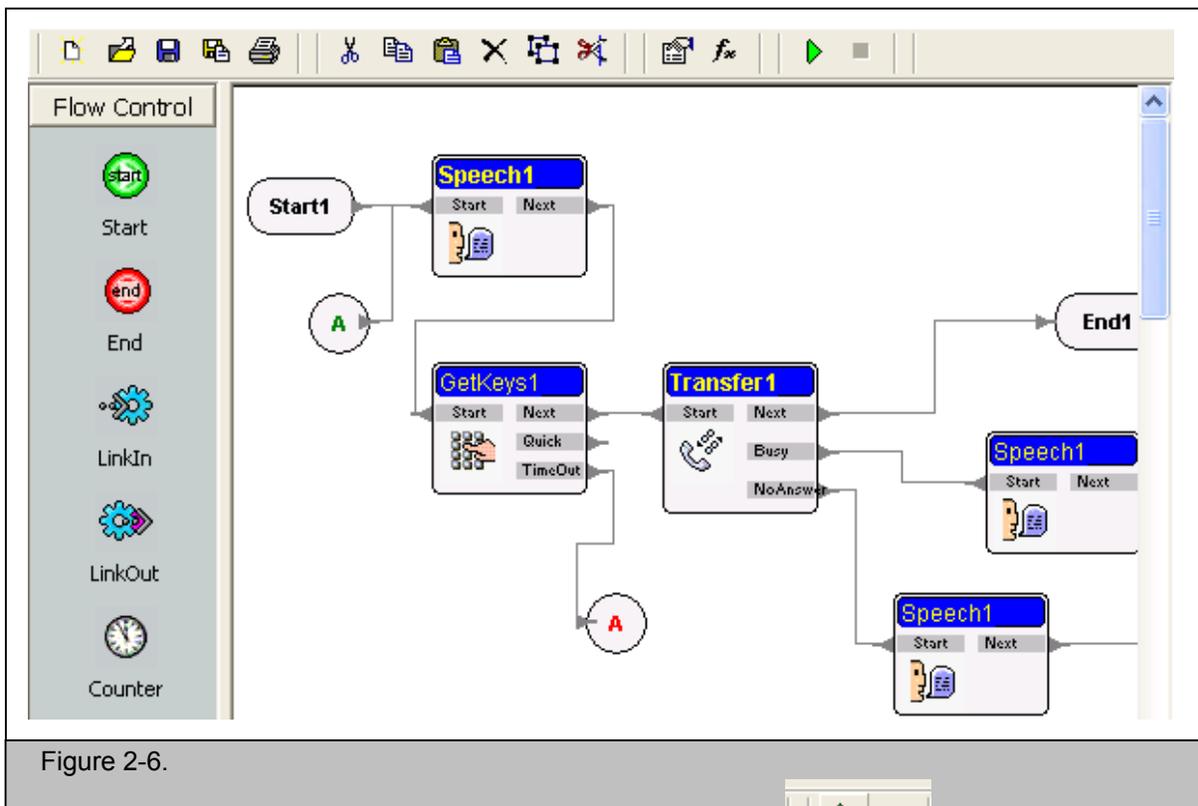


Figure 2-6.

Now, click on the **Run** button, the green triangle in the top panel. The program **Simulator** (see figure below) will open.

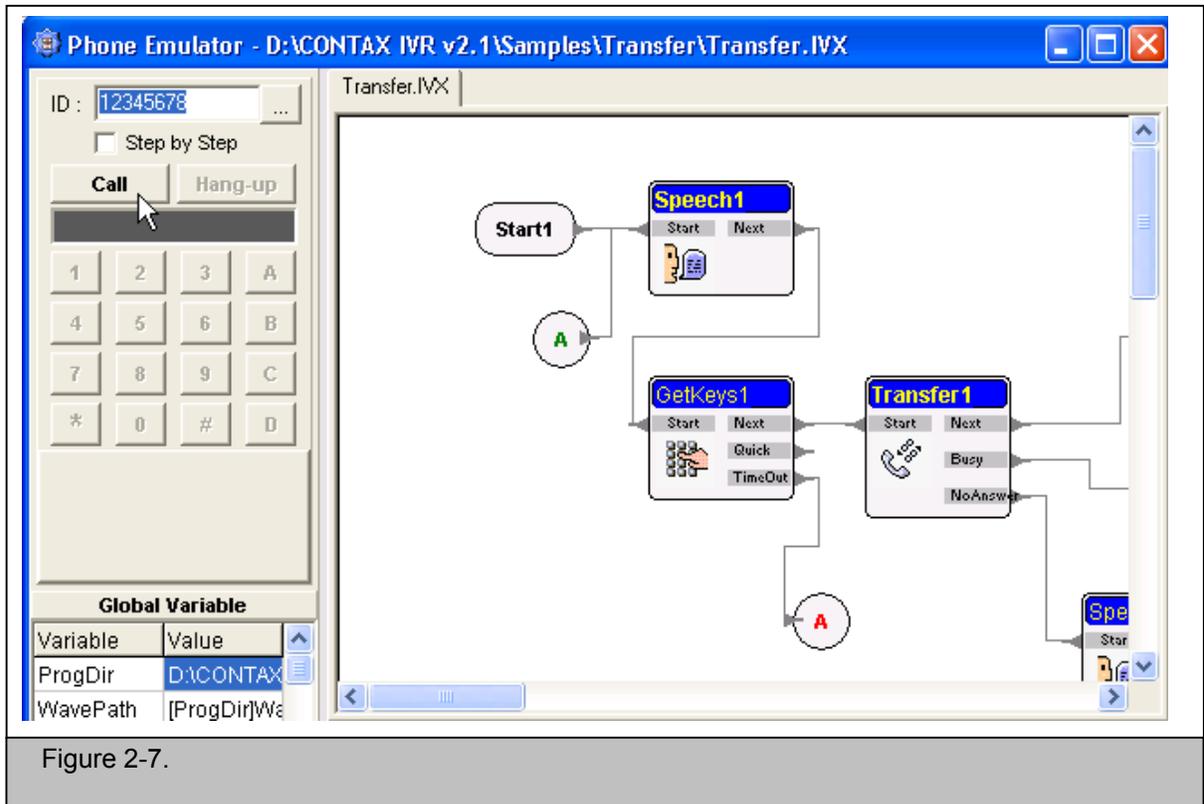


Figure 2-7.

Press the **Call** button. Transfer.IVX call flow will now execute, sensing that a call has come through from one of its channels. You will hear the greeting: “Welcome to Eletech. Please dial the extension number.” Note that when the message is playing, the **Speech1** module turned yellow, indicating that it is the active module. You will know that a module is executing a process when it turns yellow.

If you don't do anything, the greeting will be repeated. Press any of the buttons in the “keypad”, the number **1** for example.

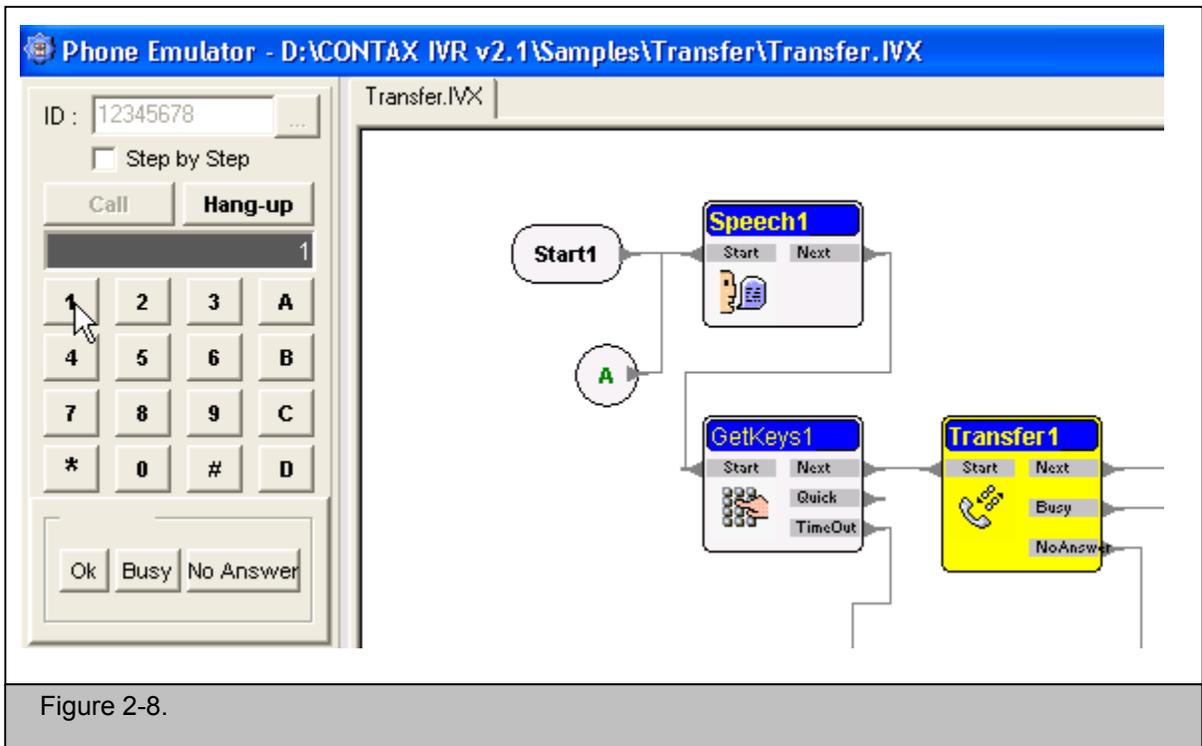
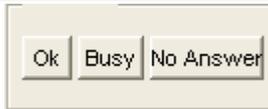


Figure 2-8.



If you press the **Busy** button in the keypad, you will hear the greeting, “Sorry, line is busy now.” The greeting will start again. This time, after pressing **1**, press the **Ok** button instead. The call will end, indicating the call flow has transferred the connection to the desired number.

That’s how a **CONTAX IVR** call flows handle calls. In the following units and chapters, you will learn how to use the **Flow Designer** to make call flows in accordance with specific needs.

# Unit II – Contax IVR Flow Designer

## Chapter 3: Main Window

Click **Start** -> **Programs** -> **CONTAX IVR v2.1** -> **FlowDesigner**. **CONTAX IVR** The Flow Designer main window will appear. You will notice that there are two toolbars, one along the top and another along the left side.

There are also three panes in the main window: the Design pane, the Property table pane, and the Variable table pane.

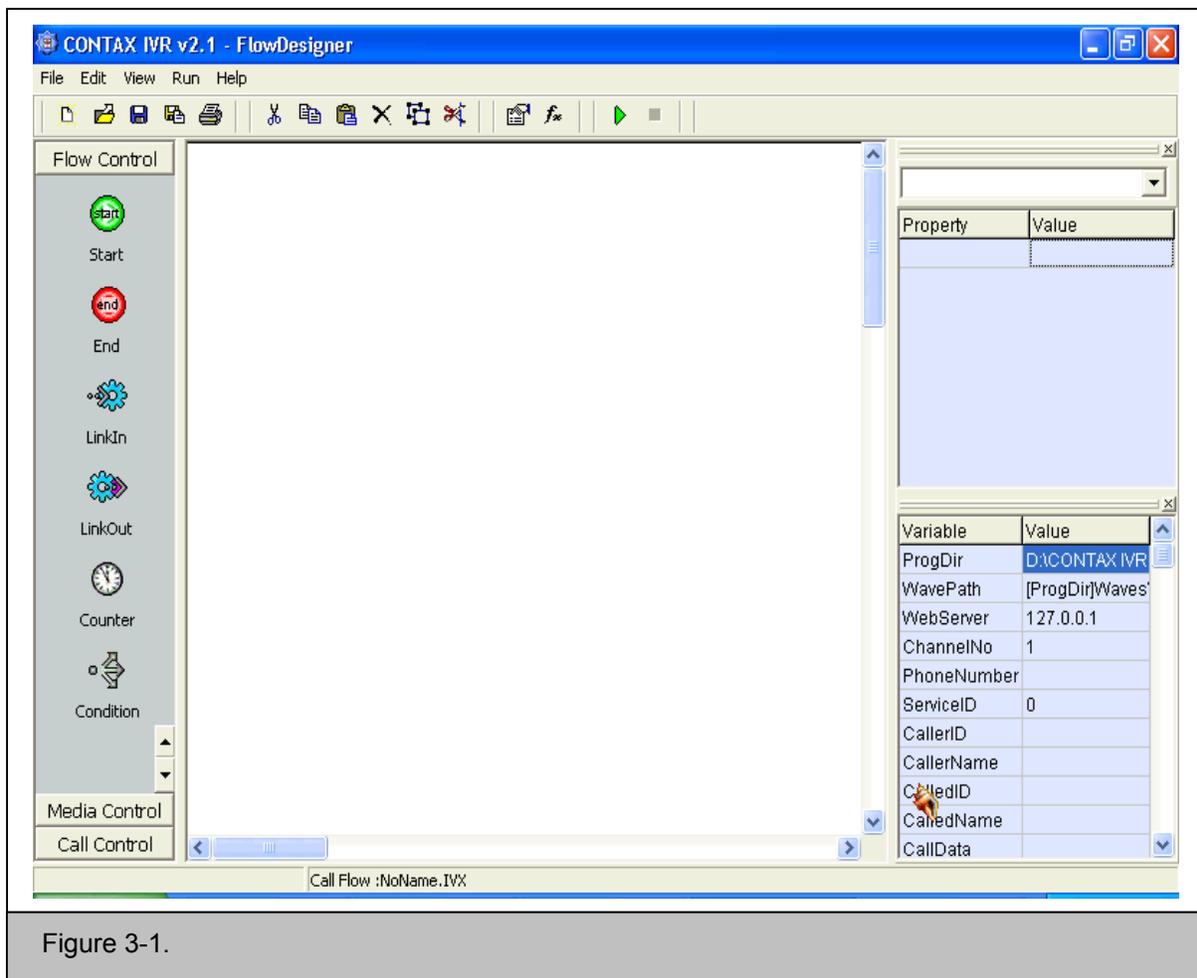


Figure 3-1.

You can adjust the size of each pane by moving the mouse pointer to the border between two panes then dragging and dropping the border line to your desired position while the mouse

pointer turns to one of the following.  

### **3.1 (renumber) Cells, Controls, Properties and Variables**

If you have programmed in Visual Basic before, then you will feel right at home with IVR. In **CONTAX IVR** a **cell** refers to a pre-defined set of actions represented by a single box. Contax IVR provides a large number of cells that do everything from playing a .WAV file, accepting user touch-tone input, to sending faxes and email. Cells are also known as controls, and the 2 terms are used interchangeably throughout this guide.

Each cell contains several properties. A property is a characteristic of a cell, such as the cell's name, the .WAV file to play, etc. Properties make creating IVR applications easier because it allows you to customize those characteristics of a cell that is specific to your requirements, while the rest remain the same. Therefore, creating **CONTAX IVR** applications mainly involve choosing cells, connecting them, and changing its properties to achieve the desired outcome.

Variables are placeholders that allow you to store pieces of information. You will learn more about variables in a later chapter, "**Variables and Expressions**".

### 3.2 The Side Toolbar

The toolbar along the left side of the main window contains tool cells for the designer to select.

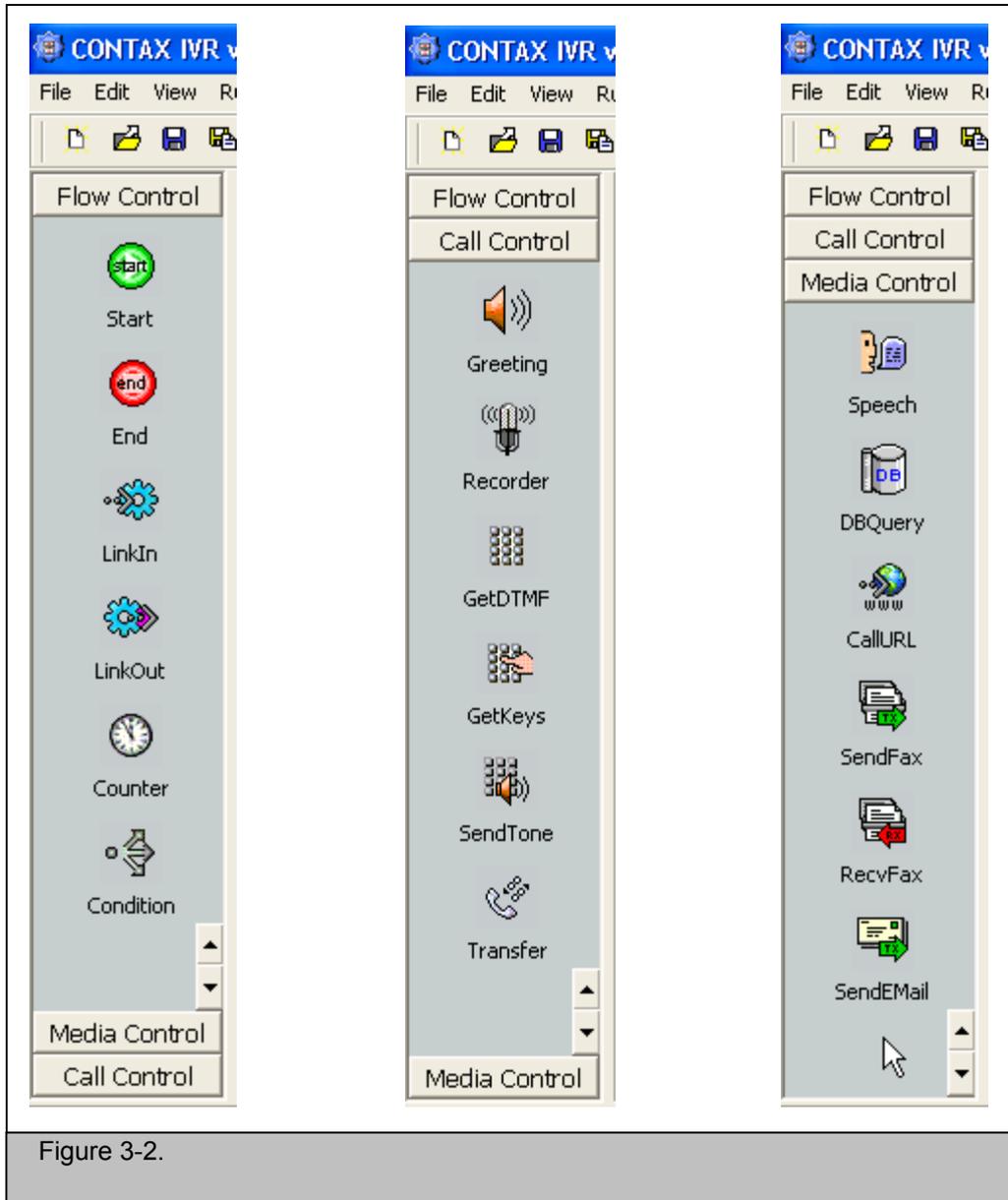


Figure 3-2.

There are **Flow Control**, **Call Control**, and **Media Control** cells. Flow Control cells are used to determine how your IVR application will run. Call Control cells are used to interact with the telephone. Media control cells are used for adding voice, fax, SMS, email and database capabilities to your IVR application. The default main window displays the 7 **Flow Control** cells as shown on the above left picture.

Click the **Call Control** button to switch to view the **Call Control** cells. To view more **Call Control** cells, just click the down arrow at the bottom right corner of the side toolbar as shown on the above middle picture.

Click the **Media Control** button to switch to view the **Media Control** cells as shown on the above right picture.

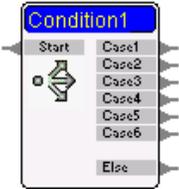
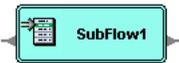
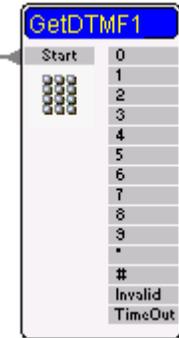
Click the **Flow Control** or **Call Control** button to switch back to view the **Flow Control** cells or the **Call Control** cells.

You can select one of the cells by clicking it from the side toolbar, and while holding down the mouse button, moving the mouse pointer towards the design pane, then releasing the mouse button to place that cell at any point on the design pane. This operation is often called drag-and-drop. Then, an icon for that cell appears on the design pane with the cell icon's upper left corner positioned at the insertion point.

The table below lists a summary of the cell buttons on the side toolbar, the corresponding cell icons, and their function descriptions:

### Flow Control Cells

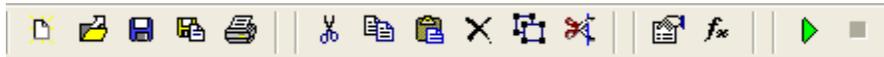
Cell	Button	Icon	Function Description
Start			Starts a call flow.
End			Ends a call flow.
Linkin			Goes to <b>LinkOut</b> cell with the same name in the same call flow.
Linkout			Comes from <b>LinkIn</b> cell with the same name in the same call flow.
Counter			Counts a loop

Condition			Executes on condition.
Sub-flow			Counts a loop.
Sub-flow			Goes to a sub-flow (another call flow) in the same channel line.
<b>Call Control cells</b>			
Greeting			Plays a greeting.
Recorder			Records an audio message as a (.wav) file.
GetDTMF			Gets a single touch-tone input (DTMF) from a (caller's) phone keypad.
GetKeys			Gets a certain number of touch-tone (DTMF) inputs from a (caller's) phone keypad.
SendTone			Sends DTMF tones.

Transfer			Transfers a call to another extension via a PBX.
Conference			Connects a call with another extension via a PBX.
MakeCall			Calls via another channel to execute a new call flow.
Switch			Connects two channels.
<b>Media Control cells</b>			
Speech			Converts text to speech and reads it to the caller.
DBQuery			Executes a database operation.
CallURL			Links to a web page using defined URL.
SendFax			Sends a fax.

RecieveFax			Receives a fax.
SendEMail			Sends an email.
RecieveEMail			Receives an email.
SendSMS			Sends a Short Message (Text Message).

### 3.3 Top Toolbar



The toolbar at the top of the main window has several buttons.

These buttons have something to do with File operations:

	Opens a new IVR file
	Opens an existing IVR file.
	Saves the current IVR file.
	Saves the current IVR file as another filename.
	Prints out the IVR call flow layout as a document.

The following buttons are used to **Edit** the applications you are designing:

	Cuts the selected cell icon(s), its property table(s), and link trace(s) on the design pane.
	Copies the selected cell icon(s), its property table(s), and link trace(s) on the design pane.



Pastes the copied cell icon(s) and link trace(s) to the position at 20 pixels lower and right of the original cell icon(s). The cell's property table(s) is/are also be pasted.



Deletes the selected cell icon(s) and link trace(s) on the design pane as well as its/their property table(s) on the property table pane.



Selects All (in the Edit drop-down function list).



Deletes a trace line between two cell icons.

These two buttons are used to toggle the **View**, meaning, show or hide tables:



Shows properties Table.



Shows Global Variable Table.

And these are the buttons used to **Run**, or stop the application:



Runs the current call flow.



Ends the current call flow.

### **3.4 Getting the Hang of It: Exploring Flow Designer**

As in any learning process, the best way to learn is to just do it. Let us begin by doing the common tasks:

#### ***3.4.1 Creating Cell icons***

From the side toolbar, click on a cell button once, move the mouse pointer towards the design pane, you will notice that the mouse pointer turns to a small cell symbol on the design pane as shown below:



Move your mouse pointer to a desired position (which will be an insertion point) on the design pane; click your mouse again, the desired cell icon will appear with its upper left corner at the insertion point.

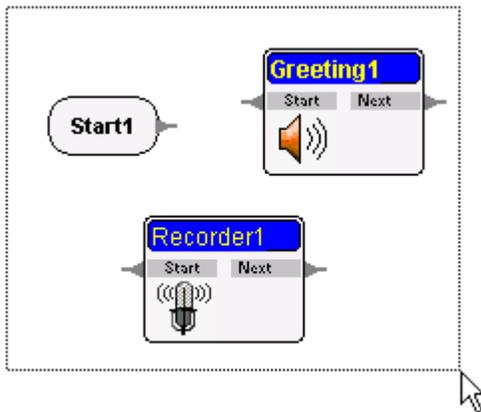


### 3.4.2 Select and Deselect Cell Icons

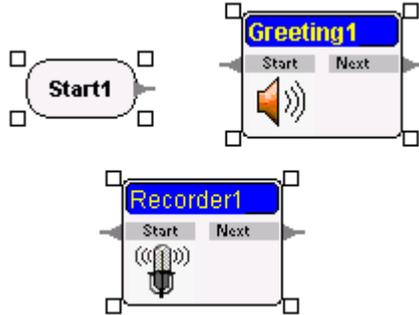
On the design pane, after a cell icon is placed or it is selected by clicking on it, you will see 4 small square blocks at its corners (as shown above). To deselect a cell icon, you need to click on a point on the design pane anywhere outside the cell icon's imaginary rectangular area defined by its 4 corners.

You can also search and select a cell icon from a number of cell icons in a call flow by clicking its caption name in the cell drop-down list on the property table pane. To select multiple cell icons, you can do it with either way of the following:

1. Click on a point outside the cell icons, drag the pointer to make a dot sided rectangular to contain all the cell icons you want to select, then release your mouse.



2. Click to select one of the cell icons, its 4 small square block corners appear. Move your mouse pointer to another cell icon, hold down the control key <Ctrl> and then click on the cell icon. Continue move your mouse pointer and hold down the <Ctrl> + click on the next cell icon until all cell icons you want to select are selected.



### 3.4.3 Deleting cell icons

1. Select the cell icons you want to delete.
2. Click the delete button on the top toolbar.

The cell icon and the property content in its property table will be deleted. All trace lines connected to this cell icon will be deleted too.

### 3.4.4 Place Link Trace between 2 Cell Icons

A cell icon represents a set of predefined actions. Most cell icons have pins in both sides. Cell icon **Start** and **LinkOut** have only right side pin. Cell icon **End** and **LinkIn** have only left side pin. Left side pin is the entrance of a cell icon. Right side pin or pins is/are the exit or exits of the cell icon. Depending on conditions, the program will exit from one of the right side pins during runtime.

Connecting a cell icon's right side pin to another cell icon's left side pin means linking two program sets with the right cell being executed after the left cell.

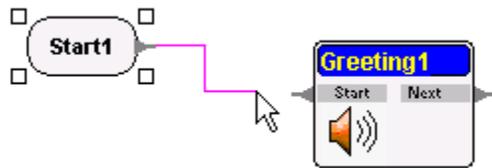
\*\*\*\*\* 2 icons execute first, then next

Every cell icon has one or several pins for its connection with other cell icons. The pin area includes pin itself and the dark gray pin text area shown below. Note that some cell icon's pin areas have pin itself only and have no pin text area with them (e.g. **Start**, **End**, **LinkIn**, **LinkOut**, and **SubFlow**).

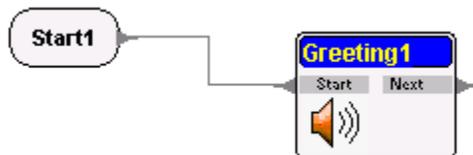


To connect pins between two cell icons, click the pin area at the right side of one cell icon, move your mouse pointer towards the pin area at the left side of the other cell icon. A pink bent line will

follow your mouse pointer trace. When your mouse pointer touches the pin area of the other cell icon, click your mouse again.



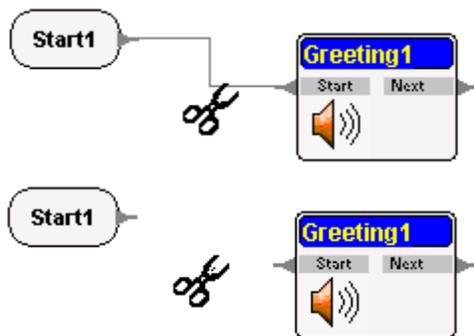
You will see that the pink trace line be fixed and turned to gray as shown below. Thus the two cell icons are connected via their respective specified pins.



If the pink line appears (perhaps by your unintentional click) and you do not want to connect it to any other cell icons, before it turns fixed and gray, just click one more time at any point free from any left side pin area on the design pane, the pink line will disappear.

### 3.4.5 Delete a Trace Line between 2 Cell Icons

Click the Delete Trace Line button on the top toolbar. You will see the mouse point turns to a pair of small scissors on the design pane. Move the scissors mouse pointer to touch the trace line you want to delete. Click on the trace line, and the trace line will be deleted as shown below:



### 3.4.6 Moving Selected Icons

1. Select the cell icon you want to move. Place your mouse pointer within the selected cell icon's rectangular area.
2. Drag the selected cell icon to your desired location then drop it. Please be careful not to place your mouse pointer at any right pin area before you drag the selected cell icon. Because thus will generate a pink line to connect to another cell icon. So place your mouse

pointer at the cell name (caption) area will be an easy way to drag or move the selected cell icon.

Moving a single cell icon will change the relative position for this cell icon to other cell icons. All link traces connected to this cell icon will automatically be rerouted.

If you want to drag or move multiple cell icons at the same time, place your mouse pointer at the cell name (caption) area of any of the selected cell icons. Move multiple cell icons at the same time will keep their relative positions and link traces among the moved/dragged cell icons the same.

### **3.4.7 Cut/Copy Cell Icons and Link Traces**

1. Select the cell icons you want to cut/copy.
2. Click the cut or copy button on the top toolbar.

The cell icon and the property content in the cell's property table (please refer to section **"Property Table Pane"**) will be cut/copied. All trace lines connected from outside to the selected cell icons will be deleted after cut/copy. All trace lines connected among the selected cell icons remain the same and will be stored in the memory buffer/clip board after cut/copy. You can paste them to other positions on the design pane of the same call flow as well as of other call flows.

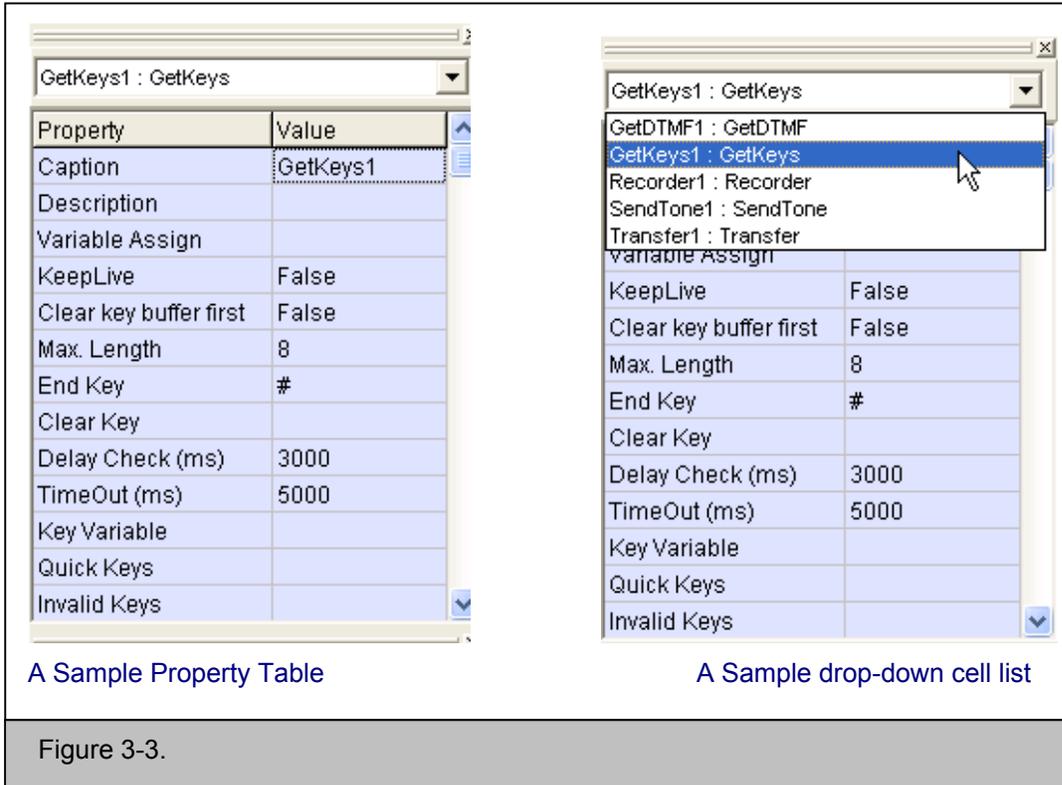
### **3.4.8 Paste Cell Icons and Link Traces**

After cutting or copying the selected cell icons, click the paste button on the top toolbar. The selected cell icons that have been cut or copied will appear at the position 20 pixels lower and right to that of the original cell icons. You can drag/move these cell icons and link traces to your desired position after they are pasted.

The cell icon and the property content in its property table (please refer to section **"Property Table Pane"**) as well as all trace lines connected among the cut/copied cell icons will be pasted. If you paste the cut/copied cell icons to another call flow, they will appear at the same relative position on the design pane as if they were pasted to the same call flow. Copying cell icons save you time since you do not need to fill-in the property and variable table from scratch. Simply Edit or change the captions and/or variables in the property tables and/or variable table to form a similar call flow.

### 3.5 Property Table Pane

The property table pane is next to the design pane and is located on the upper right portion of the Flow Designer main window. Each cell created on the design pane carries its own property table. A cell's property table will appear on the property table pane if you click to select a cell icon on the design pane.



A cell's property table records property values and parameters for that cell. The number of properties/parameters for different kind of cell varies. Different kinds of cells carry various property and parameter items. Different cell of the same kind carries the same property items and the same number of properties with different property values (at least the table name (Caption) will be different). Among property items include **Caption** (table name), **KeepLive**, **TimeOut** value, **Max.** key **Length**, **Quick Keys**, **Invalid Keys**, etc.

All cells share four common properties, namely **Caption**, **Description**, **Variable Assign**, and **Keep Live**.

**Caption** is the label you may assign to the call flow control cell. The default name is the name of the call flow control cell and a number from 1 increasing with the number of the same call flow control you used within a single call flow. For example: "Condition1".

**Description** is similar to Caption in that they both provide a tag for the call flow control cell. However, Description allows you to provide an explanation for the purpose of the cell, providing better documentation of your application so that if someone else would have to use the application you designed, or to alter it to suit a new need that arose, he would have information on why you used that specific call flow control.

In programming terms, the property Description is analogous to comments---lines that do not actually form part of the program but was included to make either remind the programmer of the purpose of either the preceding or following routines or steps.

**Variable Assign** is the property where you may specify a variable that would be used either within the call flow control cell itself or in the succeeding ones. You may also opt to specify an initial value for your variable, which you may call later to determine the steps the call flow may take.

**Keep Live** determines if the call flow would continue even if the telephone is hung up during the call flow's execution. When the telephone is hung up during a call flow execution, the program will check the current cell's **KeepLive** value. If the value is false, then the call flow will end accordingly. The default value for **KeepLive** is false.

If the telephone is in hang up status and the **KeepLive** field of the currently executing cell is selected to be true, then the program won't end until it executes a next cell with a false **KeepLive** value.

The **KeepLive** feature can be useful in relation to database operations to prevent the call flow from ending immediately after the telephone is hung up and thus enable the call flow to execute a count, for example, or other functions like recording the failed call, after the hang up.

You can define and edit a property table on the property table pane by clicking its value fields and then entering values in those fields. Property details and how they influence a call flow will be discussed in later chapters.

If an item in the field is too long to be displayed completely, you can always adjust the column's width by moving the mouse pointer to the edge of the top field and then dragging and dropping the separator line to your desired position when the mouse pointer turns to the following:

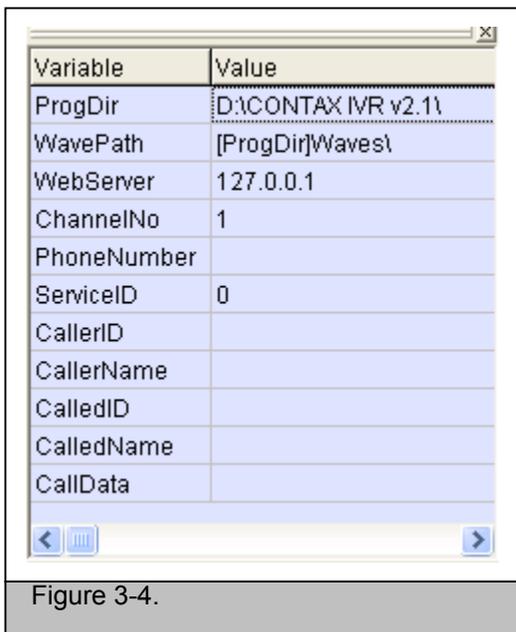


### 3.5.1 Drop-Down Cell List

There is a drop-down cell list at the top of the property table pane (as shown above). If there are many cells in a call flow, you can also switch to a specified cell's property table by clicking its name from the drop-down cell list.

## 3.6 Variable Table Pane

The variable table pane is below the property table pane and is located on the lower right portion of the main window. It displays the variable table for a call flow. Usually each IVR file contains a single call flow, which carries its own variable table. The values in a call flow's variable table are shared among the call flow and its sub-flows. You can define and edit a variable table on the variable table pane by clicking its value fields then entering values in those fields.



Variable	Value
ProgDir	D:\CONTAX IVR v2.11
WavePath	[ProgDir]Waves\
WebServer	127.0.0.1
ChannelNo	1
PhoneNumber	
ServiceID	0
CallerID	
CallerName	
CalledID	
CalledName	
CallData	

Figure 3-4.

Manipulating the variable table, like adding variables, assigning variables, and similar topics will be discussed in later chapters, after you have learned the basics, that of developing simple **CONTAX IVR** applications.

# Unit III-Basic Application Development

Now that you have tried your hand at using the **CONTAX IVR** Flow Designer you are ready to go into developing flow control applications to fit specific needs. In the following chapters you will create **CONTAX IVR** applications, progressing from the simplest to the more complex ones.

## Chapter 4: Your First Application

When you develop applications for the, **CONTAX IVR** using the Flow Control Designer, you are—in effect—developing computer programs.

To develop computer applications, other programmers use “programming languages” like C+, Python, Visual Basic, or Java. Most of these programming languages are encoded, or written, using lines of text and then compiled, or translated into another language the computer can understand and use.

But others, like Visual Basic, provide programmers with graphical user interface (GUI) to allow development of applications using click and drop operations, allowing easier development. **CONTAX IVR** Flow Designer is similar to these later types, offering you GUI-based tools to enable you to develop your applications.

If you already have some programming background, you may skip the discussion below and go to the first sample application. You will still benefit by looking at the examples to find out how each application is made by linking the various cell controls.

### 4.1 Break it Down

For those who might not have any programming background, programming may be an intimidating word. But don't worry. Programming is in fact a natural ability you already have. It is simply the process of breaking down what you want done into a series of steps or actions.

For example, if you want to drink instant coffee, you might do the following steps:

Boil water.

Pour hot water into a cup or mug.

Get instant coffee.

With a teaspoon, measure desired amount of coffee into the water.

Add sugar and stir.

If desired, add creamer or milk.

Enjoy your hot cup of coffee.

In programming, or designing applications for the **CONTAX IVR**, you follow similar a similar route: break down the task into a series of logical steps. When using computer-programming languages, making the actual application means writing the proper code for each step.

In, **CONTAX IVR** creating the application is just a matter of selecting the control icons corresponding to each step and linking them. Of course, you may have to do some tweaking on the default setting of each control icon for the application to do exactly what you want. This matter will be discussed in later chapters.

## **4.2 Taking the Plunge**

As in other skill, like swimming, there is only one-way to learn it: get in the water and swim. In other words, just do it. And that is what we are going to do here. Follow the instructions. Don't worry if you're not yet familiar with the characteristics of the control cells used. Refer to the **CONTAX IVR** User's Guide when you need to find out more details on the properties of each control cells and how they can be changed or tweaked in order to get a certain outcome.

### ***4.1.1 Application No.1: Greeting***

For your first programming task, let us begin with a simple one. Create a "Greeting" call flow, one that automatically answers a call with a greeting.

Open the Flow Designer. You'll notice a blank design pane, indicating you are ready to develop a new application. From the Flow Control cells, click Start.

Move the cursor over to the design pane and click the mouse anew. You should see the Start icon appear in the design pane.

Now, click the Media Control button. The panel displays the various Media Control icons, and from there, click Greeting.

Again, move the cursor over to the design pane, to the right of the Start icon and click. The Greeting icon would appear with the label "Greeting1".

Click the Flow Control button and click End from among the cells. Move the cursor to the design pane to drop the End icon.

Click on the pin of the Start icon and drag your mouse to the right. You will see a purple line trailing the point of the cursor no matter how you move it. Bring the cursor to the Start pin of "Greeting1" and click. A gray line should now connect the two icons.

**Tip:** You may find it difficult to connect the icons if you click on the point of the pin. Try clicking on the body of the pin instead.

Connect the Next pin of "Greeting1" to the End icon.

*Now you should have something like this:*

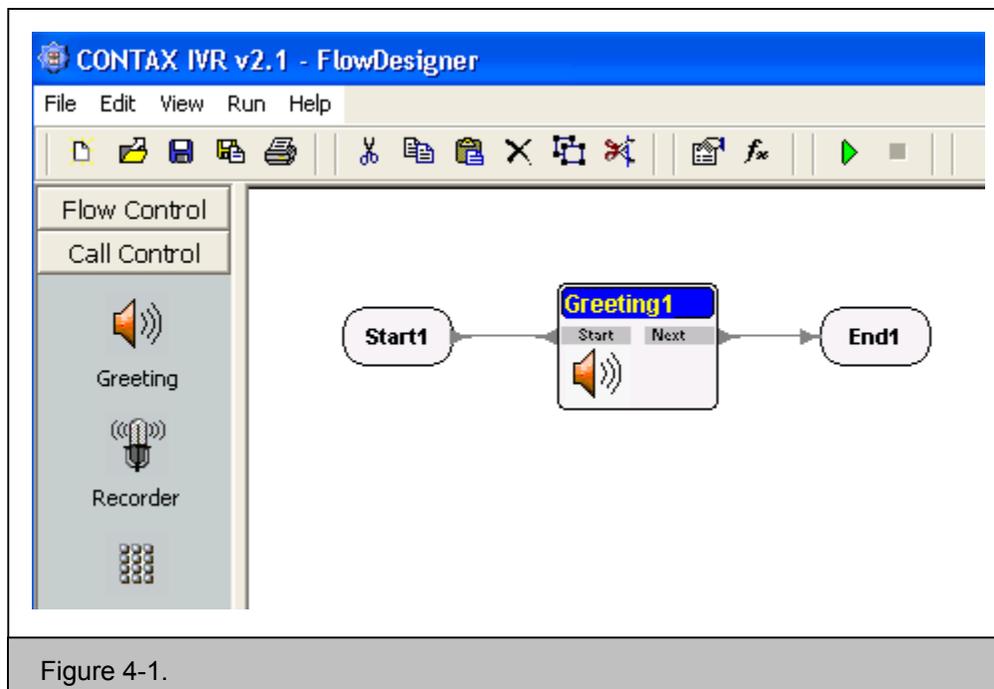
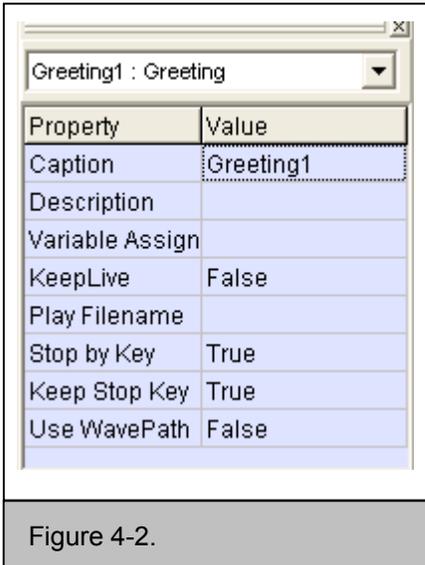


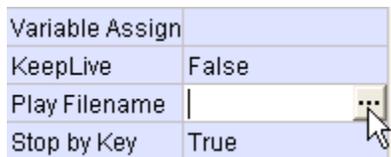
Figure 4-1.

Okay, now we're going to supply the greeting you want played anytime the number is called. But first, take note that the Greeting icon uses wave files (.wav). You may record your own greetings and save it as .wav files using the Recorder software with the **CONTAX IVR** software. We will discuss later in the chapter how to do this.

For now, let's keep things simple by using the .wav files in the example call flows included in the **CONTAX IVR** installation. Now, select the Greeting1 icon by clicking on it. On the Property Pane, the top window in right side of the design pane, properties of the "Greeting1" icon would appear.



What is important now is the "Play Filename" property. Click on the blank Value box on its right side. Click once more and a blank white space would appear with the ellipses (three dots) box on the right.



Click on the ellipses and an Open File dialogue box will open with the default **CONTAX IVR** installation directory. From the folders in the box, open "Samples", then open the "Greeting" folder. Select the "morning.wav" and click Open. You will notice that the "Play Filename" variable would now contain the directory where the "morning.wav" file is located.



You have just created your first call flow!

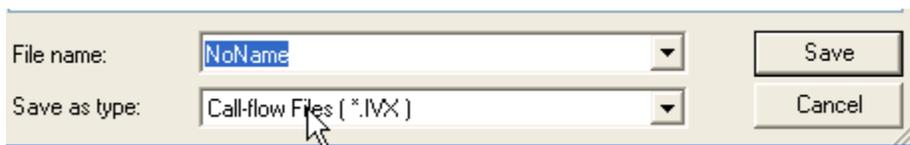
### 4.1.2 Saving the Finished Application

But before running or testing the application you made, you must save it first. Bear in mind, however, that other programmers may be developing .IVX applications in the same computer. To provide details about your finished application, click View/Information on the menu bar of the main window. An Information dialog box appears. Enter the details for your .ivx file and click OK.

You can even keep track of the changes you've made on the call flow you are working on by checking the "Auto-increment version number" box.



To save your call flow, click "File" above the Top Toolbars of the **CONTAX IVR** Flow Designer. From the drop-down list, click "Save As". A "Save As" dialogue box will open. The default filename is "NoName"



But you can change this with any name you want. For our purpose, type "Goodmorning" and click "Save."

Note that the default directory where your application was saved is in the **CONTAX IVR** installation directory. You may create a new directory to save your finished application. Use that directory in selecting the application you want to run or edit.

### 4.1.3 Proof of the Pudding: Simulating Your Call Flow

Once you have saved your application, you must test them for possible errors, or debug them, before deploying them for actual use.

“Bugs” in computer lingo are program defects or errors. To debug means to correct the program flaws.

As you have seen in the first example you ran in Chapter 1, **CONTAX IVR Phone Emulator** is a versatile debugging tool that allows you to follow the processes in the call flow, enabling you to see the active flow control icons is active and hear the messages, if any, being played automatically.

So, let’s run your “**Goodmorning**” call flow. Open the **CONTAX IVR** Flow Designer. Click **File>Open** on the menu bar of the main window and an open file dialogue box will appear. Click the **Goodmorning.ivx** file from the existing files and its name will appear on the File name field. Click the “**Open**” button on the file open dialogue box. The “Goodmorning call flow will appear on the design pane of the main window. Click Run button on the top toolbar, the **Phone Emulator Window** appears.

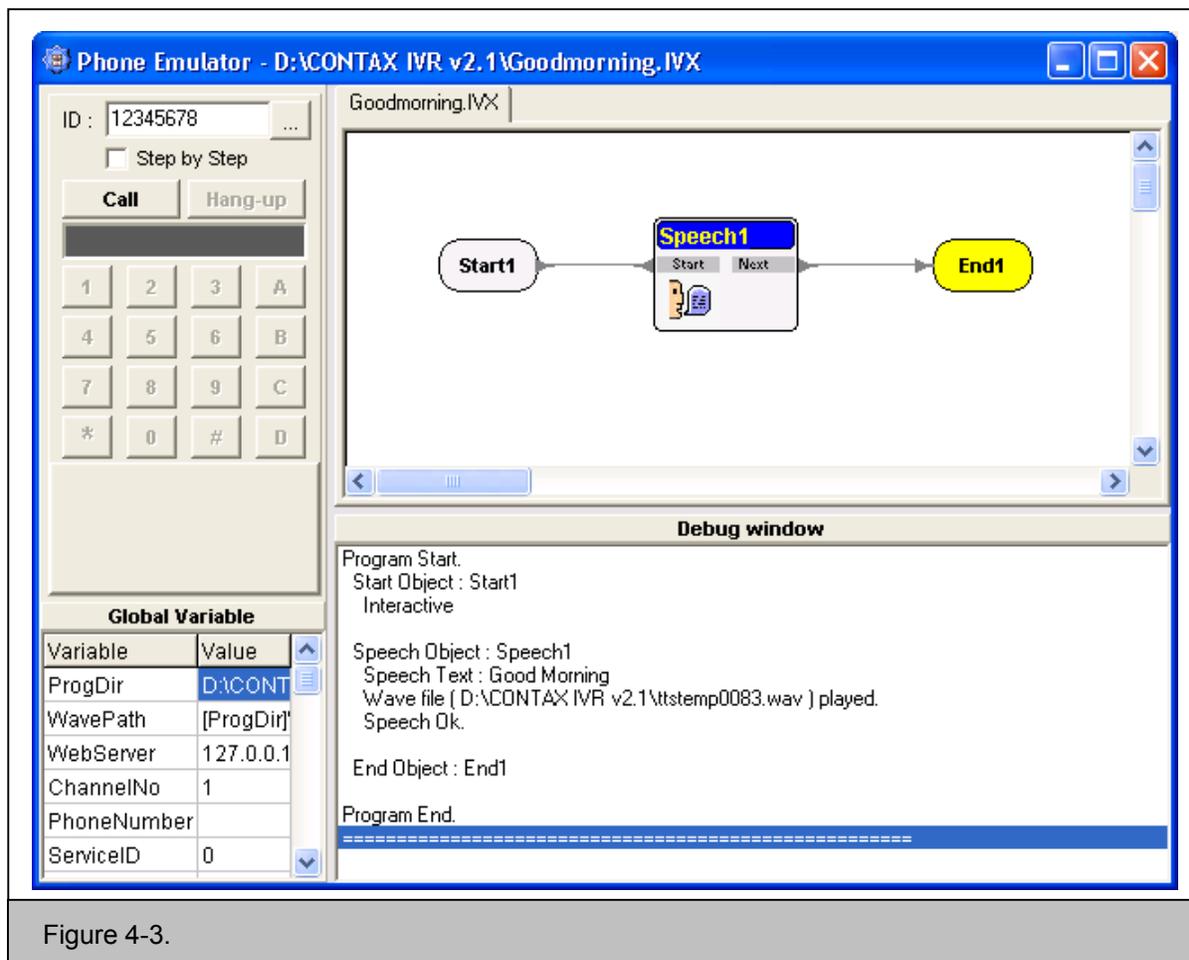


Figure 4-3.

Click the Call button on the Phone Emulator to run the call flow. Notice that the Start icon of the call flow turned yellow, followed by the “Greeting1” icon. You will hear a greeting in Chinese (the recorded message is in Chinese) but it says “Good morning” anyway.

If you want to really get a step-by-step running of the application, check the **Step-by-Step** box. When this box is checked, the simulator stops at each step or break point.

To run the flow against, you need to stop the current call flow. Just click the highlighted End button  then click the Run button again.

Note that the **Phone Emulator** has a **Debug Window** where you can see in code how the program proceeded. The Debug window will tell you if there is any error in your application, like a Syntax error. The Debug window is very useful to application development because it will test the application you designed and you will catch the errors before the program is fielded for actual use.

#### **4.1.4 Recording Your Message**

But how do you record your own .wav files for use in your application?

**CONTAX IVR** has bundled its own Voice Recorder in the software package but you may also use other voice recording software like Microsoft’s Sound Recorder, normally included in the standard installation of Microsoft Windows Operating systems (Win9x, WinME, Win2000, WinXP). Please refer to the included Help file if you don’t know how to record your voice using Sound Recorder. **CONTAX IVR** own voice recorder operates similarly with Microsoft’s own recording software.

You can replace the greeting with your own recorded .wav file by doing the following steps:

1. Open the “Goodmorning” .IVX file. Select the Greeting1 icon. Click twice on the “Play Filename” property.
2. Go to the folder where you recorded your message and select the appropriate .wav file. Click Open. The file directory appears in the “Play Filename” property.
3. Click File/Save on the Top Toolbar to save the changes you’ve made.
4. Run the application anew. The greeting you’ll hear is the message you have recorded.

#### 4.1.5 Using Text-To-Speech

You can also create a similar “Goodmorning” application by using the text-to-speech (TTS) capability of **CONTAX IVR** instead of playing back a .wav file as your greeting.

**CONTAX IVR** supports Microsoft Speech API (SAPI) engine. Microsoft Windows XP already includes Text-to-Speech (**TTS**) function. For other Windows operating systems, you can install the TTS function from CONTAX IVR CD (run **setup.exe** under **SAPI51** folder to install Microsoft Speech SDK5.1 and run **setup.exe** under **SAPI51LangPack** folder to install Microsoft Speech SDK5.1 Language Pack).

Using the “Speech” icon, just write the greeting you want played and the computer will “read”--- playback the text as voice.

In order to create the a similar application using TTS instead of .wav file as greeting, do the following:

Go back to the steps of creating your “Goodmorning” application and do steps 1 and 2.

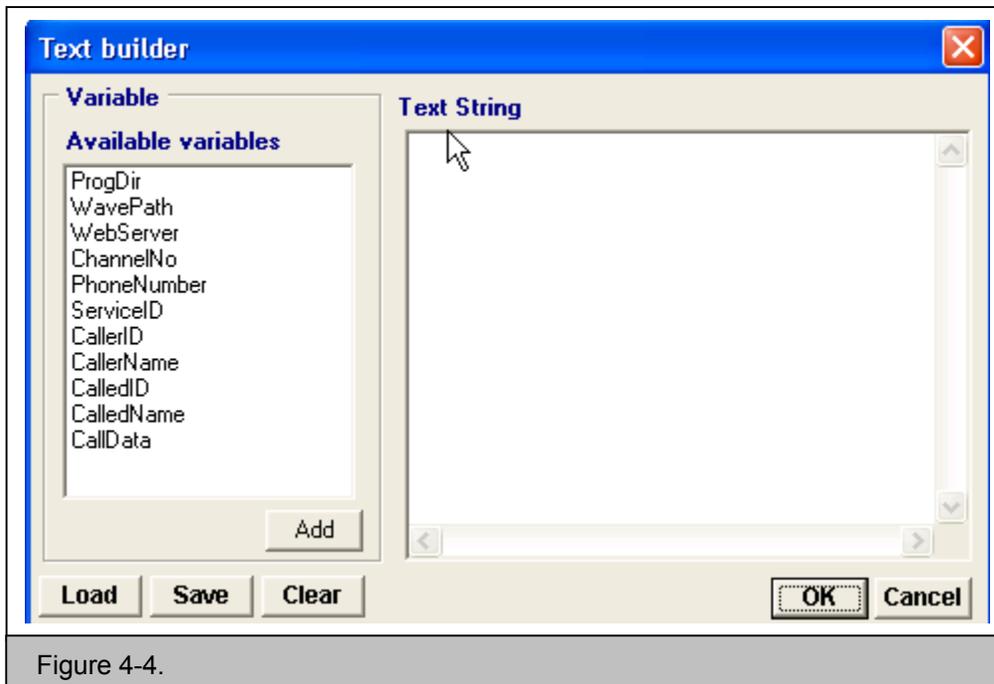
Instead of using cells in Flow Control bar for step three, click Media Control bar and click on “Speech” cell.

Click on the design pane and “Speech1” cell appears.

Do steps 5 to 7.

Select Speech1 cell. On the property pane, click twice on the Value column of the Text property.

Click on the ellipses that appeared and a Text Builder dialogue box opens.



6. Click inside the Text String box and type: "Good Morning". Click OK.
7. Save the application as "Goodmorning2".  
Run "Goodmorning2".

Did you hear the computer voice saying "good morning"?

## Chapter 5: Choice and Consequence

If all your application would do is greet the caller, there is no point in having a computer system do it.

Remember that in Chapter 1, we said that **CONTAX IVR** designed to provide applications for taking over repetitive tasks that humans used to do. In the following discussion, you will go into the process of creating **CONTAX IVR** application for the normal communication need of an office.

In actual situations, things are not so simple or straightforward. So expect that developing **CONTAX IVR** application for regular office communication needs would be more complex compared to the process in making the “Greeting” call flow.

In offices with private branch exchange, the operator would normally ask the caller for either the person or the local number of the department he wants to contact and transfer the call accordingly.

### 5.1 Conditions and Loops

**CONTAX IVR** has the power to automatically handle such choices or **conditions**. Likewise, Contax IVR call flows can be designed to handle situations where the flow must go back to a certain point---or make a **loop**, like when a caller presses the wrong numbers and must be instructed to press the correct ones again. By using **conditions** and **loops**, you can design applications allowing the caller to choose from among alternative actions.

#### *5.1.1 Application No. 2: Automatic Call Transfer*

In order to find out how **CONTAX IVR** can handle a more complex situation with conditions and loops, design an application that automatically transfers incoming calls to the proper local number.

For this purpose, design a call flow to automatically handle incoming calls to XYZ Computer Company and transfers the call to any of its four departments---Sales, Accounting, Management, and Technical---or to a live operator.

You can design your Automatic Transfer Call Flow by breaking the task into a series of logical steps:

1. Receive incoming call.

Greet the caller.

Present caller with the choice of numbers corresponding to the departments he may contact or the option of talking to a live operator.

Find out the number the caller dialed.

If the caller dialed 1, transfer him to the Sales Department.

If the caller dialed 2, transfer him to the Accounting Department.

If the caller dialed 3, transfer him to the Management Department.

If the caller dialed 4, transfer him to the Technical Department.

If the caller dialed 5, transfer him to a live operator.

2. If the caller dialed other numbers, say “Sorry” and instruct him to dial the correct one anew and go back to Step 2.

If the caller pressed the wrong key thrice, go to end of the call flow and hang up.

If the desired number rings and answered, go to the End of the call flow.

If the desired number is busy, say “Sorry” and go to the End of the call flow.

If the desired number rings but does not answer, go to the End of the call flow.

End of call flow.

Programmers normally prepare such a list in the form of a flow-chart, a diagram consisting of differently shaped blocks—each representing a specific action---connected by lines to show sequence of execution.

Even if you have no programming background, it is a good idea to make your own custom flowchart. It would serve as a visual guide to help you create your actual application in the Flow Designer. You can just draw square or rectangular blocks to represent each step and label them accordingly.

Now, you can make the actual application by running the **CONTAX IVR** Flow Designer, selecting the appropriate control icons and linking them.

Click Start cell and put icon in design pane.

Click Media Control button. Click Speech icon and drop it next to Start icon.

Link the two icons.

Select the Speech icon.

From the Variable panel, click vacant value column of Text property. Click on the ellipses and a Text Builder dialogue box will open.

Click inside the Text String box and type the following:

“Welcome to XYZ Computer Corporation. If you know the local number you want to call, please dial it now. Dial 1 for Sales Department. 2 for Accounting. 3 for Management. 4 for Technical. Or dial 5 if you need to talk to the operator.”

Your text box should look something like this.

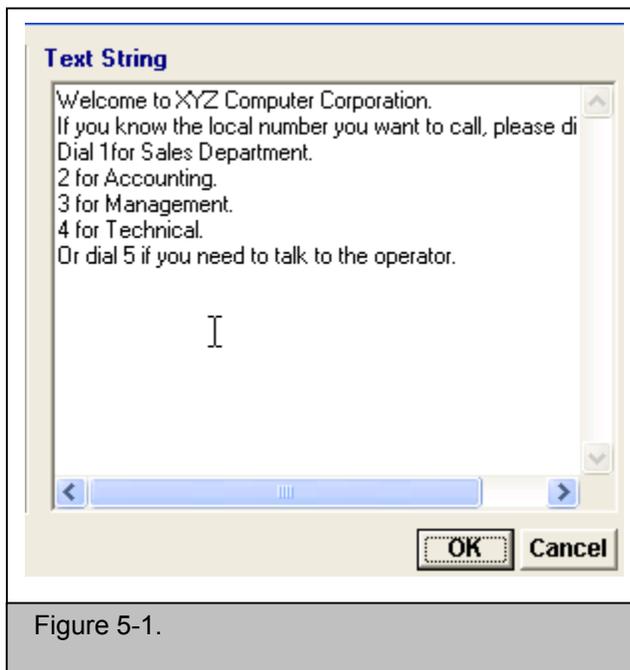


Figure 5-1.

Click **OK**.

Select **GetDTMF** cell from Call Control panel and drop icon next to **Speech1** icon. Link them.

**Note:** GetDTMF waits for a key press from the telephone keypad and exits in the pin corresponding to the button pressed. If you don't link some of the exit pins, pressing their corresponding number in the keypad would be interpreted as an invalid key press and GetDTMF would exit in the Invalid pin.

To further understand the properties of **GetDTMF** cell, please refer to the corresponding section in the Contax IVR User's Guide.

From Call Control panel, click Transfer icon and drop it to the right of the **GetDTMF** icon. Make four more Transfer icons.

Note: **Transfer** cell does what its name suggests: transfers a call to the specified extension number or to an outside phone. Specify the desired local or outside line by clicking on the Value column beside the cell's Target property and typing the number.

Connect number 1 exit pin of **GetDTMF** to first Transfer icon; **GetDTMF** exit pin number 1 to second **Transfer** icon, and so forth, until 1-5 exit pins are connected to a corresponding Transfer icon.

Put an **End** icon next to the right of the first Transfer icon.

Connect the Next pin of the **Transfer** icon to End.

On the right side of the first **Transfer** icon, put a **LinkIn** icon and connect it with the Busy pin.

**Note:** A **LinkIn** icon goes out to a **LinkOut** cell of the same name in the same call flow. You can change the default name, or letter, of the LinkIn icon by clicking the Value column corresponding to the **LinkName** property and typing the desired letter. For our present purpose, use the default letter "A". You may opt not to use a **LinkIn** or **LinkOut** controls but they are useful particularly in avoiding a confusion of the lines connecting the various icons used in the application.

Make another **LinkIn** icon below the first and connect it to the *NoAnswer* pin of the first **Transfer** icon. Make sure the label of the LinkIn icon is "B"; change its Link Name property if needed.

Repeat steps 12 to 14 until each of the three exit pins of the five **Transfer** icons are similarly connected.

Put another Speech icon below the **GetDTMF** icon (default name is **Speech2**); connect the latter's Invalid exit pin to the former's Start pin.

Open the **Text Builder** of the Text property. On the Text String box, type the following: "Sorry, you pressed an invalid key. Try again."

Put a **Counter** cell next to **Speech2**. Connect the *Next* pin of **Speech2** to the *Start* pin of the **Counter1** cell.

Put a **LinkIn** icon next to **Counter1** and label it “**C**”. Connect it to the *Next* exit pin of **Counter1**. Put another **LinkIn** icon next to **Counter1** and label it “**D**”. Connect it to the *Over* pin of **Counter1**.

Between **Speech1** and **GetDTMF** icon, and slightly above them, put a **LinkOut** icon. Label it “**C**”. Connect it to the *Start* pin of **GetDTMF**.

Put two more **Speech** icons below **Speech2**.

Open text builder for **Speech3**. Type the following: “Sorry, the line is busy now.”

Open text builder for **Speech4**. Type the following: “Sorry, nobody is answering the phone.”

Put a **LinkOut** icon, labeled “**A**” before **Speech2** icon. Connect **LinkOut** with **Start** pin of **Speech3**.

Do the same for **Speech4** icon but label the **LinkOut** icon “**B**”.

Beside **Speech3** and **Speech4** icons, put a **LinkIn** icon labeled “**D**” and connect each of them with the *Next* exit pins of the speech icons.

Above the **End** icon, put another **LinkOut** icon labeled “**D**”. Connect the two icons.

Save your finished application as **Autotransfer.ivx**.

Run the **Phone Emulator** to test it. Try out the various options until you are sure the call flow is doing what you designed it to do.

Now, you have a better idea of how an actual **CONTAX IVR** application is designed and made.

### **5.1.2 Application No. 3: Doing The Same Thing Differently**

There are many ways to skin a cat, and so it is with application development. In the previous example, you used the control cell GetDTMF to transfer the call to a local number. That is one of the simplest way to let handle caller choice, particularly when used in connection with the Transfer cell, and route calls automatically to the desired numbers.

But you can also use other controls to handle caller choice. Used in tandem, the GetKeys and Condition cells can handle from simple to more complex choice of action to the caller. Since you would be using the GetKeys cell with the Condition cell as a sort of traffic policeman to direct where the call flow would have to go, take a look at the comparison of some of the characteristics of GetKeys with GetDTMF control cells. These factors would help you decide later what control cell to use when designing applications with provisions for handling caller inputs.

### **5.1.3 GetKeys versus GetDTMF**

So, what's the basic difference between a GetDTMF cell and a GetKeys cell? Both cells processes caller input from the phone's touch-tone keypad which the call flow can use to decide what actions to execute.

#### **Number of Inputs**

GetDTMF could handle only a single input. Even if the caller presses several key, the GetDTMF cell will remember only the first key pressed and ignore the others.

On the other hand, the GetKeys cell could handle a certain number of touch-tone inputs—or key presses. You can specify the number of inputs GetKeys by altering the values of the Max. Length property. The default value is 8.

#### **Error-Handling**

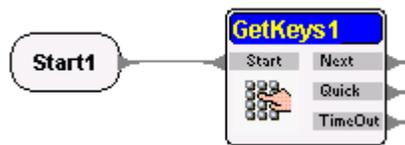
Both control cells can determine invalid key presses. As you may have observed in Application No.2, GetDTMF interprets as invalid input any key pressed, when its corresponding exit pin is not connected to any other control cell. When an invalid key is pressed, GetDTMF exits from its Invalid pin.

**GetKeys** has its own built-in property for handling invalid key presses. However, you must specify the invalid keys for the control cell to treat it as such. To do this, click the value column of

the **Invalid Keys** property and type in the invalid numbers. For example, the voice menu uses key 1 through key 5. You can define 6,7,8 as **Invalid Keys** (use comma between two keys).

When GetKeys detects an invalid key press, it loops back to the Start pin and waits for another input. If you specify a .wav file to play in the value column of the **Invalid Voice** property, GetKeys plays the message. GetDTMF doesn't have this built-in capability for message playback when an invalid key press is detected.

To understand better the error-handling capability of GetKeys, run Flow Designer and make the following call flow.



Define keys 6, 7 and 8 as Invalid keys.

Property	Value
Invalid Keys	6,7,8

Record the following message: "Sorry, you pressed the wrong key. Please try again." Save it as Wrong.wav in the **ContaxIVR\Wave** folder. Select the same file in the same location after you click the value column of the Invalid Voice property.

Save the call flow as Experiment1. Run it and press any of the invalid keys defined. You should hear a playback of the Wrong.wav file.

Now, you are ready to design a variant of the Automatic Transfer call flow. To make things interesting, add a voice-recording feature in the call flow. Here are the requirements:

Design an automatic transfer call flow for the XYZ Computer Corporation.

Caller option consists of call transfer to the four departments specified in Application No 2, or to a live operator.

Instead of 1 digit input, use the following keys to effect call transfer:

- 101 for Sales Department
- 102 for Accounting

103 for Management

104 for Technical

105 for Operator

Use voice messages (.wav files) for instruction.

Give error message for pressing invalid keys and prompt caller to retry.

For busy or unanswered calls, instruct caller to record his name and phone number for call back purpose.

To begin record first the .wav files you are going to use as messages or prompt. Open the recorder of your choice and record the following messages:

Greeting1: "Welcome to XYZ Computer Corporation. If you know the extension number you want to reach, please dial it now. Dial 101 for Sales Department; 102 for Accounting; 103 for Management; 104 for Technical; or dial 105 if you want to talk to the Operator." Save the file as Transfer.wav in the Contax\IVR\Wave folder.

Error: "Sorry, you pressed the wrong keys. Please try again". Save file as Wrong.wav in the Wave folder.

Busy: "Sorry, the line is busy now." Save file as Busy.wav in the Wave folder.

No Answer: "Sorry, no one is answering the phone." Save file as NoAnswer.wav in the Wave folder.

Prompt: "Please state your name and telephone number so we can call you later. Press any key when you are finished. Thank you." Save the file as RecName.wav.

Let us walk through the process of creating this application. But instead of the detailed instruction in the previous examples, let's use portions of the actual call flow as illustrations. Important features of the call flow will be discussed when these are crucial to the execution of the call flow.

Now begin creating your application:

1. Link the following control cells:



Use Transfer.wav as the value of the Play Filename property of Greeting1.

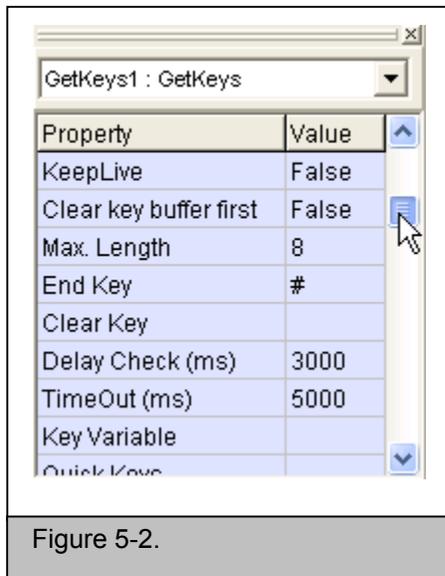
Next, you would have to specify a variable where you would keep the value of the keys pressed. Think of a variable as a box in the computer’s memory that holds a data value. You must supply a label, or name, for each variable to enable the computer to know which “box” to look in when you need the data contained in a certain variable. You will find out more about variables in later chapter. For now confine yourself to the variable where you would put the value of the keys the caller pressed.

To do this, select the GetKeys1 icon in the design pane. Notice that the Property and Variable table for the GetKeys1 is now displayed in the right side of the design pane. The nine default variables displayed are common to all control cells, but none of these would serve your purpose. You can add a new variable by clicking on the value column of the last variable displayed, in this case “Call Data”, and then pressing the down arrow key in your keyboard.

A pair of new blank cells is created. Click in the Variable column and type the name of your variable. For the present purpose, name the variable as “Key1”. Keep the Value column blank.

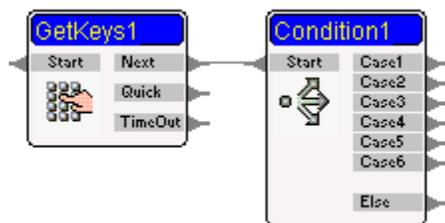
CallData	
Key1	

Next, go to the Property table and click on the value column of the Key Variable property. You may have to click and drag the slider down on the side of the Property table to see the Key Variable property.



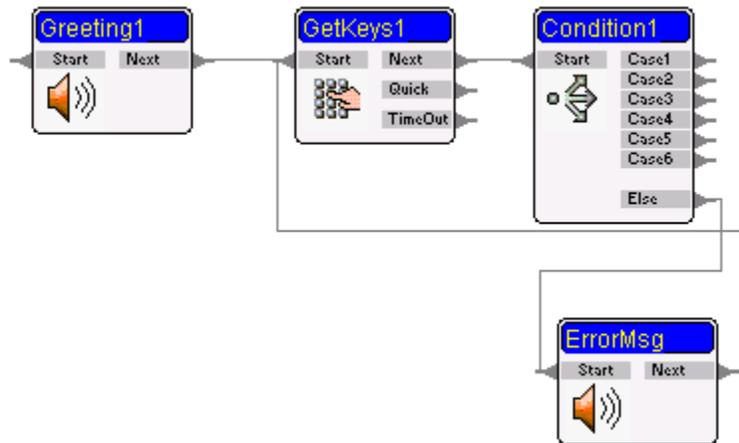
Type the name of the variable you earlier created, enclosed in brackets “[Key1]” and hit the Enter key. The brackets direct the computer to look for the value of the variable Key1.

2. Next connect a Condition cell to the GetKeys cell.



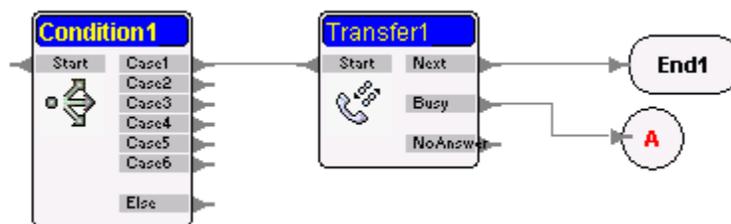
Go into the Property table. If necessary, drag the slider down, to see the Case1-Case6 properties. On the Value column of the Case1 property, click and type the following: “[Key1]=101. This tells the computer that when the caller presses the keys 101, Condition1 cell would exit on its Case1 pin.

Remember that in our requirement, the company has five local numbers: 101, 102, 103, 104, and 105? So for Case2, make the value of Keys1 equal to 102, 103 for Case3 and so on until Case5. What if the caller pressed the wrong number? If he does, Condition1 cell would exit by its Else pin. So here you can connect another greeting cell to give an error message and a prompt to try anew. Label the Greeting cell as “ErrMsg” and the Wrong.wav file in the Wave folder.



Next, you have to provide for the possibility that the caller would press the correct local numbers. If he presses 101, the computer would put this value in the variable Keys1 and compare it with the Key Variable you have specified. Since you said Case1 would be true if the value of Keys1, or [Keys1], equals 101, the cell would exit by its Case1 pin.

Similarly, if the caller presses 102, the computer would then exit Case2 pin; if 103 on the Case3 pin; and so on. You can affect the proper call transfer, by using the Transfer cell. Link Transfer cells with Case1 until the Case5 pins.



**Note:** The most important property of the Transfer cell is Target.

Property	Value
Target	101

You can put in its value column the actual number of the phone where the call would be transferred or any variable to hold the intended phone number. If you fail to put any specific phone number or variable, the call won't be connected to the intended local or outside line.

The Transfer cell detects if the desired number called is answered, busy, or if nobody is answering the phone. If the call is answered, the requirement of the program---to automatically

transfer call---is met and so you can terminate the call flow by linking an End cell with Transfer cell's Next pin.

If the number is busy, the Transfer cell exits through its Busy pin. Under the program requirement, you must then generate a message informing the caller that the line is busy. You can do this by linking another Message cell with the Busy pin of the Transfer cell. Label the Message cell as "Busy" and select the Busy.wav file for playback.

In the same manner, create another Message cell, labeled as "NoAnswer", and connect it with the similarly-named pin of the Transfer cell. This time, use the NoAnswer.wav as the response.

### **LinkIn, LinkOut**

As an alternative, it might be a good idea to use the Linkout and Linkin cells again to prevent the confusion of criss-crossing connection lines. Here's one way of doing it:

1. Cut the connections between the Transfer cells, on the one hand, and the Busy and NoAnswer message cells on the other. Select the two message cells and drag them in a vacant space in the design pane.

Create a new LinkIn cell, and label it "A". Connect the Busy pins of the five Transfer cells to "A". Make another LinkIn cell and label it "B"; connect all the NoAnswer pins of the Transfer cells to "B".

Make a LinkOut cell and label it "A". Connect it with Start pin of Busy cell.

Make another LinkOut cell and label it "B". Connect it with Start pin of NoAnswer cell.

When the Transfer cell detects a busy signal, it would exit through its busy pin. When it finds the LinkIn cell connected, it would look for a LinkOut cell with the same name and continue its execution. Finding the LinkOut cells connected to the Busy and NoAnswer messages; the computer would then play the respective .wav files specified in each cell. This takes care of the required prompts for busy and no answer calls.

## Recording Messages

However, you must remember that you also want **CONTAX IVR** to leave a recording of his name and phone number for possible call back. And before the recording, you want to give instructions to the caller.

You can address this problem by doing the following.

Connecting another message cell to the Next pins of both the Busy and NoAnswer cells. Label the new message cell as RecOpt and use the RecName.wav for playing back the message.

Creating a Recorder cell and connecting its Start pin with the Next pin of RecOpt message cell.

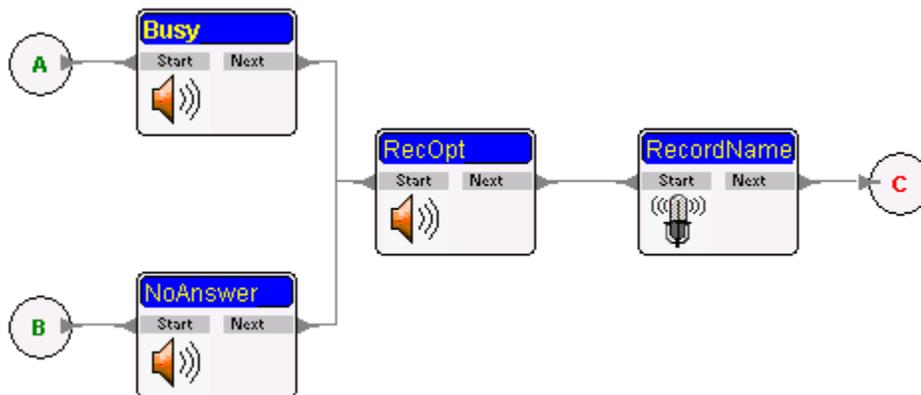
**Note:** A Recorder cell, as its name says, records a .wav file and stores it in the specified folder. The default folder is **C:\Program Files\Eletech\Contax IVR\Waves\**. Of course, you can also specify your own directory where all recorded messages will be stored. Please refer to the User Guide on how to do this.

You can end the recording anytime by pressing any key if you select the value of the **Stop by Key** property as True. If False, the message will be recorded till its end: the value of MsgLength property in seconds. For the present purpose, keep the value of Stop by Key as True.

After recording the message, the program should end. To do this, create a new LinkIn cell and label this "C." Connect the Next pin of Recorder cell with "C."

Create a LinkOut cell, labeled as "C" near the End control cell. Connect the two cells.

Your control cells may look like this:



You have just finished your new application, a new version of the your first automatic call transfer program, but more complex, and with an additional function of recording messages from the caller. Save your application and run the Phone Emulator for debugging until you are satisfied that the call flow is doing what it was designed to do.

Remember the characteristics of the control icons you've used because you would meet them again in the next chapter, when we begin making advanced call flows.

# Unit IV – Advanced Call Flow Design

Now that you have gotten down some of the basic skills in developing common applications for, **CONTAX IVR** it's time to get to more interesting things. In the following discussions you will learn the following:

- Use sub-flows
- Define and manage variables

## Chapter 6: Subflow: Keeping Complex Things Simple

Creating a call flow that would automatically take care of most communication needs of a good-sized company may be too daunting a task because of its complexity, not to mention the tangle of lines connecting the various control cells you would use. But there is a way to manage things and keep it simple: use a sub-flow.

### 6.1 What is a Subflow?

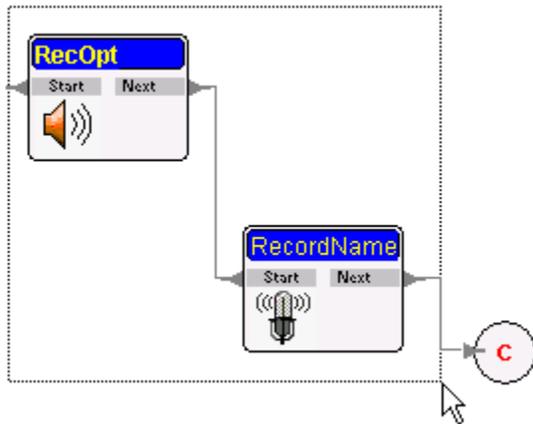
A Subflow is similar to a subroutine in computer programming languages, a procedure separate from the main program but which the main program can call to do something. After it has done its job, the subroutine passes back the ball, so to speak, to the main program.

A Subflow control actually represents a separate call flow **CONTAX IVR** calls. After the subflow finishes execution, the main call flow continues its own.

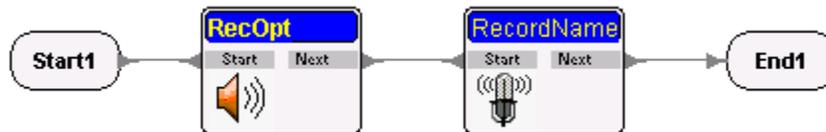
This is by far the most important characteristic of the sub-flow control. This means you can develop a complex application by making a few simple ones and stringing them together to work as a single application.

#### **6.1.1 Application No. 4-Automatic Transfer with Voice Mail**

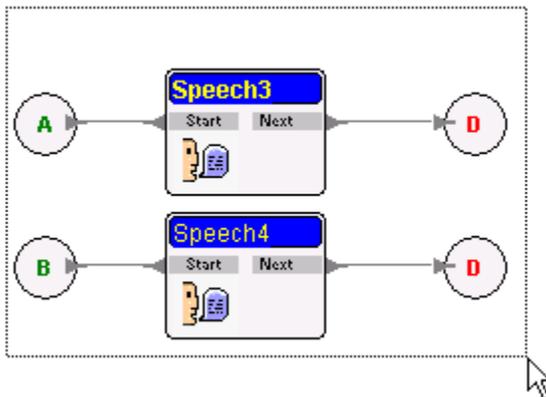
To illustrate how a Subflow can simplify creation of an application, we are going to combine the AutoTransfer application with the Record call flow. You can actually use the recording portion (see figure below) of the second one to extend the functionality of the AutoTransfer call flow by using a Subflow.



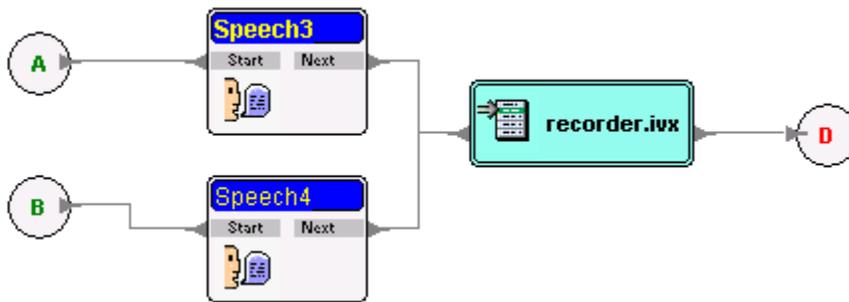
You can define the above portion of the Record call flow as a subflow simply by copying and pasting it to make a new application like the following:



Save the new application under a new filename. For our example, you may save the above call flow as Recorder.ixv. Then go to the following portion of the Autotransfer before the LinkIn:



*In effect you would have something like this:*



Save the application as **Sub\_flow.ivx**. See how simple it is to make a new one by using a Subflow?

### 6.1.2 Saving and Naming Subflows

In defining subflows, it is crucial that you indicate the correct file name (\*.ivx) and directory of where you saved the new call flow.

Supply only the filename if the **Use ProgDir** value is True. The default directory is:

ProgDir = C:\Program Files\Eletech\ContaX IVR\

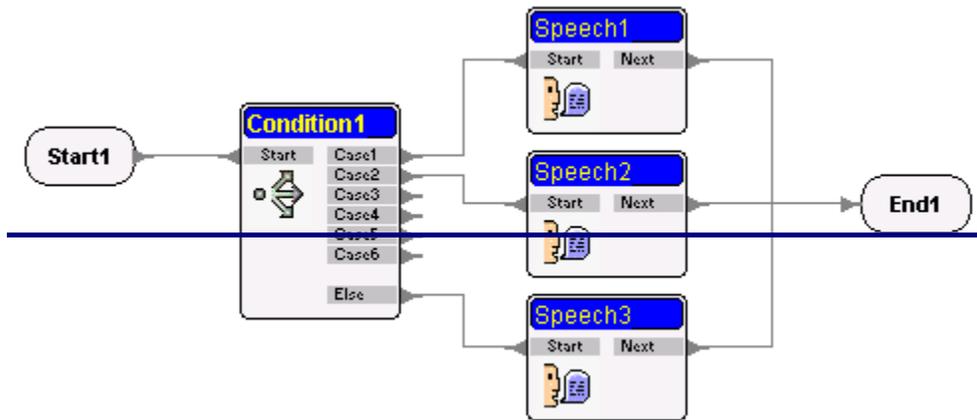
This is where the new call flow would be saved and where **CONTAX IVR** would look once it finds a **Subflow** control in the application. If you save your call flow in another folder or directory, you would have to use False as the value of the Use ProgDir and indicate in the ProgDir value the actual folder or directory where you saved the call flow. For example:

ProgDir = D:\Contax IVR\Subflows

Application No. 5: Time Check

Here's a short application that you may use as a sub-routine immediately after your Start cell and before another Message cell that gives the typical message: Welcome to (Name of Company). This application simply determines the time of the day and gives the appropriate greeting: "Good Morning", "Good Afternoon", or "Good Evening."

By using this short application, you can be sure to give the correct greeting to the caller anytime he calls. Open the application **Timecheck.ivx** and you will see that it consists merely of the following control cells:



The application uses variables, the function NOW and an expression to get the correct time. The values returned from the expression are compared to the following values: 1200 and 1800.

If the value is less than 1200, the appropriate greeting played is “Good Morning”; if greater than 1200 but less than 1800 (equal to 6:00 p.m. in military time), “Good Afternoon; and if greater than 1800, “Good Evening.”

You will learn more about variables, functions and expressions in the next chapter. After reading the next chapter, it would be a good idea to open the [Timecheck.ivx](#) application anew and run it to find out how it works.

### 6.1.3 Building a Library of Subflows

When you design your **CONTAX IVR** applications as a collection of subflows, you won't have to design future applications from scratch. Since there is similarity in the communication needs of most businesses, or departments of a certain company, you need only to use previous subflows you have designed---probably with minor modifications---to develop required applications. You would likewise save time developing the applications.

For example, your library of Subflows could consist of the following:

- Automatic transfer subflow
- Send fax subflow
- Receive fax subflow
- Send e-mail subflow
- Greeting subflow
- Record message subflow
- Send SMS subflow

As you go along developing applications, your collection of subflows would grow and enable you would not only develop applications faster, but more confidently. Because you have already tested each of the subflows and have debugged them---made them error-free---there are lesser areas in your application where bugs could occur.

## Chapter 7: Variables and Expressions

Variables, as you learned from the previous chapter, is like a hole in the computer's memory, or a virtual box, that can contain data. Variables hold two kinds of data: numeric (numbers) or strings (letters, combination of letters and numbers).

Variables can help you simplify the design of your **CONTAX IVR** application by holding data you may want to compare or evaluate to determine where the program would branch into.

### 7.1 Variable Table Pane

You will see variables of each call flow control in the variable table pane, which is below the property table pane and is located on the lower right portion of the main window. Usually each IVR file contains a single call flow, which carries its own variable table. The values in a call flow's variable table are shared among the call flow and its sub-flows.

You can define and edit a variable table on the variable table pane by clicking its value fields then entering values in those fields.

#### **7.1.1 Adding Variables to The Variable Table**

The following are the default variables for any call flow: ProgDir, WavePath, WebServer, ChannelNo, CallerID, CallerName, CalledID, CalledName, CallData and ServiceID.

**ProgramDir**, as it's name suggests is simply the directory where your **CONTAX IVR** application is located, for example: C:\Program Files\ContaxIVR 2.1.**WavePath** is also a directory, but one where the .wav files are saved. The default path is [ProgDir]\Waves.**WebServer** is either the computer where the web pages you want to access is stored. The default value is 120.0.0.1 but you can also use a URL (Universal Resource Locator), like <http://www.eletech.com>

**ChannelNo.** specifies the channel being used by the running **CONTAX IVR** application; the default value is 1. The values of the remaining variables, **CallerID**, **CallerName**, **CalledID**, **CalledName**, **CallData** and **ServiceID** are left vacant as default but putting values in these variables could be useful particularly in monitoring the calls.

Apart from the default variables, you can define a new variable in the property table, by adding this to the variable table in order to be shared throughout the call flow.

As you have done in the previous chapters, you add variables by moving the mouse pointer to the last row of the variable table and clicking on it. The cursor will appear on the last row (either on the variable field or on the value field). Press the down arrow key to create a new row of variable and value fields. Press the up arrow key or Delete key to delete a created row.

After creating a variable in your variable table, you can set an initial value to this variable. Click to select the value field for the variable you want to set an initial value to. Click the field again to edit the field. You will notice that an empty field with a blinking cursor at the first character of the field appears. Enter either a number or a string in this field as an initial value for this variable.

You can set another variable or string combination of other variables as the initial value here by using a pair of square parentheses “[ ]” to enclose a variable name in the value fields (e.g. set the initial value for variable A to be Dial [B] for [C], when B=123 and C=Mary, then A=Dial 123 for Mary. A will change its content when B or C changes its content, e.g. when B=1-800-7661234, C=help, then A=Dial 1-800-7661234 for help). Later you can use **Speech** (text to speech) function to read variable A.

### **7.1.2 Assigning a Variable**

Variables can be assigned in the value fields (at the right portion) of a cell's property table. When you do so, you must use a pair of square parentheses “[ ]” to enclose a variable name in the value fields, for example, [EmployeeID], [UserPIN], [ZIPCode], etc. By doing so, the program will not mistakenly treat the variable as a text string.

Whenever a cell is executed in a call flow, its variable assign field will be executed once too.

### **7.1.3 Using the Expression Builder**

You can use the **Expression builder** dialog box to create an expression in the value field of a property table:

Click to select the value field of the variable assign property in a property table. Click the value field again to edit the field. You will notice that an empty field with a blinking cursor at the first character of the field appears. Also a button with ellipses (three periods) mark appears at the

right edge of the empty value field. You can direct enter an expression in this field. You can also double click the empty field or just click the ellipses button, and then the expression builder dialog box appears.

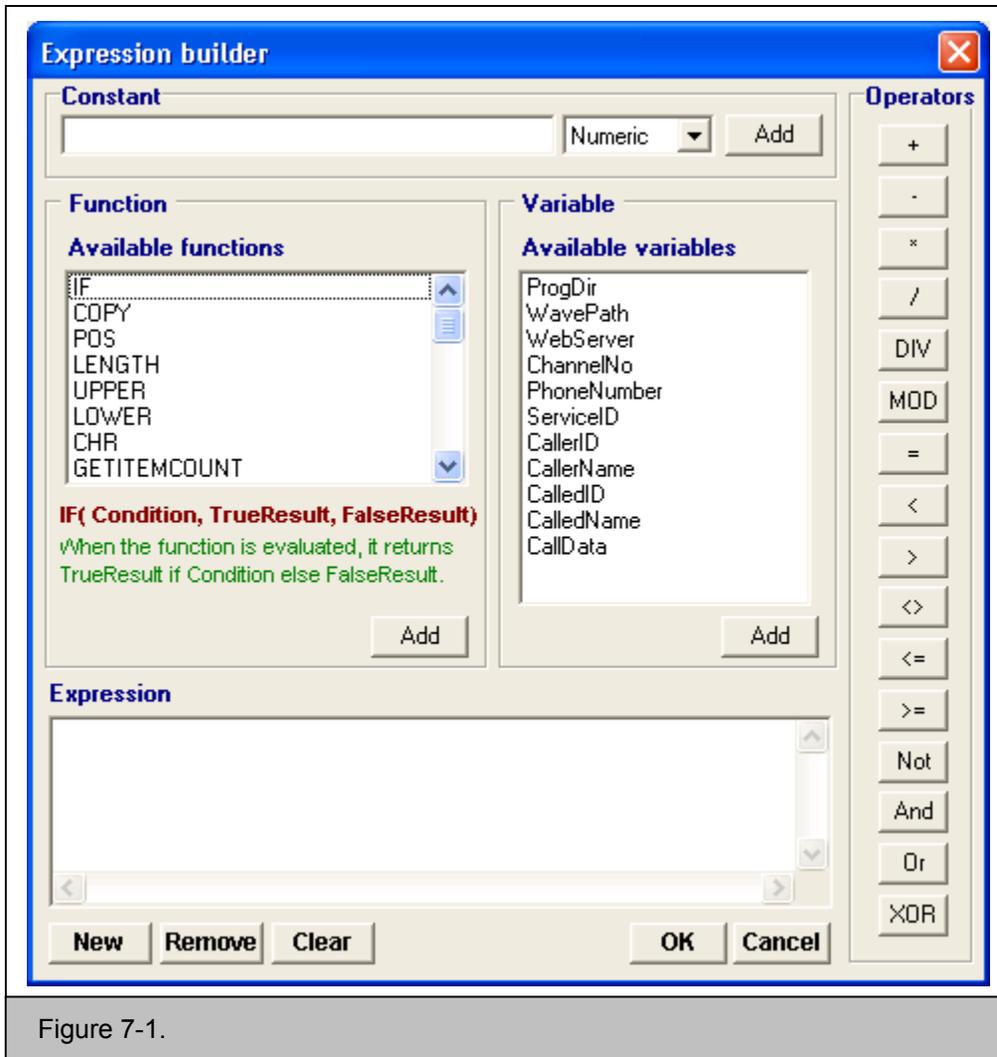


Figure 7-1.

### Enter a Constant

Click the constant field in the constant section of the **Expression builder** dialog box to enter a constant.

If you enter a number as a string here, you need to select this constant to be string instead of numeric type. Click the down arrow in the constant section. Select and click String from the drop down list. Click the **Add** button to add this constant to the expression pane.

Instead of using the **Expression builder**, if you direct enter a number as a string in the value field of the property table, please remember to use a pair of apostrophes (e.g. ['\$123']) to enclose the number.

If you use the expression builder to enter a number string, a pair of apostrophes will automatically be added.

### **Enter a Function**

Click to select a function from the function list in the function section of the **Expression builder** dialog box. The description of that function will also appear at the lower portion of the function section. Click the **Add** button to add the function onto the expression pane. The function containing a number of vertical bars (|) separated by commas (,) appears in the expression pane. You can replace any | with an expression by direct entering it or by using the **Expression builder** again to enter it.

### **Enter a Variable**

Click to select a variable from the available variable list in the variable section of the expression builder dialog box. Click the **Add** button to add the variable to the expression pane.

### **Enter an Operator**

Click an operator button in the operators section of the **Expression builder** dialog box to add the operator onto the expression pane.

### **Enter Expressions in a Property Table**

If you want to enter more expressions after one expression is entered in the expression pane, then press the **Enter** key on your keyboard to change to the next line of the expression pane to enter the next expression. Repeat the above method until all your desired expressions are entered.

After all expressions have been entered in the expression pane, then click the **OK** button at the bottom of the expression builder dialog box. You will notice that all expressions are entered onto the value field of a cell's property table and that a semicolon (;) separates them between two adjacent expressions.

### **7.1.4 Local and Global Variables**

Variables can either be made local or global. If you have an application consisting of a main call flow and several call flows, you may wish to assign a variable that would be used entirely within a single sub-flow. Such variable, is called a **local variable**.

But, you may also want to pass the content of the variable to the main call flow for processing. Variables that are shared by several call flows are called **global variables**.

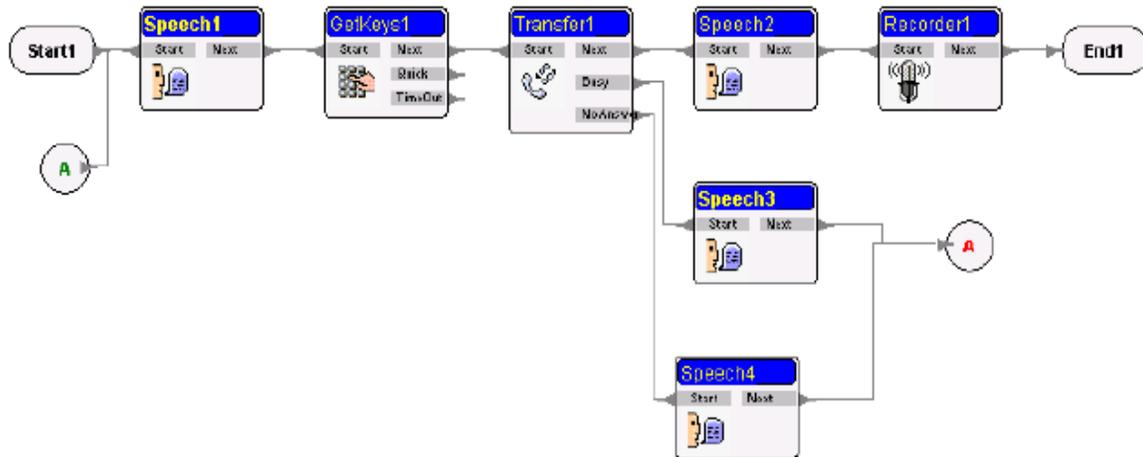
For example, you may want to assign the variable **Email\_doc** in a subflow giving the caller choice of available documents he want to be sent to him through e-mail. Then, when the subflow exits, you can pass the Email\_doc variable by assigning it to other call flow control cells in the main call flow to enable the application to send the appropriate document.

### **7.1.5 Application No. 5-Another Variant of AutoTransfer**

For this application, you will find how using variable could make life simpler for you. Again we would go back to the basic application you have made, the AutoTransfer application. You may have found it quite complicated for so simple a task. And true enough it is. There is a more, simple and elegant design to do the same thing. By using variables to store the key pressed---the local number chosen by the caller---and passing the value of such variable to the Transfer call flow's Target property, you eliminated the need to use a Transfer cell for each local number. All you need is one!

Let's look at a simple application that is actually a variation of the previous applications you have been doing. Open the **CONTAX IVR** Flow Designer. Click File>Open. The Open File dialogue would open at the default installation directory of ContaxIVR2.1. Double click the folder Samples and when a new set of folders open, double click on the TransRec folder. Select to open the TransRec.IVX file inside the folder.

The whole TransRec call flow consists of the following:



Yet, this very few collection of call flow controls does the same job as the AutoTransfer application with Recording capability. This was made possible by the use of variables!

Let's find out that variable which made all the difference.

First stop, click on the GetKeys1 control and look at the Property and Variable Table panes. Notice that on the Variable table pane, a variable **ext\_no** has been added.

CallerID	
ext_no	
rec_file	

In the same manner, on the Property table, you will find that the corresponding value of the Key Variable property is the value of the variable **ext\_no**, specified by the following entry: **[ext\_no]**. This means that the keys pressed would be stored in the ext\_no variable.

Now, select the Transfer cell. Again, look at the Property table. Notice that in the value column of the **Target** property, [ext\_no] is indicated, meaning the desired local number is made equal to the values of the keys the caller earlier pressed, stored in the variable ext\_no. So when the Transfer cell executes its operation, it would look into the value of the Target property to find out where the line should be transferred. But instead, it would be directed to look into the ext\_no variable and use the value inside this virtual box as the local number, which the caller wants to contact.

Run the application in the Phone Emulator. Look closely into the Debug window. You may have to resize the divider between the main window and the Debug window to see better the information you want to find out.

When you press the Start button, and the greeting is played, input 123 as the local number. When you do so, you will find the following in the Debug window:



```
Debug window
GetKeys Object : GetKeys1
User Input = "123"
Set ext_no = "123"
```

The first line means that it is the GetKeys1 control cell that is active at that time. User Input is the keys you pressed. The third line sets the value of the variable ext\_no to be “123” or equivalent to the keys you pressed.

When you press the OK button when the Transfer cell is active, you would receive a prompt that your message is being recorded. Pressing another key would end the recording, similar to the Record application you have made earlier.

### Using Expressions

You might have noticed the rec\_file variable. This is simply the variable that assigns a filename to the .wav file that the Recorder cell would have as an output after recording the message of the caller. Notice that in the Assign Variable portion of the Property table, the value is in the lengthy expression:

```
[rec_file]= [ext_no] + COPY( DATE, 6, 2 ) + COPY( DATE, 9, 2 ) + COPY( NOW, 12, 2 ) + COPY( NOW, 15, 2 )
```

An expression is a statement that the computer evaluates to get a certain value. It may contain a variable, a constant and a function. You already know what a variable is. A constant is simply any fixed value. For example if you set the expression:

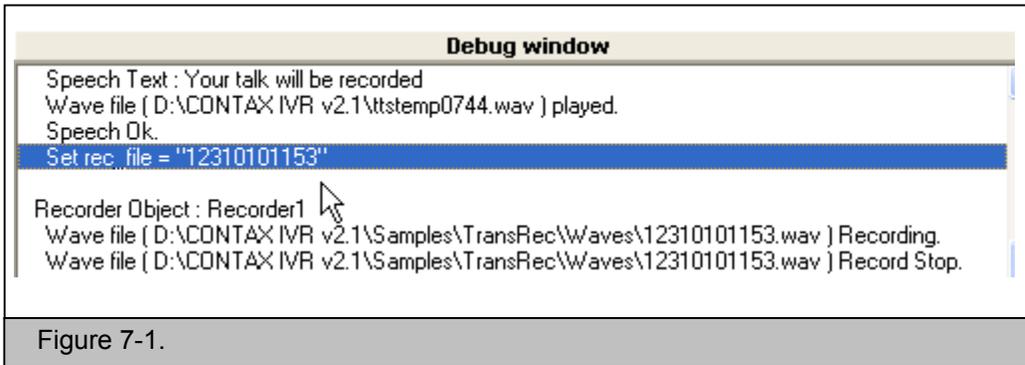
```
Emp_No=1234, Emp_Name=Lina
```

In the first example, 1234 is a numeric constant. In the second, the value “Lina” is a string constant.

**COPY** and **DATE** are among the functions **CONTAX IVR** has built into its system to increase the flexibility of call flows and enable designers to tailor-made applications for specific needs. Please refer to the User Guide for discussion of the specific functions.

For now, let’s leave the use of functions for more experienced application developers. Still, we will discuss the expressions used here just to give you a taste of how expressions and functions work.

If you looked closely at the Debug window when you are running the application, you would have noticed that after the TTS message “Your talk will be recorded” played, the program puts the following line: Set **rec\_file**=“12310101153”. In other words, it puts the value 12310101153 into the variable **rec\_file**.



And then, later the Recorder records the file “12310101153.wav” as its output.

If we break down the first part of the .wav filename in the following manner, 123-1010-1153, things begin to get clear. It is now obvious that the first three digits of the filename is the extension number the caller pressed. Remember that the first part of the expression says:

[rec-file]=[ext-no]

The second set of four-digit number is the date, meaning October 10. You see, when the function Date is invoked, it returns data in the following format: **yyyy/mm/dd 00:00:00**. So you have a string with 16 characters all in all. The **COPY** function returns a portion (sub-string) of a string containing a number of characters (Count) starting at Index (I) or in the format: Copy(Substring, Index, Count).

Thus the expression **COPY( DATE, 6, 2 )**, would return a substring of the function DATE starting from the sixth character, consisting of only two characters. If you count the format of the DATE function, you will find out that what the expression **COPY( DATE, 6, 2 )** merely returns the month portion of the DATE function.

The second similar expression **COPY( DATE, 9, 2 )**, returns two characters representing the exact day of the month.

So, assuming that today is October 10, 2002 all the expression **COPY( DATE, 6, 2 )+COPY( DATE, 9, 2 )** would return is the value 1010.

The last set of four-digit number is the time the call was recorded. How is it so? The function NOW returns a data of the same format with the DATE function. However, although the DATE function has a provision for time in the last eight characters of the format, or 00:00:00, the figures remain zero. On the other hand, the NOW function returns not only the actual date but also the correct time.

Thus the expression **COPY( NOW, 12, 2 )** returns two characters from the NOW function, starting from the 12<sup>th</sup> one, representing the hours.

On the other hand, the expression **COPY (NOW, 15, 2)** returns two characters from the NOW function, starting from the 15<sup>th</sup> character, representing minutes. Combining the two expressions with **COPY (NOW, 12, 2)+ COPY (NOW, 15, 2)** returns the exact time of the day, in hours and minutes.

It might be quite long but this is how programmers exploit the way computers and programs, like **CONTAX IVR** generates and handles data. With enough imagination, logic and persistence an application developer for **CONTAX IVR** can address most of the requirements of a company's telephony and interactive information systems.

## Chapter 8: Working with Databases

Another feature of **CONTAX IVR** that makes it a powerful application development platform for interactive information system is its ability to work with databases. In this chapter, you will get to know the following:

**What is a database?**

**How to use a database using DBQuery**

### 8.1 What is a Database?

Simply defined, a database is a collection of records. A record, on the other hand, is a set of related information. Think of a database simply as a set of data separated into sets that resemble a table structure. This table structure consists of individual data elements called columns or fields. A single set of a group of fields is known as a record or row. For instance, to create a relational database consisting of employee data, you might start with a table called EMPLOYEE that contains the following pieces of information: Name, Age, and Occupation. These three pieces of data make up the fields in the EMPLOYEE table, shown below:

#### **EMPLOYEE table**

Name	Age	Occupation
Will Smith	25	Electrical engineer
Dave Matthews	34	Museum curator
Jan Morrison	42	Chef
Bill Clinton	19	Student
Don Marcus	32	Game programmer
Becky Bouchard	25	Model

The six rows are the records in the EMPLOYEE table. To retrieve a specific record from this table, for example, Dave Matthews, a user would instruct the database management system to

retrieve the records where the NAME field was equal to Dave Matthews. If the instruction is to retrieve all the fields in the record, the employee's name, age, and occupation would be returned to the user.

SQL is the language that tells the database to retrieve this data.

**Note:** The initials stand for Structured Query Language, and the language itself is often referred to as "sequel." It was originally developed for IBM's (International Business Machine) DB2 product (a relational database management system, or RDBMS).

SQL is a nonprocedural language, in contrast to the procedural languages such as COBOL (Common Business Oriented Language) and C. **Nonprocedural** means **what** rather than **how**. For example, SQL describes what data to retrieve, delete, or insert, rather than how to perform the operation.

You don't have to learn SQL entirely in order to create **CONTAX IVR** applications that uses databases. At most you just have to learn a few basic SQL commands. Likewise you might have to learn about using common programs, which are actually implementations of SQL.

## **8.2 Creating a Database**

There are many popular implementations of SQL, or programs for the creation, management and use of databases. Each of these SQL implementations have their own unique capabilities as well as limitations.

One of these popular implementation of SQL is Microsoft Access, a PC-based data base management system (DBMS). It is relatively easy to use, featuring graphical user interface and wizards to help the user in creating and managing databases.

Another company, Oracle, has several implementation of SQL---like their Oracle 9i Database---but their products are normally used in large corporate databases. , but this is normally used in large corporate databases.

For our purpose, you can use Microsoft access to create databases that you can access and manipulate using **CONTAX IVR** DBQuery control cell. Please refer to Microsoft Access help files for specific instructions on creating and manipulating databases.

The following is a simple table of a database created using Microsoft Access:



ID	NAME
1	English
2	Japanese
27	Traditional Chinese
40	French
43	Thai
*	(AutoNumber)

### Important Points To Remember in Creating a Database

In designing your database, it is important to document, or provide key information like the following: a) the purpose of the database and who will be using it, b) description of each table within the database explaining its purpose and the internal structure of each table, including all fields and the type of data each field contains.

You should also break your data into separate components, or tables, to reduce repetition of data. This factor may seem trivial in small databases, but when you consider the demands for storage space of a large corporation and the costs involved, a properly-designed database would not only result in saving money but also result in easier management and retrieval of data.

### Open Database Connectivity (ODBC)

With so many vendors selling their own implementation of SQL, there should be a way in which a database made in one implementation can be recognized and used by another. ODBC is a functional library designed to provide a common Application Programming Interface (API) to database systems. It communicates with the database through a library driver, just as Windows talks to a printer via a printer driver. Depending on the database being used, a networking driver may be required to connect to a remote database.

With ODBC, you can use the same code to perform queries against a Microsoft Access table or an Informix database with little or no modification. However, it cannot be helped that most

vendors would add some their own extensions to the SQL standard, so you should always consult the documentation before beginning to work with a new data source.

ODBC has been adopted into many products and the SQL implementation you are using is probably ODBC compatible.

### **8.3 Harnessing the Power of a Database**

**CONTAX IVR** can tap into databases to extend the kind of information basic telephony alone can't provide, even with the use of automated PBX systems (PABX) and answering machines. This capability of tapping into the information stored in databases is what makes **CONTAX IVR** applications a truly interactive one—where the caller can retrieve information of his choice.

**CONTAX IVR** can look into the information contained in a database through the **DBQuery**

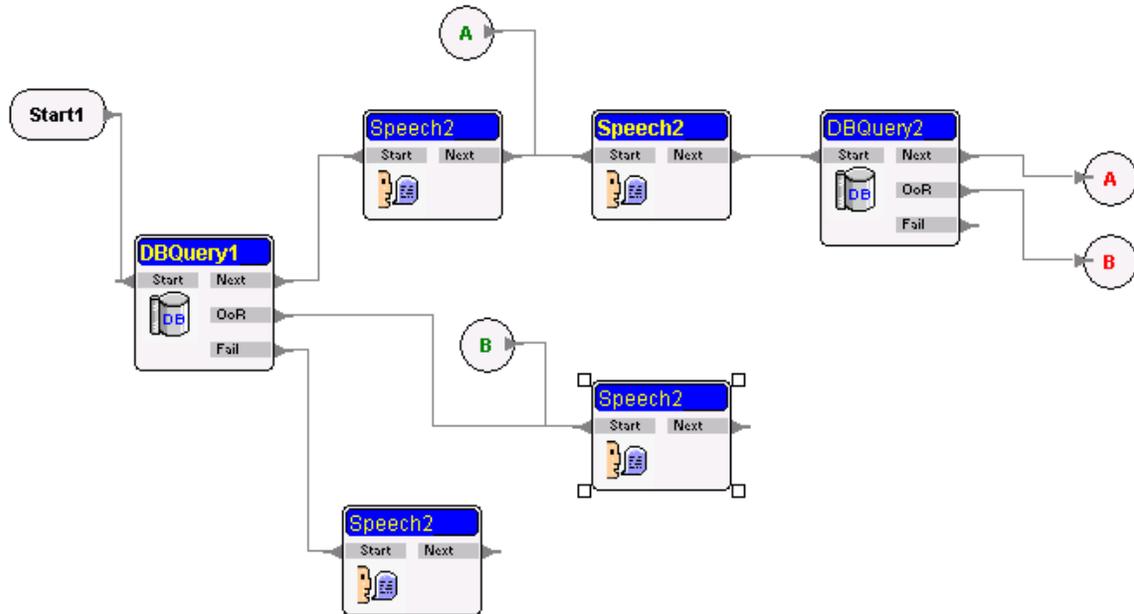


control cell.

When a call flow gets to this control cell, it executes a Database operation and then exits from its **Next** pin. If the operation result is out of range, not found, or database empty, the cell then exits from its **OoR** pin. If the operation is not successful, the cell then exits from its **Fail** pin.

To illustrate how DBQuery works, open the **CONTAX IVR** Call Flow Designer and click File>Open. From the ContaxIVR 2.1 folder, double click the Samples folder. When the folder opens, double click on the DBConnect folder and open the DBConnect.ivx file.

You should see the following call flow:



Run the program. It tells the caller the number of records in the database and tells the contents of one column: the name of the language.

How does it do it? Let us examine the call flow.

Select the DBQuery1 control cell and look at the values on the Property table pane. The first important property of the DBQuery control cell is the Conn Name (Connection Name). This is the name of the database being queried, in this case DB1. This is in fact the first property the computer looks into as when the DBQuery cell executes.

Conn Name	Db1
-----------	-----

Next, the computer looks for the SQL Command. In this case the value of the SQL Command property is the following:

SQL Command	Select * from Lang
-------------	--------------------

**Note:** SELECT coupled with FROM is the most basic of all SQL command or statement. The Select...From statement chooses the table where the computer would look into for information. Thus, the **Select \* from Lang** statement indicates that the query would be made on the Lang table. (Please refer to the first database example provided above). The \* in the statement tells the database to return all the columns associated with the given table described in the From clause.

Next, the computer would look into the **Action** property. Notice that under this property, the value is set to **Query Open**, meaning the call flow opens the database for query and looks into the records it contains.

Once it makes a successful query into the database, the DBQuery puts the number of records it found into the value of the Record Count Variable, in this case, to the **[RCnt]** variable. In our example, the computer finds that there are five records in the database and stores this number in **RCnt**.

When the call flow exits from the Next pin of the DBQuery1 control cell, it executes the connecting **Speech2** cell. This cell does two things, play a TTS of the message telling how many records were in the table, based on the value of the **RCnt** variable, as shown in the figure below:

Text	There are [RCnt] records in this database.
------	--

The cell then puts the value of **RecNo** (record number) variable as 1 and exits into the next cell. The next control cell, another Speech cell, in turn, does two things: 1) Plays a message using the value of the current value of the **RecNo** variable plus the value of the first record in the **Db1** database under the **NAME** column. This is made possible by the following expression:

Text	Language [RecNo] is "[Db1.NAME]".
------	-----------------------------------

In addition, the Variable Assign property of the same cell increments, or increases the value of the RecNo variable by re-defining it as **[RecNo]=[RecNo]+1**.

Another **DBQuery** cell then follows. And why is this one necessary? The answer lies in the value of the **Action** property: **Move Next**. As the term implies, the action commands the computer to look into the next record and loops back into the second **Speech2** cell until the call flow exhausts all the records. When the computer reaches the end of record, it would naturally

return a blank record. The **DBQuery** cell would then exit in its **OoR** (Out of Range) pin and plays the corresponding message in a **Speech** cell for such purpose.

## **Chapter 9: Working with Fax, E-mail and SMS**

**CONTAX IVR** also has the capability to automate the sending and receiving of fax and e-mail, as well as using the short-messaging system feature of compatible Nokia phones, making **CONTAX IVR** an all-around communications platform. In this chapter, you would become familiar with the workings of the following control cells:

SendFax

RecvFax

MakeCall

SendEMail

RecvEMail

SendSMS

### **9.1 Sending Faxes**

With **CONTAX IVR** integrated fax features, you can design an application to enable fax on demand services: to send faxes anytime of the day, to allow customers to select from available documents and have their selection faxed on to them immediately or by later call back.

#### **Sending a Fax**

**CONTAX IVR** sends a fax document through its SendFax Module.



However, the SendFax module does not have the capability to dial a fax number. This job is done by a previous control cell.

In case the call flow application sends fax on the same number that started the call, the **SendFax** module can simply be used after informing the caller to give a fax tone. But when the application

calls for a call back—sending the fax later on the same number, or through another fax number---  
-the sending of faxes can be done, usually, by using the **MakeCall** control cell.



The MakeCall cell is actually a kind of a SubFlow control cell. When triggered, it calls either a local or outside line and executes another call flow. After this, the module exits on its Next pin.

### Fax Formats

The SendFax and RecvFax control cells support the following file formats:

- Text (.txt)
- TIFF (.tiff)
- JPEG (.jpg)
- GIF (.gif)
- BMP (.bmp)
- PCX (.pcx)
- DCX (.dcx)

When the SendFax control cell is activated, it looks into its Source File property, where the image file to be sent, in any of the above formats, are specified.

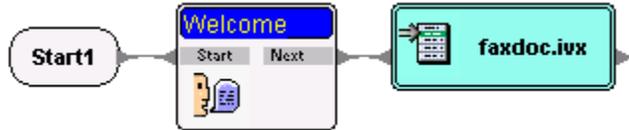
## 9.2 Application No: Fax on Demand

To illustrate just how fax on demand works, open **CONTAX IVR** Flow Designer and open the Samples folder. Open **Sendfax1.ivx**.

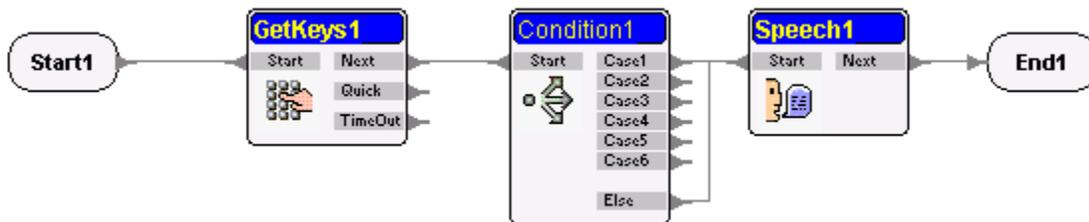
This fax on demand application actually consists of three separate ones. **Sendfax.ivx** is the main application that calls two other applications: one, through a **Subflow** control, and the other, through the **MakeCall** control.

## Run the application.

You will notice that after the call and the welcome message is played, your call flow will execute the SubFlow named **faxdoc.ivx**.



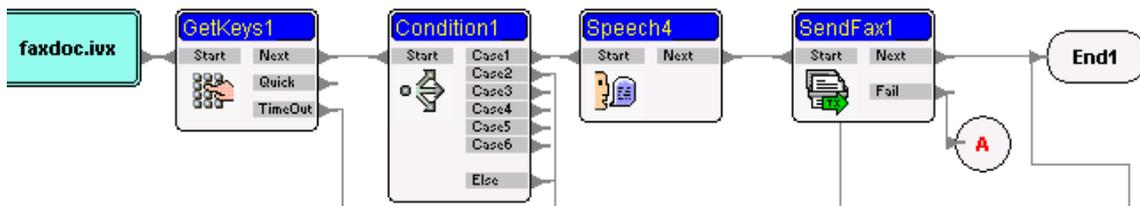
Now, *faxdoc.ivx* consists of the following call flow control cells:



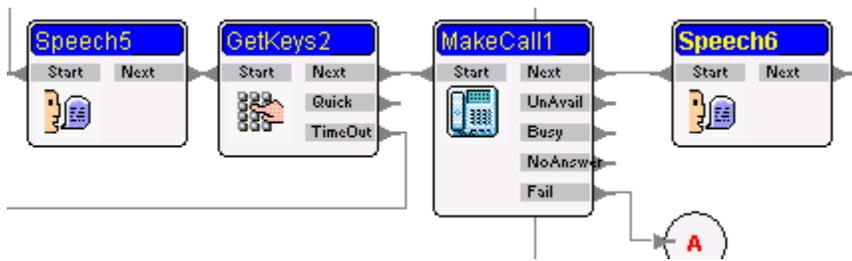
This subflow informs the caller of available documents from which he could choose by entering a certain number on the phone's touch pad, after which the subflow ends and goes back to the main flow. But take note that under Condition1, the variable **faxdoc** is assigned to the selected document. You will notice later that the same variable is passed to the main application and used to determine the proper document to send to the caller.

```
Variable Assign [faxdoc] = IF( [DocID] = 1 ,
```

When the *faxdoc.ivx* exits, the execution of the main program *sendfax1.ivx* continues.



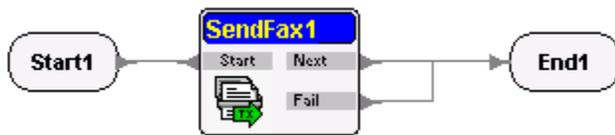
The GetKeys1 gives the caller a choice of having the document faxed right away or on a later callback; the input is then processed by Condition1. If the caller wants to get the fax immediately on the same line, the Speech4 module instructs him to send a fax tone by pressing the Start button of his fax machine and the SendFax1 module kicks in to send the fax. Notice that in each of the control cells from GetKeys1 to SendFax, all were using the global variable **faxdoc** to determine which fax files to send.



If the caller wants to receive the fax on another number, the MakeCall control cell can take care of that. In the above example, the Speech module provides the instructions for the caller to input his number, which the GetKeys2 cell processes and passes on to the MakeCall1 control cell. The latter dials the number the caller provided and triggers into action another call flow, the sendfax2.ivx specified under the control cell's Call flow Filename property.

Call flow Filename	sendfax2.IVX
--------------------	--------------

Sendfax2.ivx is actually a very simple call flow that merely uses the number MakeCall has dialed and sends the fax. It consists of the following elements:



### 9.3 Receiving a Fax

Just as it can automate sending a fax, **CONTAX IVR** handle receiving of faxes automatically, through the use of the RecvFax control cell.



The RecvFax control cell receives incoming fax and saves the Fax to an assigned file. The fax file is saved in the directory specified under the cell's **Destination path** property, like **C:\Recv\_docs** for example.

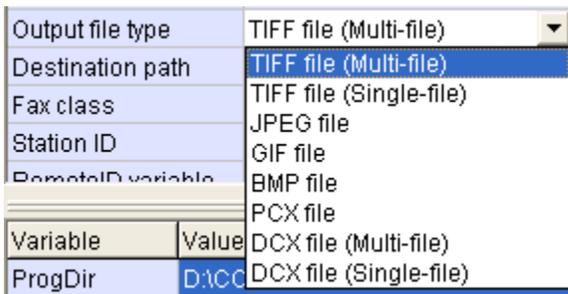
The fax filename is determined by the **Name Mode** property. You can choose from either the Channel-Serial-page mode or the Channel-MonthDay-page mode.



If you choose the **Channel-Serial-page** mode, each fax page will be saved with the filename automatically assigned by Channel Number, a Serial number (accumulated from the total fax events in the system), and **page** number. So a fax file with the name **5-28-1** means the first page for fax serial number 28, received on channel 5.

Under the Channel-MonthDay-page mode, each fax page will be saved with the filename taken from the channel used, the day and time, and the number of the page. A fax file name of **7-04101731-1** means the first page of the fax received on 17:31 of April 10, through channel 7. The time stamp is taken when the cell completes receiving the fax and before saving the file. Since a fax process, including the handshaking, exceeds a minute, there won't be any duplication of a fax file saved.

The received faxes will be saved under the formats also used by the SendFax module. You can specify the type of file under the **Output file type** property.



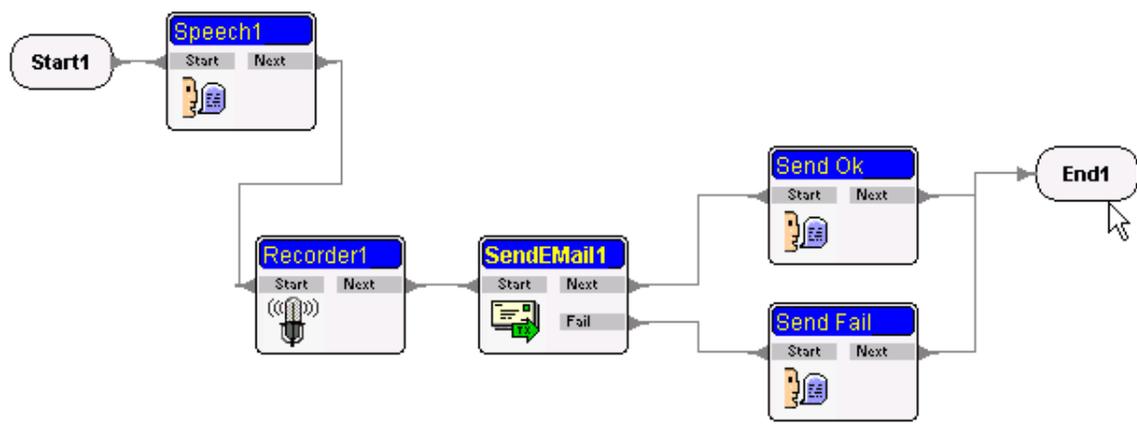
## 9.4 Working with E-Mail

Apart from the capabilities of **CONTAX IVR** we have discussed so far, it has the added flexibility of the automated sending and receiving of e-mail messages through the **SendEMail** and **RecvEMail** call control cells.

To illustrate these capabilities better, let us look into another example.

### 9.4.1 Application No. 7: Send Mail

Go into the **CONTAX IVR** installation directory, open the Samples folder. Double click the Email folder and open the SendEmail folder inside it. Open the SendEmail.ivx. You should see something like this in the **CONTAX IVR** Flow Designer.



The application instructs the caller (through Speech1) to record a message (Recorder1), after which, the call flow automatically sends an e-mail message to the caller.

You are already familiar with the first two call control cells so let us focus on the SendEmail control.

This cell sends email via the SMTP (simple mail transfer protocol) server specified in the Property pane. It detects whether or not you are connected to the Internet and if not, dials your default connection to send the email. The sender's email address and the recipient, including those who will receive a carbon copy (CC) or blind carbon copy (BCC) are specified in the respective value fields in the property table, as shown below:

From Address	contaxivr@elettech.com
Recipient	contaxivr@elettech.com
CCList	
BccList	

To send the same email to several recipients, indicate the various email addresses, separated by a comma.

**Note:** You can use a string variable to represent an email address or many email addresses. For example, [sales] to represent all email addresses for the sales department.

Email address variables can be used with database operation to create powerful e-commerce or v-commerce tools. For example, when a caller purchases a product, an email will automatically be sent to the product's supplier and/or its dealer. The email Subject and/or content (Body) may be different regarding whom it is sent to.

You also need to specify in the property table any the **Subject** and the **Body** of the email. To write the body of the message, double click on the value column of the Subject property and then click on the ellipses. The Text Builder would then appear and you can type your message there, or paste any other prepared message that you have made in another email application or even a word processor program. If there is any, you could also specify an attachment through the property table pane.

To successfully send the message, you have to specify in the property table the SMTP server you are using **CONTAX IVR** uses the default number 25 as port of the SMTP server.

#### 9.4.2 Receiving Email

Although sending electronic mail is relatively simple, receiving email is a little bit more complicated. You have to remember that the RecvEMail control cell has two basic functions: 1) to query the server if you have any email message; and 2) to retrieve the email from your email box.



You can specify what specific action you want done (to query the server or retrieve messages) in the value field of the cell's **Action** property.

The RecvEMail control cell receives email from a POP3 (Post Office Protocol) server, specified in the POP3 Server property. The default port for the POP3 server is 110. Ask the system administrator for the name of your POP3 server and the port it uses.

You should also specify the **User ID** for the email account and the appropriate **Password**. As an alternative, you may specify a variable to get the caller's input via the GetKey cell and compare it to a database of email User ID's and passwords.

In the property table, you can also indicate the variables to store the Sender (Sender Variable), the Date of the message(Date Variable), the Subject (Subject Variable), the Body of the message

(Body Variable), the attachment (Attachment Variable)—if any—and the directory where you want the attachment saved (Attachment Path), as shown in the example below:

Retrieve content : RecvEMail	
Property	Value
Retrieve Body	False
MessageCount Variable	
Sender Variable	[Sender]
Date Variable	[EMailDate]
Subject Variable	[Subject]
Body Variable	[Body]
Attachment Path	[ProgDir]
Attachment Variable	[Attachment]

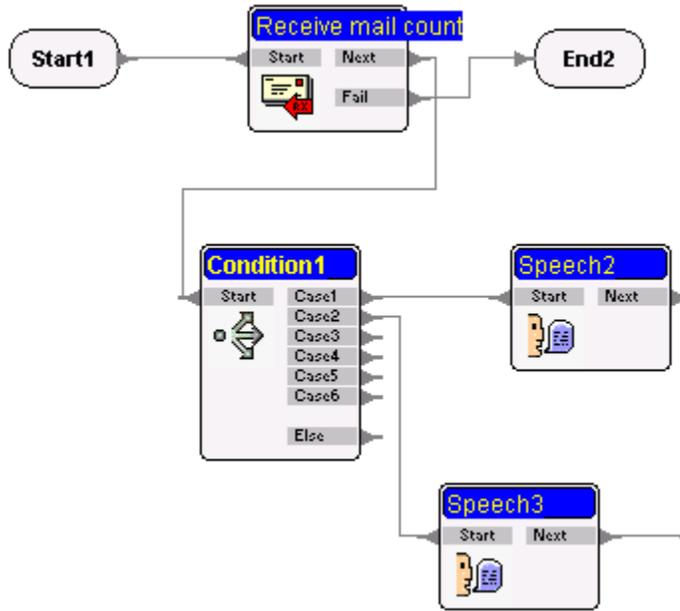
*To make things clearer, let us examine another example.*

### 9.4.3 Application No. 8: Receive Email

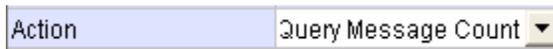
You may wonder, where would a **CONTAX IVR** application for receiving email be useful? Consider for example that you are a salesman and working in the field. Now, you have a client who emailed you the things he wants to buy from your company but you have no PC to retrieve the email nor a remote terminal where you can log on to your POP3 server to read the message. With a Receive Email application, **CONTAX IVR** can retrieve your email. Using a touch-tone phone you can dial your company and let **CONTAX IVR** TTS capabilities to read the message to you. This capability is shown in the example call flow, RecvMail.ivx.

Open the Samples folder in the directory of **CONTAX IVR**. Open the Email folder and double-click on the folder labeled "RecvMail ". Click the RecvMail.ivx to open the application in the Flow Designer.

You will notice that immediately after the Start button, you already have a RecvEMail cell labeled "Receive mail count".



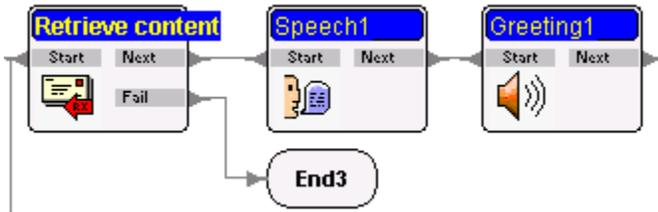
If you click on this cell and look into the property table, you will find out that it's meant only to query the POP3 server for messages, as indicated in the value of the Action property.



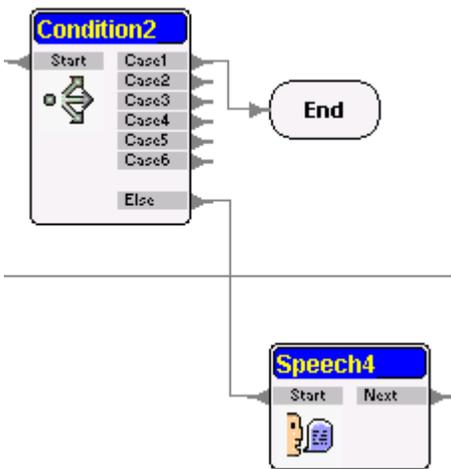
The flow then activates Condition1 where the application branches to two separate flows. If there is no email message, the call flow triggers the Speech cell to inform the caller that there aren't any messages for him. But if there are incoming email messages Condition1 exits through its Case2 pin and activates the Speech3 cell, telling the caller that he has a number of messages.

Another **RecvMail** control cell (Retrieve content) then executes, with the Action property set to "**Retrieve Mail**", to get the incoming emails from the POP3 server. Take note that under the Property table, the variable **[a]** is assigned to the Message Index property and that in the Variable Assign property, the following: **[a]=[a]+1**.

**Note:** The variable assignment is meant to increment---increase sequentially---the count of the message processed so that the call flow can stop when all the email messages are processed. You will encounter the same variable later.



The **Speech1** control reads to the caller the details of the email message (Date, Sender, Subject, Attachment); the next cell plays the wave file attached to the email. Next, the call flow executes another Condition cell.



Take careful note that the Condition2 cell compares the value of [a], the variable we have been tracking, with the value of the [MsgCount] variable the first RecvEMail control cell has retrieved earlier.

Case1	[a] > [MsgCount]
-------	------------------

If the value of [a] has not exceeded that of **[MsgCount]** the call flow would go back to the **RecvEMail** control cell to process another email message, and so on, until all the messages from the POP3 server for the specified email address were processed.

## 9.5 Sending Information using SMS (Short Messaging System)

With the increasing use of mobile phones, it has become possible for companies to be in touch with their employees and clients even when they are out of the office. However, calls to mobile phones are generally more expensive than calls to local numbers. In some cases, it is cheaper for the company to take advantage of the SMS capabilities of mobile phones to send messages.

**CONTAX IVR** also included in its arsenal of features the ability to send SMS. This is made possible through the **SendSMS** control cell.



This cell sends SMS (Short Message Service) to a target mobile phone and then the cell exits from its **Next** pin. If it fails to send the SMS, then it exits from its **Fail** pin.

This control cell can send SMS message in the following configuration:



You may use the basic configuration shown above as a **Subflow** to add to your existing application.

However, this feature is compatible only with the more popular Nokia Phones. Compatible Nokia mobile phones include the following models:

- Nokia 3xxx (3210, 3310 etc.)
- Nokia 51xx
- Nokia 61xx
- Nokia 62xx
- Nokia 71xx
- Nokia 8xxx

Specify the correct model of the mobile phone you are using in the SMS Model property. The mobile phone is installed beside **CONTAX IVR** runtime server connected to one of the COM ports on the server.

Just click on the value field of the SMS Port property to select a serial port on **CONTAX IVR** runtime server to send short messages from the server via the mobile.

You can connect up to 8 mobile phones with COM1 through COM8. Connection is either through the proper data cable or via infrared ports. You should indicate the proper connection in the SMS Mode property. (Please refer to the mobile phone manual for the appropriate connection modes).

The **SendsSMS** control cell dials the desired mobile phone number as indicated in the value field of the Target property. Here, you may use a variable to hold the desired mobile phone number determined via a DBQuery operation.

You can compose the SMS message you want to send by double-clicking on the value field of the **Contents** property and then clicking the ellipses. The **Text builder** dialog box appears. Enter the content on the **Text String** box. You can insert variables among the text to have a programmable content by selecting variables from the **Available variables** box and then clicking the **Add** button.

You can also copy the content of a file and paste it to the **Text String** box.

There is no limit to the maximum number of characters you can enter in the **Text String** box. But most mobile phone limits the message to no more than 140-160 characters in length (depending on the mobile service system).

If a double-byte mobile phone set is used (e.g. in China, Japan, Korea, Taiwan, Hong Kong, etc.), the character length is around 70. The mobile phone will not send all extra characters, exceeding the character length limit.

## 9.6 Web Integration

**CONTAX IVR** also has the capability to tap into the Web to extend its functionality via the **CallURL** control cell.



This cell links to an URL (Uniform Source Locator or the Internet address you want the call flow to call).

The Internet address or Web page may be an Active Server Page (ASP) that contains HTML (hypertext mark up language) and embedded programming code written in VBScript or Jscript. When Microsoft's Internet Information Server (IIS) encounters an ASP page requested by the browser from this cell, it executes the embedded program. This will allow the call flow to interact with databases and other programs via the called Web pages. The **CallURL** enables the **CONTAX IVR** application to derive data or information otherwise not possible with the use of **DBQuery** control cell alone.

To enable the control cell to call the proper Internet address, enter the desired address in the value field of the **Web URL** property.

Study the example provided in the **Samples** folder of the **CONTAX IVR** to find out how **CallURL** control cell works and how you can use this to provide more functionality for your applications. Open the **CallURL** folder and click the .ivx file of the same name.

The application calls an ASP page that calculates how many more days would come before your birthday.

Applications using CallURL controls are some of the more advanced techniques in **CONTAX IVR** application development and should be properly dealt with more extensively in development guides for more experienced programmers.

## 9.7 Other Control Cells

Apart from those discussed previously, **CONTAX IVR** has provided other control cells to enable application developers to address almost every conceivable communication needs. The following control cells may be used for more advanced application designs. For now, we will simply discuss their characteristics.

### 9.7.1 Conference



This cell connects a call with an extension or with an outside phone.

You can specify the desired extension number or outside line in the value field of the **Target** property. e.g. 2385 or +1 (408) 766-1234.

If you set 0 or 9 for **Outgoing prefix dialing** in the **Runtime Engine**, you do not need to enter the prefix here to get an outside line.

If you are dialing a local number but the country code and area code are entered in **Target**, then the country code and area code will be screened providing you have set them in your Windows system (**Start -> Settings -> Control Panel -> Phone and Modem -> DialingRules/My Location**). The system will only dial the local number for you.

You may enter the local phone number without country code and area code in **Target** if you are sure it is a local number.

### 9.7.2 Switch



This cell is used to connect two channels in T1/E1 connection. Specify the desired channel number in the value field of the **Target Channel** property.

In the **Switch Action** property, you may select from three options available in the drop down list”

1. **Connect**: The cell connects the other channel then exits from its **Next** pin.
2. **Monitor**: The cell monitors the other channel then exits from its **Next** pin. If the other channel hangs up, the cell then exits from the **Close** pin.
3. **Disconnect**: The cell disconnects the other channel then exits from its **Next** pin.

Once you have completed the application you developed, you must test it properly and remove the bugs---errors or program flaws---before using it in actual work situation.

# Unit V-Program Deployment and Monitoring

Unlike .com or .exe files, **CONTAX IVR** applications are not executable files and so they do not run by themselves alone. In the preceding two short chapters, you will learn the following:

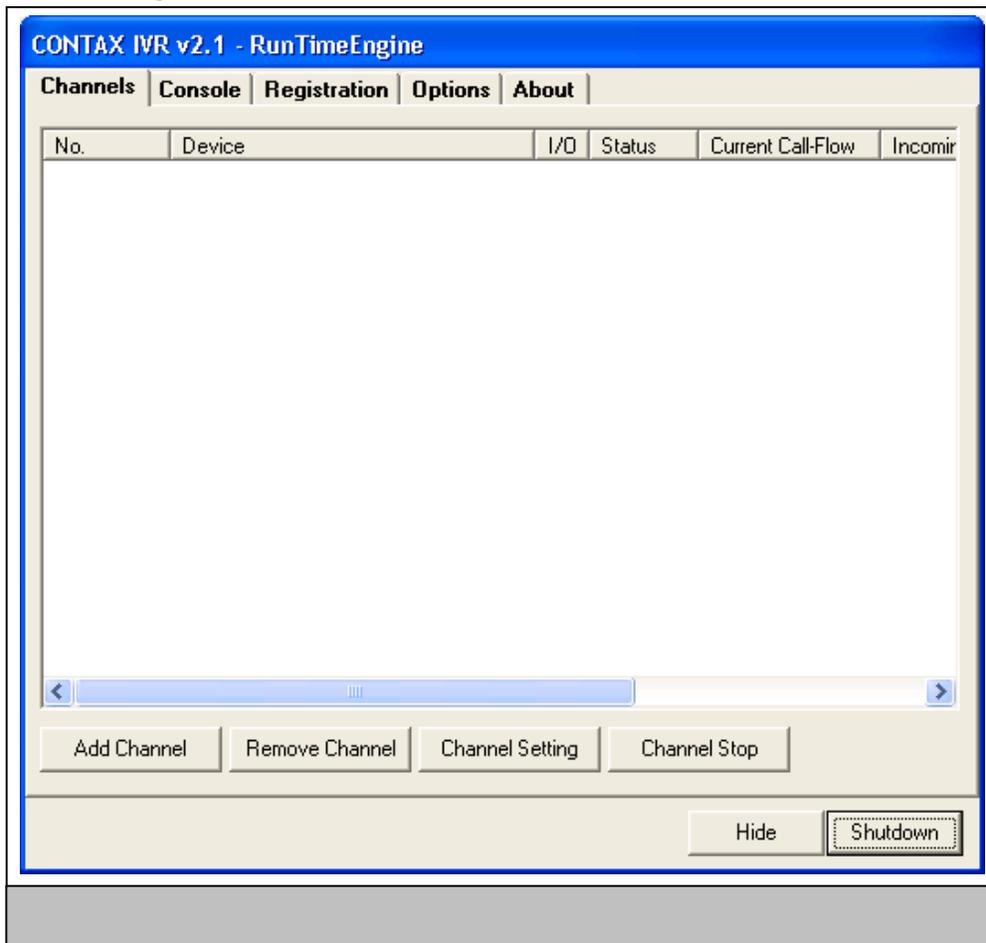
- Deploying application through the Runtime Engine
- Monitoring applications

## Chapter 10: Program Deployment

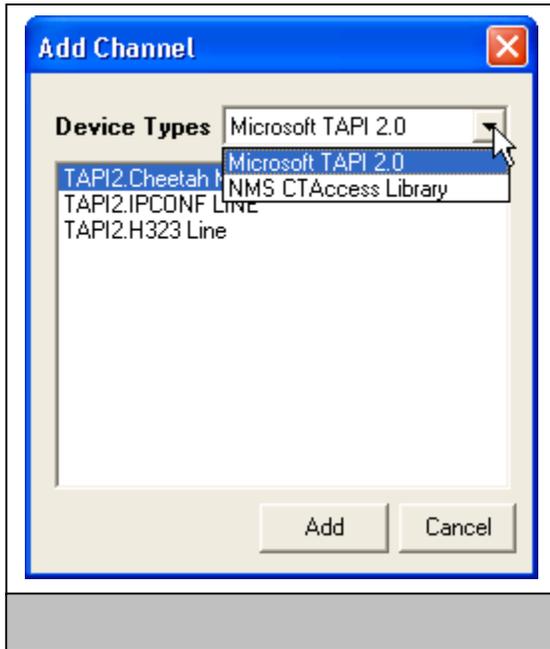
You can select which application to use through the **CONTAX IVR** Runtime Engine. If you click the shortcut of the Runtime Engine, it would run in the background and you will only notice it as a small telephone handle icon in the system tray:



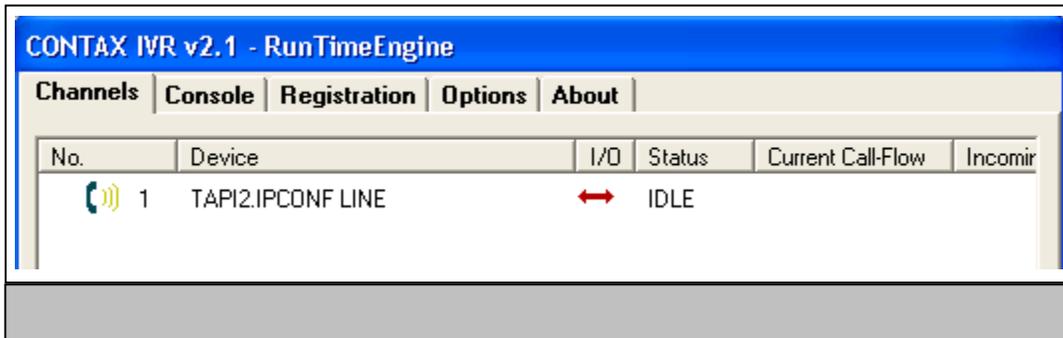
However, if you double-click the tray icon, the Runtime Engine panel pops-up and you will see the following:



Click the Add Channel button and the dialogue box opens:



Select the appropriate channel and it will be added into the Runtime Engine panel:



Depending on your license and voice board, you may be able to run several active channels and call flows at the same time. You can run the desired call flow by clicking on the channel and then on the Channel Setting button. Once you do, the following dialogue box appears:

**Channel Setting**

**Channel Information**

Channel No : 1

Device Name : TAPI2.IPCONF LINE

**Channel Configuration**

Description :

Phone Number : 5443950

Incoming Call-flow file : D:\CONTAX IVR v2.1\Samples\KeyPress\KeyPre ...

CallURL WebServer : 127.0.0.1

Internal prefix dialing :

Outgoing prefix dialing :

Voice Encoding Format : PCM 8kHz 8Bit Mono

FaxDevice ( ComPort ) : Unused

Enabled Inbound  Enabled Outbound

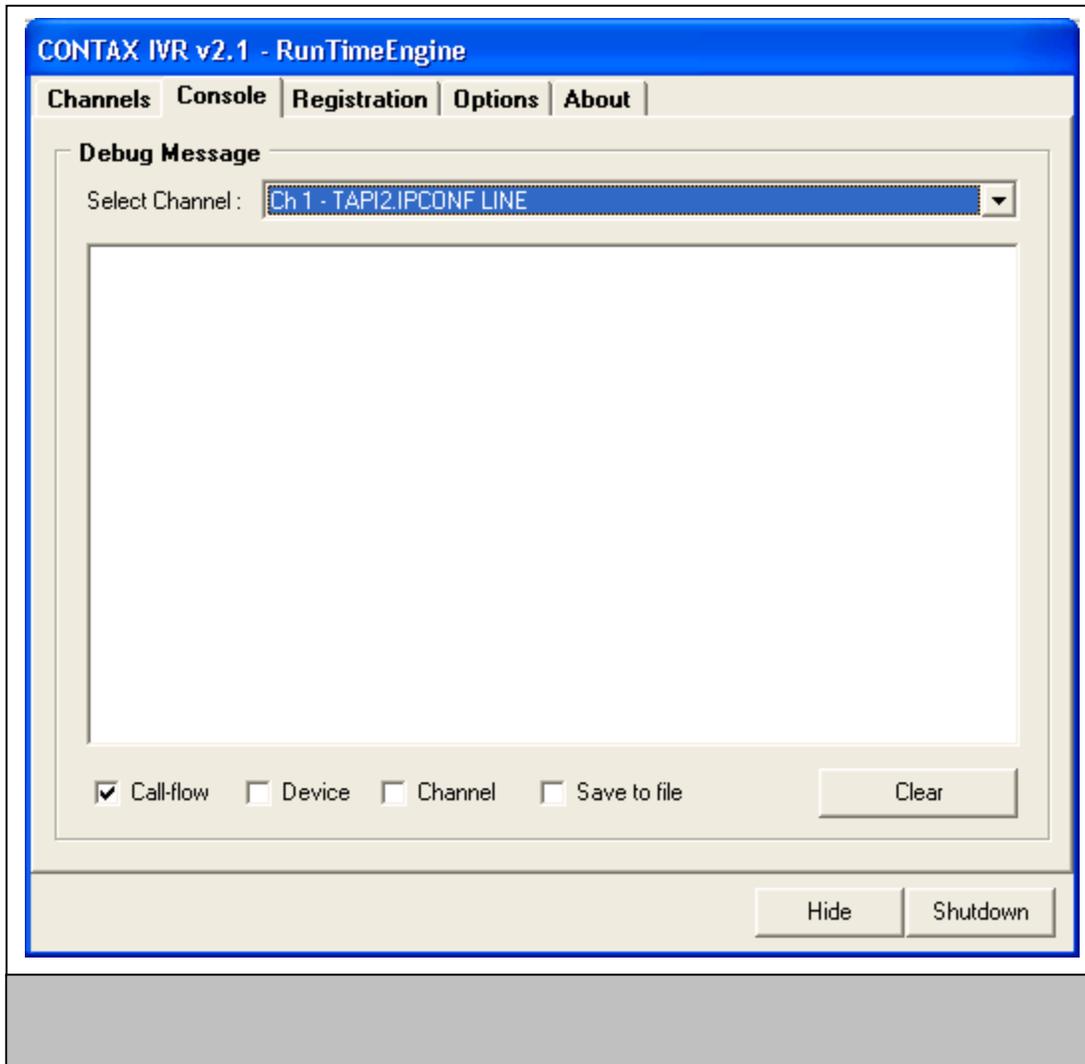
Apply to all channels

Ok Cancel

Click on the ellipses (...) in the right side of the **Incoming Call-flow** file box and an open file dialogue box will open. Go into the directory of where your call flow application (.ivx file) was saved and click on the desired call flow. Click Open and the application's directory will be reflected into the Incoming Call-flow file box.

Indicate the appropriate PBX/PSTN phone number in the "Phone Number" box. Fill in the other boxes as necessary. (For other settings please refer to the **CONTAX IVR** User's Guide). Click the **Ok** button and your application is now up and running!

To monitor or debug an running application, click the **Console** tab. From the **Select Channel** drop-down list, click to select a channel. You will see the runtime activities of the channel displayed on the **Console** tab window.



There are four check boxes you may select for the following options:

**Call-flow:** displays the call flow components log

**Device:** displays Device information

**Channel:** display Channel information

**Save to file:** save the Log Data to a file named install directory \ ChX.log. (file path is same as RunTimeEngine.exe)