Contents

About Avaya IVR Designer ................................................................. 9

Introduction to Avaya IVR Designer .................................................. 9
System requirements ........................................................................ 10
Installing Avaya IVR Designer on your PC ........................................ 11
Internationalization and globalization ............................................. 13
Checking for the SCI tools on the target system ............................. 15
The graphical user interface .............................................................. 15
Getting started with IVR Designer .................................................. 17
Converting Voice@Work and IVR Designer applications .................. 17
Creating an application using Avaya IVR Designer ......................... 18
Plan the application design .............................................................. 18
Create the call flows and other application components ............... 19
Generate the source code ................................................................. 20
Transfer the application to the target system .................................. 21
Install the application on the target system .................................... 21
Setting user preferences ................................................................. 21
The Application Parameters tab ....................................................... 22
The Environment tab ...................................................................... 22
The Printing tab ............................................................................ 24
About Help .................................................................................... 24
About the Help window .................................................................. 25

Voice application basics ................................................................. 28

Voice response application uses .................................................... 28
Voice response system versus an attendant ..................................... 30
Defining successful voice response applications ........................... 30

Planning a voice response application ............................................ 32

Use human factors and usability engineering resources ................ 32
Use prerecorded speech ................................................................. 33
Offer a user guide .......................................................................... 33
Offer caller training ....................................................................... 34
Provide an attendant ...................................................................... 34
Plan for disabled callers ............................................................... 34
Diagram your application design .................................................... 35
## Contents

Use flowcharts ............................................................................................................35  
Use outlines ................................................................................................................36  

**Designing a voice response application** .................................................................37  
  Application usability ...................................................................................................37  
  Prompts ......................................................................................................................40  
  Announcements .........................................................................................................42  
  Menus ........................................................................................................................43  
  Digit input ..................................................................................................................45  
  Entering digit sequences ..........................................................................................46  
  Yes/no questions ........................................................................................................47  
  Pace the application ..................................................................................................48  
  Application errors .....................................................................................................49  
  Touch-tone and speech recognition .........................................................................50  
    Application examples ...............................................................................................51  
    Touchtone input used with spoken input ..................................................................52  
    Use touch-tone input when speech recognition fails ..............................................54  
    Modify a touch-tone-only application to include spoken input ..............................55  
  Dial through and barge-in .........................................................................................56  
  Text-to-Speech in applications ................................................................................57  
    Using Text-to-Speech for prompts and announcements ......................................57  
    Using both Text-to-Speech and prerecorded speech prompts and announcements ..58  
    Getting the most out of Text-to-Speech ...............................................................58  
    Testing Text-to-Speech applications ....................................................................59  

**Creating a voice response application** ..................................................................61  
  Working with nodes ..................................................................................................61  
    Placing a node in a call flow ..................................................................................62  
    About node branches .............................................................................................63  
    Node editing guidelines .........................................................................................64  
    Name conflicts during paste actions .....................................................................65  
    Selecting multiple nodes for cut, copy, and paste ...............................................65  
    Using drag-and-drop to cut, copy, and paste .......................................................66  
    Editing node attributes .........................................................................................67  
    Right-Click menu options for nodes and empty node branches ............................67  
  The Node Inspector ..................................................................................................70  
    Selecting a node to view or edit attributes ............................................................71  
    Writing node notes .................................................................................................71  
    Right-click menu options for the Node Inspector ...............................................72  
  Searching for information in applications ................................................................72  
    The Named tab ........................................................................................................73
Verifying and testing an application ................................................................................. 139
  Checking an application for errors .............................................................................. 139
  Right-click menu options for the Verification Results window ...................................... 140
  Simulating an application ............................................................................................ 141
  Running an application simulation ............................................................................. 142
  Examining the simulation script ................................................................................ 144
  Right-click menu options for the Simulation tool .......................................................... 144

Generating and transferring an application ..................................................................... 146
  Generating a VoiceXML application ............................................................................ 146
  Generating the VoiceXML application files ............................................................... 148
  The code generation process ................................................................................... 148
  Menus in the Code Generation window ....................................................................... 149
  Transferring the application ......................................................................................... 150
  Setting application transfer options ........................................................................... 150
  Transferring files from floppy disks to the target system ........................................... 153
  Transferring application source files ......................................................................... 154
  Application Transfer window menus ......................................................................... 154
  Transferring files using a modem ................................................................................ 155
  Windows 2000 PPP configuration ............................................................................. 156
  Transferring an application to the target system ......................................................... 159

Assigning an application on the target system ............................................................... 162
  Assigning a VoiceXML application to channels or numbers ......................................... 162

VoiceXML application development tours ....................................................................... 164
  Basic VoiceXML application development tours .......................................................... 165
  VoiceXML IVR Designer Interface .............................................................................. 166
  Configuring the VoiceXML IVR Designer desktop ....................................................... 173
  Creating a basic VoiceXML application ..................................................................... 181
  Testing a VoiceXML application ................................................................................ 199
  Generating and assigning VoiceXML applications ....................................................... 202
  Branching based on time, day, or date in a VoiceXML application ................................ 212
  Copying nodes and VoiceXML applications .................................................................. 219
  Presenting menus in VoiceXML applications ................................................................ 224
  Collecting information in VoiceXML applications ...................................................... 233
  Using calculations in VoiceXML applications .............................................................. 240
  Using variables and speech in VoiceXML applications ............................................... 247
  Using embedded call flows in VoiceXML applications ................................................. 263
  Advanced VoiceXML application development tours .................................................. 272
  Handling strings in VoiceXML applications .................................................................. 272
  Using asynchronous events in VoiceXML applications ................................................. 288
  Running TAS applications from VoiceXML ............................................................... 292
© 2005, Avaya Inc.  
All Rights Reserved  
Notice  
Every effort was made to ensure that the information in this document was complete and accurate at the time of printing. However, information is subject to change.

Preventing Toll Fraud
“Toll fraud” is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or working on your company's behalf). Be aware that there may be a risk of toll fraud associated with your system and that, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Fraud Intervention
If you suspect that you are being victimized by toll fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1 800 643 2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Web site: http://www.avaya.com

Support > Escalation Management > International Services.

Providing Telecommunications Security
Telecommunications security (of voice, data, and/or video communications) is the prevention of any type of intrusion to (that is, either unauthorized or malicious access to or use of) your company's telecommunications equipment by some party.

Your company's "telecommunications equipment" includes both this Avaya product and any other voice/data/video equipment that could be accessed via this Avaya product (that is, "networked equipment").

An “outside party” is anyone who is not a corporate employee, agent, subcontractor, or working on your company's behalf. Whereas, a “malicious party” is anyone (including someone who may be otherwise authorized) who accesses your telecommunications equipment with either malicious or mischievous intent.

Such intrusions may be either to/through synchronous (time-multiplexed and/or circuit-based) or asynchronous (character-, message-, or packet-based) equipment or interfaces for reasons of:

- Utilization (of capabilities special to the accessed equipment)
- Theft (such as, of intellectual property, financial assets, or toll-facility access)
- Eavesdropping (privacy invasions to humans)
- Mischief (troubling, but apparently innocuous, tampering)
- Harm (such as harmful tampering, data loss or alteration, regardless of motive or intent)

Be aware that there may be a risk of unauthorized intrusions associated with your system and/or its networked equipment. Also realize that, if such an intrusion should occur, it could result in a variety of losses to your company (including but not limited to, human/data privacy, intellectual property, material assets, financial resources, labor costs, and/or legal costs).

Your Responsibility for Your Company's Telecommunications Security
The final responsibility for securing both this system and its networked equipment rests with you - an Avaya customer's system administrator, your telecommunications peers, and your managers. Base the fulfillment of your responsibility on acquired knowledge and resources from a variety of sources including but not limited to:

- Installation documents
- System administration documents
- Security documents
- Hardware/software-based security tools
- Shared information between you and your peers
- Telecommunications security experts

To prevent intrusions to your telecommunications equipment, you and your peers should carefully program and configure:

- Your Avaya-provided telecommunications systems and their interfaces
- Your Avaya-provided software applications, as well as their underlying hardware/software platforms and interfaces
- Any other equipment networked to your Avaya products.

Federal Communications Commission Statements
Part 15: Class A Statement
This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Part 68: Answer-Supervision Signaling
Allowing this equipment to be operated in a manner that does not provide proper answer-supervision signaling is in violation of Part 68 rules. This equipment returns answer-supervision signals to the public switched network when:

- Answered by the called station,
- Answered by the attendant, or
- Routed to a recorded announcement that can be administered by the CPE user.

This equipment returns answer-supervision signals on all direct inward dialed (DID) calls forwarded back to the public switched telephone network. Permissible exceptions are:

- A call is unanswered.
- A busy tone is received.
- A reorder tone is received.

Canadian Department of Communications (DOC) Interference Information
This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.
European Union Declarations of Conformity

Avaya Inc. declares that the switch equipment specified in this document bearing the ‘CE’ (Conformité Européenne) mark conforms to the European Union Radio and Telecommunications Terminal Equipment Directive (1999/5/EC), including the Electromagnetic Compatibility Directive (89/336/EEC) and Low Voltage Directive (73/23/EEC). This equipment has been certified to meet CTR3 Basic Rate Interface (BRI) and CTR4 Primary Rate Interface (PRI) and subsets thereof in CTR12 and CTR13, as applicable.

Copies of these Declarations of Conformity (DoCs) signed by the Vice President of R&D, Avaya Inc., can be obtained by contacting your local sales representative and are available on the following Web site:
http://support.avaya.com

Trademarks
Avaya, CONVERSANT, and Intuity are registered trademarks of Avaya, Inc.
Adobe and Adobe Acrobat are trademarks or registered trademarks of Adobe Systems, Inc. in the United States and in other countries.
U.S. Robotics and Sportster are registered trademarks of 3Com Corporation or its subsidiaries.
Alliance Generation is a registered trademark and NMS Communications, Natural MicroSystems, AG, Natural Access, NaturalFax are trademarks or service marks of NMS Communications Corporation or its subsidiaries.
Informix, DB2 are registered trademarks of IBM Corporation.
Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Netscape is a registered trademark of Netscape Communications Corporation in the United States and other countries. End-users are not entitled to receive from Netscape any support, warranty, or documentation associated with the Netscape product unless they have paid for such support, warranty or documentation. End user can purchase support from Netscape, at http://help.netscape.com.
Nuance and Nuance Vocalizer are trademarks of of Nuance Communications, Inc.
Oracle is a registered trademark, and Oracle8i, and Oracle9i are trademarks or registered trademarks of Oracle Corporation
Sybase is a trademark of Sybase, Inc.
Speechify, OpenSpeech Server, OpenSpeech Recognizer, and OpenVXI are registered trademarks of SpeechWorks International, Inc.
Sun, Sun Microsystems, docs.sun.com, Java, Solaris, Sun Blade, and Sun Fire are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries.

Technical Support
To report problems or to request assistance setting up and using your system, contact the Avaya Technical Services Organization (TSO). The telephone number for support in the United States is 1-800-242-2121.
For additional support telephone numbers:
• Visit the Avaya Support Centre Web site (http://support.avaya.com).
• Select Escalation Management > International Services.

Warranty
Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language as well as information regarding support for this product, while under warranty, is available through the Avaya Support Centre Web site http://support.avaya.com.

Ordering Information: Avaya Publications Center
Voice: +1 207 866 6701
Fax: +1 207 866 6737
Write: Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835 USA
Attention: Avaya Account Manager
Web: http://www.avayadocs.com
E-mail: totalware@gwsmail.com

Acknowledgment
This document was written by the ECAD Information Development group.
About Avaya IVR Designer

IVR Designer lets you create applications for voice response systems using a Windows NT 4.0, Window 2000, or Windows XP environment. IVR Designer has a point and click graphical user interface that makes it easy for developers to create voice response applications from a PC environment.

Introduction to Avaya IVR Designer

IVR Designer is designed to aid those who develop caller response applications for CONVERSANT® Version 8, and Release 9 systems, and for Avaya IR. This includes:

- Value added resellers (VARs) and independent software vendors (ISVs)
  These companies develop applications for customers with varied needs in terms of size and application. They produce both custom (used only by a single customer) and off-the-shelf (prepackaged) solutions.

- Software development departments within an end-user company
  Some firms have software development departments that write software for and maintain data and telecommunications systems. They are very focused on the needs of their company. They are usually have some professional development experience in the computer languages required by their company. Those languages could be anything from C and C++ to Cobol.

Avaya assumes that the primary users of IVR Designer are familiar with the following:

- Operation of the target voice response systems on which the voice response application will run.
- Their Windows operating system and common procedures, such as clicking, double-clicking, and dragging. If you are not familiar with these procedures, see the documentation for your operating system.
- Basic application development and programming techniques and concepts.

IVR Designer features

Major features of IVR Designer include:

- An easy-to-use, customizable graphical user interface (GUI)
About Avaya IVR Designer

- Standard Windows-type file operations
- A variety of tools to edit, diagnose, and debug applications
- Resource managers to create and edit global resources such as variables and phrases
- Extensive use of right-click menus
- Cut/Copy/Paste capabilities across call flows and applications
- The ability to use Secure Sockets Layer (SSL) technology for secure data transactions

  Note:
  This feature is available only with Avaya IR Release 1.2.1 and later. It is not available for use with CONVERSANT systems or earlier versions of Avaya IR.

- An Inline Code node that lets you insert VoiceXML code into your application "as is"
- Comprehensive language support
- Powerful Code Generation and Application Transfer tools
- An extensive online Help system

System requirements

The following are the minimum system requirements for IVR Designer:

- Pentium 133 MHz processor or equivalent (900 MHz processor is recommended)
- Microsoft® Windows 2000, Windows NT 4.0, or Windows XP
- 64 MB RAM (128 MB RAM is recommended)
- 65 MB free hard disk space (full install)
- VGA (256) color monitor

  Note:
  If you plan to record your own phrases and store them on the PC, you will need considerably more hard disk space. Storing applications on the PC will also require more hard disk space.

Optional hardware

The following hardware is not required, but can enhance your ability to use IVR Designer more effectively:

- Sound Blaster-compatible sound card capable of 16-bit monaural sampling at 8 KHz
- Microphone
Introduction to Avaya IVR Designer

- Speakers
- Microsoft-standard networking hardware for sharing resources and for transferring files to the target system. If you want to transfer files to the target system using networking, the target system must also be connected to the network.

Optional software

The following software is not required to use IVR Designer, but it can enhance your ability to use IVR Designer more effectively:

- Digital-audio editing software for editing audio files
- Microsoft-standard networking software and TCP/IP stack for sharing resources and for transferring files to the target system. If you want to transfer files to the target system using networking, the target system must have appropriate networking software installed.

Note:
Applications created using TAS are not compatible with MRCP servers.

System settings

For optimal performance, IVR Designer is designed to work with the following settings on your computer:

- 256 (8-bit) color
- 1024 x 768 monitor display (This setting may not be available on some portable computers.)
- Small fonts (system settings) – Do not use the Large Fonts setting in the Control Panel, Settings tab. IVR Designer does not display properly if large system fonts are used.

Installing Avaya IVR Designer on your PC

To install IVR Designer on your PC:

1. Insert the IVR Designer CD-ROM into the CD-ROM drive.
   - If your computer is configured for autoplay of CD-ROMs, the system displays the IVR Designer browser window.
   - If the browser window is not displayed within a few seconds, use Windows Explorer to locate the setup.exe file on the CD-ROM. Double-click setup.exe. The system displays the IVR Designer browser window.

   The system displays the IVR Designer Welcome window.
2. Click **Next**.
   The system displays the **IVR Designer Setup** window.

3. Click **Next**.
   The system displays the **Choose Destination Location** Window.

4. To select a destination other than the default (C:\Program Files\Avaya\Avaya IVR Designer), click the **Browse** button in the **Destination Folder** area and select the desired destination path.

5. Click **Next**.
   The system displays the **Setup Type** window.

6. Select the type of setup you prefer from the three options described in the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical</td>
<td>Installs all required IVR Designer files, all supported languages, and sample applications. Recommended for most users.</td>
</tr>
<tr>
<td>Compact</td>
<td>Installs basic IVR Designer files, but installs only the US English-female language package. Other languages must be installed separately.</td>
</tr>
<tr>
<td>Custom</td>
<td>Lets you select which IVR Designer options you want to install. For example, you may choose to install only one or two languages from all those available.</td>
</tr>
</tbody>
</table>

7. If you chose the **Custom** setup option, select the components you want to install.

8. The system displays the **Select Program Folder** window.
   Choose one of the following options:
   — Use the default. (To use the default, do nothing here. Proceed to Step10.)
   — Type a new name for the folder where you want the program files to reside.
   — Select from the list of Existing Folders.

9. Click **Next**.
   Avaya IVR Designer is installed on your PC according to the options you selected.
   When the installation process is nearly complete, the system displays the **Default Language Selection** window.

10. Select the language you want to install as the default language.
    By default, the **US English - Female** checkbox is checked. If you want US English - Female as the default language, do not check anything in this window.
Whatever you select here will be the default language for all new applications until you change the language preference in the Preference Editor.

11. Click Next.

The system displays the following message:

One of the benefits of Avaya IVR Designer is the ability to share resources between developers. To accomplish this the directory C:\Program Files\Avaya\Avaya IVR Designer\shared must be shared by windows. Click the YES button to launch a Windows Explorer screen and right click on the C:\Program Files\Avaya\Avaya IVR Designer\shared directory and choose "Sharing..." from the pop up menu. Sharing this directory can be performed at any time. For detailed information on sharing folders please refer to your Windows manuals.

12. To designate the Shared directory as a shared directory at this time, click Yes and follow the procedure for designating a directory as shared.

If you want to designate the Shared directory as a shared directory later, click No. For details about sharing directories, see the documentation for your Windows operating system.

The InstallShield Wizard displays the readme file for Avaya IVR Designer. Read it to note any changes that were made to the product after the documentation release.

13. When you have read the readme file, click Next.

The system displays the Setup Complete window, notifying you that IVR Designer is successfully installed on your PC.

14. To launch Avaya IVR Designer immediately, check the I would like to launch Avaya IVR Designer checkbox.

To launch Avaya IVR Designer later, clear the I would like to launch Avaya IVR Designer checkbox.

15. Click Finish.

---

Internationalization and globalization

Operating system and hardware considerations

IVR Designer supports in-language (Brazilian Spanish, French, German, Japanese, Korean, Portuguese, and Simplified Chinese) Microsoft Windows Operating Systems (Windows 2000, NT, and XP) and keyboards to develop IVR Designer applications.
About Avaya IVR Designer

IVR Designer is designed to operate with keyboard input using the US English keyboard character map settings. If you have mapped your keyboard to use the keyboard character map of another language, you may not get the appropriate input from your keyboard.

The following IVR Designer menu shortcut keys work on the appropriate in-language keyboard (IBM PC compatible):

- **Undo (Ctrl + Z)** – Reverse the last action or series of actions.
- **Redo (Ctrl + Y)** – Reverse any actions that were undone using the **Undo** command.
- **Cut (Ctrl + X)** – Remove the selected nodes and copy them to the clipboard for pasting.
- **Copy (Ctrl + C)** – Copy the selected nodes to the clipboard for pasting.
- **Paste (Ctrl + V)** – Paste all nodes currently on the clipboard to a selected open branch.
- All menu shortcut keys.

The recommended screen resolution of 1024 x 768 pixels does not work well with the default drivers of the in-language operating systems. For optimal screen settings, use the appropriate video driver for your display adapter.

Languages supported by IVR Designer

IVR Designer text fields in the Prompt Editor and Phrase Editor allow text to be entered and displayed in many languages. To see the list of available languages or to use a different language, select Tools > **Preference Editor** from the IVR Designer menu bar. The code generated for VoiceXML applications uses the little-endian Unicode-16 file format.

**Note:**
For VoiceXML, some fields allow entry of double-byte data. See Using double-byte data in VoiceXML applications on page 126 for additional information. For TAS applications, double-byte data is only allowed in the prompt and phrase editors.

The IVR Designer user interface has not yet been translated into other languages. Therefore, regardless of the language you select, the windows and dialog boxes in IVR Designer remain in US English. The exception to this rule is that, if you have selected another language as the default or primary language within your Windows operating system, Windows-standard text and dialog boxes will be in the language of the operating system.

Languages supported by the target voice response system

For TAS applications, IVR Designer provides support for all CONVERSANT V8, CONVERSANT R9, and Avaya IR languages, including all:

- Enhanced Basic Speech (EBS) languages and formats
- WholeWord speech recognition packages
• FlexWord™ speech recognition packages (FlexWord speech recognition is not supported on Avaya IR.)

For VoiceXML applications, the language selected for an application applies only to prompts that are spoken using Text-to-Speech. Also, the language selected for an application must be installed on the Text-to-Speech server.

You select the default language for IVR Designer at the time of installation. You can select only one language per application. You can change the default language at any time using the Preference Editor.

For information about selecting or changing the default language for an IVR Designer application, see Setting user preferences on page 21.

Checking for the SCI tools on the target system

The Service Creation Installation (SCI) tools can be used to transfer VoiceXML applications created using IVR Designer to a Avaya Interactive Response system. For details about using the SCI tools, see Transferring files from floppy disks to the target system on page 153.

Checking for the SCI tools on a Avaya IR system

To determine whether the SCI tools are installed on the Avaya IR system, at the UNIX prompt, enter `pkginfo sc`

If the SCI tools are installed on the Avaya Interactive Response, the system displays the following message:

IVR AVsc Service Creation Integration Package [Release number]

If the SCI tools are not installed on the Avaya IR, the system displays the following message:

UX:pkginfo: ERROR: information for sc was not found

The graphical user interface

The graphical user interface (GUI) is, for the most part, based upon standard Windows conventions and practices. This includes the use of clickable icons or buttons, many common keyboard shortcuts, and right-click menus. For more information about the user interface, see VoiceXML IVR Designer Interface on page 166.
The IVR Designer GUI has many customizable features, which are described throughout this help system. Of particular interest are the following:

- The ability to dock and undock toolbar palettes
- The option to view encapsulated call flows using either a **Multiple Document** interface, which has multiple windows, or a **Tabbed** Interface, which uses tabs to access call flow windows
- The ability to resize the **Globals Manager** and to dock and undock individual resource managers
- The ability to specify when the Node Inspector is displayed

When you first open a new Avaya IVR Designer application, the system displays a window similar to the following.

At the top of the window is the title bar. This area includes the name of the current application and the minimize, restore, and close window buttons.

Just below the title bar is the menu bar. The menus on the menu bar list the most common commands used in creating an application. For information about menus and menu options, see **Menus and menu options** on page 330.

Just below the menu bar are the palettes. The following palettes, each with its own set of icons, are available:
• The Toolbar
• The Node Palette

These palettes can be docked to or undocked from the main window, configured separately from the others, and customized. For information about palettes, see Palettes on page 336.

The white space below the palettes is where application call flows are created. It is called the design pad. This is where you do most of the work in creating an application. For information about creating an application, see Creating a voice response application on page 61.

At the bottom of the window is a status bar, where the system displays context-sensitive help or status messages.

---

**Getting started with IVR Designer**

IVR Designer offers powerful and flexible options for controlling your workspace. Some of these options include:

• Setting global preferences for using the workspace
• Using multiple nonmodal windows
• Customizing the palettes

You will want to set most of these options only once, but you can change any of them at any time.

For more information about configuring the IVR Designer workspace, see Configuring the VoiceXML IVR Designer desktop on page 173.

---

**Converting Voice@Work and IVR Designer applications**

Avaya IVR Designer can automatically convert TAS and VoiceXML applications that were created using an earlier release of IVR Designer or Voice@Work to the current release of IVR Designer. For TAS applications that were created with Voice@Work Release 1.x, 2.x, or 3.x, the application must first be converted to Release 4.x before it can be converted to Release 5.0 or later.

To convert an application that was created using an earlier release of IVR Designer or Voice@Work:

1. Select File > Open.
About Avaya IVR Designer

IVR Designer displays the **Open Avaya IVR Designer Application** window.

2. Select the application that you want to convert.

3. Select **OK**.

IVR Designer displays the following message:

   Application application name needs to be converted from version x.x to 5.1.

4. Do one of the following:
   - To convert the application, select **OK**.
   - To cancel the application conversion, select **Cancel**.

You are now ready to modify the application, if necessary, or save, generate, and transfer the application to the target platform.

---

Creating an application using Avaya IVR Designer

Use the following process to create an IVR Designer application:

1. Plan the application design.
2. Create the call flows and other application components.
3. Generate the source code files.
4. Transfer the application to the target system.
5. Assign the application on the target system.

For more information about creating an application, see Creating a voice response application on page 61.

For practice creating an application, see Creating a basic VoiceXML application on page 181.

---

Plan the application design

Before you create your application, you must have a clear idea of what you want to accomplish with it. This involves thorough planning to make sure you have thought of everything you want your application to do. You must plan for every contingency. For more information about planning an application, see Planning voice response applications on page 32.
Creating a flow chart or diagram can be helpful when mapping out what you want your application to do. Writing an outline and a script is another helpful technique.

Create the call flows and other application components

Once you have a clearly defined design, you can create the call flows and other components required to make the design work. You do this by dragging nodes onto the design pad (see Working with nodes on page 61 for the procedure). You can set attributes for nodes and edit them using the Node Inspector tool. For more information about creating an application, see Creating a voice response application on page 61.

IVR Designer components

In addition to the nodes offered on the Node Palette, IVR Designer offers a variety of other components you can use in building a call flow. The following table describes the major components, called global resources, offered in IVR Designer.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrases and phrase tables</td>
<td>Phrases are bits of spoken (or written) speech, often used to instruct customers or prompt them for pieces of information. Phrases are grouped together and contained in phrase tables. Phrase tables can be created within IVR Designer application or imported from other applications or from text files.</td>
</tr>
<tr>
<td>Variables</td>
<td>Variables are application-specific information holders. Variables may be used to collect and hold data, act as counters, or make use of system performance information.</td>
</tr>
<tr>
<td>Prompts</td>
<td>Prompts are used to organize phrases, variables, and controls into logical units for use with the Interactive Voice Response (IVR) nodes. Prompts are used to instruct callers, to get callers to select choices, and to obtain information from callers.</td>
</tr>
<tr>
<td>VXML grammars</td>
<td>VoiceXML grammars are used to provide a highly accurate recognition of spoken input.</td>
</tr>
</tbody>
</table>

IVR Designer tools

IVR Designer also offers a variety of development tools to help you edit, test, and debug your application. The following table describes the development tools offered in IVR Designer.
### Tool Description

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Flow Inventory</td>
<td>The Call Flow Inventory tool lets you see:</td>
</tr>
<tr>
<td></td>
<td>• What call flows exist for your application</td>
</tr>
<tr>
<td></td>
<td>• Which call flows use other subordinate call flows (Uses tab)</td>
</tr>
<tr>
<td></td>
<td>• Which call flows are used by or are subordinate to other call flows (Used By tab)</td>
</tr>
<tr>
<td>Code Generation/Application Transfer</td>
<td>The Code Generation tool generates the code necessary to transfer and install the application onto the target system.</td>
</tr>
<tr>
<td></td>
<td>The Application Transfer tool transfers the application files to the target system.</td>
</tr>
<tr>
<td>Node Inspector</td>
<td>The Node Inspector displays different information and editing options for each node, depending on the node’s predetermined attributes. Using the Node Inspector, you can edit node attributes easily and quickly.</td>
</tr>
<tr>
<td>Preference Editor</td>
<td>The Preference Editor sets application-wide preferences for using IVR Designer.</td>
</tr>
<tr>
<td>Verify Design</td>
<td>The Verify Design tool checks your application design for possible errors or omissions. When you open the Verify Design tool, it automatically searches through each call flow of your application. The results are displayed in the Verification Results window.</td>
</tr>
</tbody>
</table>

### Generate the source code

After you have created, tested, and debugged your application, you must convert it to a form that the target system can understand and use.

With the introduction of Avaya IVR Designer Release 5.3, you now have the option of selecting the output document type. You can now generate VoiceXML 2.0 non-compliant documents by clicking the **VoiceXML 2.0 Non-compliant** checkbox. You can also generate VoiceXML 2.0 compliant documents by clicking the **VoiceXML 2.0 compliant** checkbox.

**Note:**
VoiceXML 2.0 complaint documents are compatible only with an Avaya Interactive Response Release 2.0. VoiceXML 2.0 complaint documents can be made compatible with an Avaya IR Release 1.3 system, by installing the AVvoicxml2-0 package. Voice XML 2.0 compliant documents are not compatible with versions earlier than Avaya IR R1.3. VoiceXML 2.0 non-compliant documents are incompatible with an Avaya IR 2.0 system.

Use the Code Generation/Application Transfer tool to generate the code for the target system. For more information about generating the application code, see Generating a VoiceXML document (application on page 146).
Transfer the application to the target system

When you have finished generating the code for the target system, you must transfer the generated source code files to the target system.

If the PC with the Avaya IVR Designer application has network connectivity to the target voice response system, you can transfer the files over the network. If the PC does not have network connectivity to the target system, transfer the application to floppy disks, then install the application on the target system.

Use the Code Generation/Application Transfer tool to transfer the files to the target system. For more information about transferring applications, see Transferring the application on page 150.

Install the application on the target system

Before you can use an application that you have transferred to the target system, you must assign the application to channels or numbers. If the PC with the Avaya IVR Designer application has network connectivity to the target voice response system, the application can be assigned from Avaya IVR Designer. For more information about assigning a VoiceXML application, see Assigning an application on the target system on page 162.

Setting user preferences

The Preference Editor lets you set application and tool-wide preferences for using IVR Designer. The Preference Editor has tabs (sets of options), each of which controls a different set of preference options.

Opening and using the Preference Editor

To open the Preference Editor, select Preference Editor from the Tools menu. IVR Designer displays the Preference Editor window.

To set your preferences, apply the changes, and close the Preference Editor window, click OK.

To cancel any changes you have made and revert to the previous settings, click the Cancel button.

To apply any changes you have made without closing the window, click the Apply button.
About Avaya IVR Designer

Note:
The Apply button is active only if you have made changes in the Preference Editor.

The Application Parameters tab

The Application Parameters tab contains options for setting various parameters. The following table summarizes the options available on the Application Parameters tab.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Application Directory</td>
<td>This parameter lets you define the full pathname for the directory where applications are to be saved by default. You must include the drive letter. Note: When you reset the default pathname, the change does not become effective until you exit and restart IVR Designer. If you do not know the pathname, click the Browse button to search for the directory you want. Note: This default pathname is used by the New, Open, and Save commands on the File menu when you first open IVR Designer. If you select a different pathname, IVR Designer remembers and uses that pathname.</td>
</tr>
<tr>
<td>Application-Wide Language Selection</td>
<td>This parameter lets you select from the drop-down menu the language you want to use in your VoiceXML applications. For VoiceXML applications, the selected language applies only to prompts that are spoken using Text-to-Speech. If you want to use the language as the default for future applications, click the Use this as the default language checkbox. If you do not check this box, IVR Designer returns to the previously specified default language when you start a new application. The default is the language you selected as the default language when you first installed IVR Designer on your PC or whatever other language you have since set as the default language using the Preference Editor. Note: The language that you select for VoiceXML applications must be installed on the Text-to-Speech server.</td>
</tr>
</tbody>
</table>

The Environment tab

The Environment tab contains options for setting your working environment in IVR Designer. The following table describes the options available on the Environment tab.
<table>
<thead>
<tr>
<th>Option group</th>
<th>Option (and type)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opening/Saving</td>
<td>Save desktop upon exit (checkbox)</td>
<td>When checked, saves your design pad (desktop) options so they are used whenever you reopen IVR Designer.</td>
</tr>
<tr>
<td></td>
<td>Save desktop when saving application (checkbox)</td>
<td>When checked, saves your design pad (desktop) options whenever you save an application.</td>
</tr>
<tr>
<td></td>
<td>Automatically open last application (checkbox)</td>
<td>When checked, upon starting IVR Designer, automatically opens whatever application you were working on the last time you had IVR Designer open.</td>
</tr>
<tr>
<td></td>
<td>Automatically save application (checkbox and drop-down menu)</td>
<td>When checked, saves your application automatically at the time interval you specify. Offers a range of 1 through 179 minutes. Setting this number to zero has the same result as clearing the box.</td>
</tr>
<tr>
<td>Spell Check</td>
<td>Automatically Spell Check Phrases on Exit (checkbox)</td>
<td>When checked, uses the Microsoft Word Spell Checker to spell check phrase text whenever you exit the Phrase Editor. IVR Designer uses its own interface in accessing the spell checker tool.</td>
</tr>
<tr>
<td></td>
<td>Language for Spell Checker (drop-down menu)</td>
<td>Defaults to the designated language for the Microsoft Word Spell Checker.</td>
</tr>
<tr>
<td>Call Flow Editor</td>
<td>Automatically name nodes (checkbox)</td>
<td>When checked, suggests a name for each new node whenever you drag an icon onto the design pad. You may then either accept the suggested name or type another name.</td>
</tr>
<tr>
<td></td>
<td>Multiple Document Interface (radio button)</td>
<td>When selected, provides a different (nonmodal) window for each call flow in the application. This lets you see more than one call flow at a time.</td>
</tr>
<tr>
<td></td>
<td>Tabbed Interface (radio button)</td>
<td>When selected, lets you select call flows by clicking the call flow name tabs at the bottom of the design pad window. Each call flow, when selected, occupies the entire design pad window space.</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td>This feature works only if you have both Microsoft Word and the Word Spell Checker installed on your PC.</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td>You must restart IVR Designer before this option takes effect.</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td>You must restart IVR Designer before this option takes effect.</td>
</tr>
</tbody>
</table>
The Printing tab

The **Printing** tab contains options for printing reports in IVR Designer.

All the options on the Printing tab, except the **Output Format To Printer** option, toggle printing options on or off. The options work as follows:

- When checked, the option is available for printing if it exists in the application. If nothing exists for that option, it does not appear in the **Print Reports** window, even if you have checked it here.

- When unchecked, the designated option is not available for printing, even if it exists in the application.

The **Output Format To Printer** checkbox determines the default destination for print reports as follows:

- When checked, the default output format is to the Default Printer.

- When unchecked, the default output format is to a document (*.doc) file. The document file is saved in a folder named **print**, which is located below the folder in which the application is saved.

When you select **File>Print...** or click the **Print** icon from IVR Designer, the system displays the Print Reports window, which is similar to the Printing tab. You can use the Print Reports window to change printing options before you print a report.

About Help

Avaya IVR Designer has two separate Help files. One Help file contains information about TAS applications. The other Help file contains information about VoiceXML applications.

When you have an application open, you can access only the Help file that corresponds to the type of application you have open. For example, if you have a VoiceXML application open and you press **F1** or select **Contents...** from the **Help** menu, the system displays the Avaya IVR Designer Help file for VoiceXML Applications.

**Opening a Help file from the Start menu**

You can open either or both Help files from the Windows Start menu. To open an Avaya IVR Designer Help file from the Windows Start menu, go to **Programs>Avaya>Avaya IVR Designer>Help**, and select the appropriate help file.
About Help

Printing the Help file

You can print individual Help topics from either Help file, but there is no easy way to print the entire Help file. If you want to print the entire Help file, use the PDF version of the Help file. You must have Adobe Acrobat installed on your PC to open the PDF version of the Help file.

A PDF version of each Help file is available on the Avaya IVR Designer CD in the docs folder, and from the Product Documentation area of the following web site: http://support.avaya.com/

About the Help window

The following figure shows the Help window.

The Help window has a toolbar, a navigation pane, and a topic pane.

The toolbar

The following table describes the toolbar buttons:
### About Avaya IVR Designer

<table>
<thead>
<tr>
<th>Button name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide/Show</td>
<td>If the navigation pane is open, click the <strong>Hide</strong> button to close the pane. If the navigation pane is closed, click the <strong>Show</strong> button to open the pane.</td>
</tr>
<tr>
<td>Back</td>
<td>Click the <strong>Back</strong> button to open the previously viewed Help topic in the topic pane.</td>
</tr>
<tr>
<td>Print</td>
<td>Click the <strong>Print</strong> button to open a dialog box from which you can choose whether to print the selected topic (the topic currently displayed in the topic pane), or the selected topic and all subtopics.</td>
</tr>
</tbody>
</table>
| Options     | Click the **Options** button to open a menu that contains the following items:  
  - **Hide Tabs/Show Tabs** – performs the same function as the Hide/Show button.  
  - **Back** – performs the same function as the Back button.  
  - **Stop** – stops a web page from opening.  
  - **Refresh** – refreshes the topic that is displayed in the topic pane.  
  - **Internet Options** – opens a dialog box from which you can set various options for Internet Explorer.  
  - **Print...** – performs the same function as the Print button.  
  - **Search Highlight On/Off** – turns on or off the Search Highlight function. The Search Highlight function highlights the word or words that you search for using the Search tab. |

### The navigation pane

The following table describes the tabs in the navigation pane:

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>The <strong>Contents</strong> tab displays an expandable list of Help topics. Items in the list that can be expanded have a book icon and a plus sign to the left of the item. To expand an item, click the plus sign, or double-click the item. To select a Help topic, click an item in the list. The selected Help topic is displayed in the topic pane.</td>
</tr>
<tr>
<td>Index</td>
<td>The <strong>Index</strong> tab displays a list of Help topics arranged in alphabetic order. To select a Help topic, click an item in the list, then click the <strong>Display</strong> button. You can also type a word into the text box above the list and press <strong>Enter</strong> or click the <strong>Display</strong> button when an index entry is highlighted. The selected Help topic is displayed in the topic pane.</td>
</tr>
<tr>
<td>Search</td>
<td>Using the <strong>Search</strong> tab, you can search for one or more words in the Help topics. To search for a word, type the word to search for in the text box and click the <strong>List Topics</strong> button. A list of Help topics that contain the search word is displayed. To select a Help topic, double-click an item in the list. The selected Help topic is displayed in the topic pane.</td>
</tr>
</tbody>
</table>
The topic pane

The topic pane displays the Help topic that you selected in the navigation pane. You can move from the currently displayed topic to the next or previous Help topic by clicking the up or down arrow at the top of the topic pane.
Voice application basics

Voice application basics provides an overview of basic voice response application concepts, and of how voice applications are used.

What is a voice response system?

A voice response system consists of hardware and software that provides either full or partial automation of telephone transactions that would otherwise be performed by an operator or attendant, an employee, or a call center agent.

What is a voice response application?

An application is a set of instructions written for the voice response system that tells it how to carry out an automated transaction. Applications define the call flow and determine what callers hear and how callers respond to the system.

This section includes the following topics:

Voice response application uses..................................................... 28
Voice response system versus an attendant................................... 30
Defining successful voice response applications............................. 30

Voice response application uses

Voice response applications can be used in a wide variety of situations. The following examples can help you determine some possibilities for the best use of voice response applications for your business.

Note:  
With the introduction of Avaya Interactive Response R1.3, you can now use Secure Sockets Layer (SSL) technology for those parts of your VoiceXML applications that require secure data transmission (such as when you want to collect personal, credit card, bank account, or other types of private and/or sensitive information from a caller). For more information, see Using SSL in VoiceXML applications. Be aware that SSL is not available for use with CONVERSANT systems or earlier versions of Avaya IR. With the introduction of
Avaya Interactive Response Release 2.0, you can now use Secure File Protocol (SFTP) service with the Secure Shell (SSH) protocol to transfer the application in a secure manner.

**Voice response for repetitive information requests**

Repetitive information requests that can be fulfilled with a voice response application include:

- Interest rates, foreign currency exchange rates, and loan rates
- Real estate information on a property with a specific identifier such as a number
- Customer service information such as updates on product maintenance issues and their resolutions
- Status of automobile repairs being performed and their estimated costs
- Registration forms, entry blanks, or tax forms
- Conference/Tradeshow information such as list of speakers and schedules
- Air travel information such as departure and arrival times
- Financial statements company shareholders

Other examples include park or recreation locations, branch office and store locations, restaurant locations, bus schedules and rates, theater schedule and rates, concert information, and sporting event times and results.

**Voice response for order processing**

You can design a voice response application to accept touchtone input or spoken input, so that callers can use the system to place orders, confirm orders, or check order status. Ordering processing includes:

- Merchandise orders, restocking – A manufacturer calls a supplier to request spare parts. A confirmation of the order can be faxed to the manufacturer.
- Hotel reservations – Callers can register and pay by telephone, then receive a confirmation fax including a map and directions.
- Travel reservations – Callers can reserve a seat and pay over the telephone. Confirmation can be sent by fax.
- Benefits registration – Callers sign up for a combination of benefits, and can request confirmation by fax.
- Brokerage services – Callers can buy and sell stocks and bonds, receiving a fax confirmation of the transaction.
Voice application basics

- Talent agency – Callers can request information on actors with particular talents or physical attributes. Information can be spoken out, and photographs and resumes can be faxed to the caller.

Voice response for information gathering

A voice response system can be used to gather information from those who call a designated number. Information gathering includes:

- Consumer or political surveys
- Customer feedback and complaints
- Employee feedback

Voice response system versus an attendant

Do not eliminate all attendants. Even well-designed automated transactions cannot serve the needs of every caller. Callers may want and need to speak to a person, so it is important to provide some attendants that are familiar with your business policies and procedures.

Defining successful voice response applications

Consider multiple criteria when defining success in your voice response applications. You may have to measure some aspects of how you currently do business, and decide on one or more specific areas to target for improvement. Think about why you are interested in automating some of your business interactions. To save time? To save money? To better serve your customers? To free personnel for more challenging tasks? Take your most important reasons for automation, and determine measurable objectives for automation.

Defining measurable objectives

For example, if you want to use voice response to better serve your customers, you might choose as an objective reducing call holding times. A measurable objective might be to reduce call holding times by 50%. You can also figure out the yearly savings expected from reduced holding times, and factor this into your plan.

Here are some examples of other types of measurable objectives that you may want to achieve through the design of your application:

- Reduce your cost per call by 30%
- Increase call completion rate by 45%
- Reduce the percentage of callers who get a busy signal by 50%
• Increase customer satisfaction with the voice response system by 25%

Remember to estimate the rate of savings for each objective.

Determining your true objectives, as well as how to measure them, will allow you to judge the effectiveness of your voice response applications.
Planning a voice response application

Planning a voice response application provides general guidelines to apply when you plan the design of your voice response applications.

This section includes the following topics:

Use human factors and usability engineering resources ...................... 32
Use prerecorded speech....................................................................... 33
Offer a user guide ........................................................................... 33
Offer caller training.......................................................................... 34
Provide an attendant.......................................................................... 34
Plan for disabled callers...................................................................... 34
Diagram your application design......................................................... 35

Use human factors and usability engineering resources

Human factors and usability consultants are experts in the design and testing of interfaces between people and computers. Some consultants specialize in telephone-based user interfaces. If your company does not have people trained in usability or human factors, independent consultants are available to assist you in application design, testing, and deployment.

Contracting professional design and usability testing services can be cost effective, and can save you time and frustration. Customer satisfaction can also be increased, since you will be more likely to offer your customers a well-designed, extensively tested application.

Avaya offers human factors design consultation services on a Professional Services basis. For more information, contact your Avaya representative. For a directory of individuals and companies offering independent consulting in human factors and usability, contact the Human Factors and Ergonomics Society.
Use prerecorded speech

Prerecorded prompts and announcements spoken by a professional speaker (such as a voice-over artist or a disc jockey) can enhance the quality of your application. It is usually appropriate to use the same recorded voice throughout the application. An exception is when you have a good reason to try to distinguish between different parts of the application. For instance, you may want to have help speech recorded by a male voice and the remainder of your prompts and announcements in a female voice. This consistency may help your callers to know their location within an application.

Avaya offers professional custom phrase recording services. If you are interested in purchasing custom phrases, contact your Avaya representative.

If you choose to contract a professional speaker on your own, consider the following guidelines:

- Record the speaker's voice to ensure that you like the way it sounds. You may want to record several different speakers to compare the quality of their recorded voices.
- Record the speaker's voice at a level that does not exceed -11 dBm0.
- Make sure the speaker can maintain a constant speaking rhythm and intonation throughout the recording session.
- Make sure the speaker can maintain a constant, acceptable level of volume and distance from the telephone or microphone while recording.
- Make sure the speaker's pronunciation is clear and that words are not over-enunciated.
- Make sure the recording environment is as quiet as possible. A room with carpeted walls and floor is usually sufficient.
- Prepare the phrases for the speaker in advance of the recording session.

Offer a user guide

If you know your calling population, you can take advantage of this knowledge and provide a user guide. User guides can be very simple, providing just enough information so that callers will not be surprised by the automated transaction. More detailed user guides can include all the prompts and options, so callers can prepare answers ahead of time and become familiar with the application.

The nature of your application and calling population will determine whether or not a user guide is appropriate. User guides are most useful when you know and can easily reach the members of your calling population. Examples of known caller populations include:

- Subscribers to your service
Planning a voice response application

- Customers to whom you send bills
- Students, teachers, and parents of the school you serve
- Employees in your company

If you distribute a user guide and expect that most callers will use it, you may be able to provide shorter and more concise prompts than if callers do not have a guide. Be aware, however, that shorter prompts may adversely affect callers who do not use the guide.

Offer caller training

If you know that a particular group of callers will use the application extensively, consider offering training classes. Training is most appropriate when you have a limited calling population or a potentially complex or unusual application. The most useful training classes allow callers to call in to the application and use it.

Provide an attendant

There will always be callers who need extra assistance, especially those calling for the first time. Provide a person who can handle callers having trouble with the voice response applications. As a general rule, allow callers to try to enter information only two or three times. Then, if they are not successful, transfer the caller to an attendant. Allow callers to speak with an attendant by using a key press (like zero), or by speaking a key word (like attendant, operator, or help).

Plan for disabled callers

Applications should be written to conform with Section 255 of the Telecommunications Act of 1996, which mandates accessibility by callers with disabilities. Applications used by the federal government must meet the accessibility requirements of Section 508 of the Rehabilitation Act of 1998.

In part, this means that an application should be usable by callers:

- Without vision
- Without hearing
- Without speech
- With limited cognitive skills

Often it is a good idea to dedicate specific telephone numbers (corresponding to telephony ports that run the appropriate applications) for disabled callers to use.
Diagram your application design

When planning your application, create a diagram of how you want the application to appear to callers. A diagram is helpful to depict the structure of the application. The diagram also allows you to plan what to do when errors occur, an aspect of applications that is frequently overlooked. If the same person or team is not the application designer and developer, a thorough diagram is a very valuable communication tool. Even when one person or team is both designing and developing the application, the diagram helps to separate the tasks, so that each task receives adequate attention.

Flowcharts and outlines are two ways to diagram the application. If you already have a way to communicate the structure of the application that works for you, use it. Whatever method you choose, plan your application carefully before implementing it.

Be sure to list all prompts and announcements to be played to callers. Make sure you show what should happen each time you expect caller input. Include what should happen when callers give correct (valid) input.

If you need to use secure data transmission, be sure also to indicate on your design where/when you want to use SSL. For more information, see Using SSL in VoiceXML applications.

Note:

This feature is available with Avaya IR Release 1.2.1 and later. It is not available for use with CONVERSANT systems or earlier versions of Avaya IR.

Error cases are also important. What happens if callers give incorrect (invalid) input? How many times will you re-prompt after hearing incorrect input? Will the call be transferred to an attendant? What if callers do not respond? How many times will you re-prompt after hearing no response? A good flowchart or outline will cover all possible cases.

Use flowcharts

Flowcharts are more formal and specialized than outlines, but they are a good way to show the application in a diagram. Flowcharts use a few simple symbols to represent prompts, announcements, and decision points within the application. They use arrows to show the path from one symbol to another. Identify the symbols you use so that people can easily understand them.
Planning a voice response application

Use outlines

Another way to diagram the application is an outline, as shown below. Outlines are easier than flowcharts for some people to understand because they use only words, not symbols.

Sample outline

Caller dials in.

Prompted with: "You have reached High Technologies, Incorporated."
1: "Please say the first and last names of the person you want."
2: (listen for four seconds; listen for anything on Wordlist 1, that contains all names).

If there is no caller response or silence:
Prompt with: "Please repeat the name."
Go to 2.

If the caller response matches the word "help:"
Prompt with: "You can reach anyone at our location by saying their first and last names. If you need further assistance, say 'operator' when asked for a name."
Go to 1.

If the caller response matches the word "operator:"
Prompt with: "Transferring."
Transfer the call to the operator's extension.

Finished.

If the caller response matches a name on the wordlist:
"Calling {first name, last name.} To cancel, say 'no'."
(listen for two seconds; listen for "yes" or "no").

If match the word "no:"
Prompt with "Call canceled."
Go to 1.

If match the word "yes" or nothing (no caller response or silence):
Prompt with: "Transferring."
Transfer the call to the named person's extension.

Finished.
Designing a voice response application

*Designing a voice response application* provides general guidelines for designing voice response applications.

This section includes the following topics:

Application usability ................................................................. 37
Prompts ......................................................................................... 40
Announcements ........................................................................... 42
Menus .......................................................................................... 43
Digit input ...................................................................................... 45
Yes/no questions .......................................................................... 47
Pace the application ...................................................................... 48
Application errors ......................................................................... 49
Touch-tone and speech recognition ............................................. 50
Dial through and barge-in ............................................................ 56
Text-to-Speech in applications .................................................... 57

**Application usability**

In order to design the most successful application possible, it is a good idea to research some basic design principles. Human factors experts call these usability principles, since they are critical to designing a usable application.

Because of experiences with poorly designed applications, some callers are wary of all voice response systems. Be careful that your application does not alienate your customers. Your goal is to make an automated transaction at least as attractive and efficient as interacting with an attendant.
Know your callers

Choices you make in the application design will affect those who the application will serve. Find out as much as you can about the people who will call the system, and use the information when designing your applications.

Relevant information to gather includes the following:

<table>
<thead>
<tr>
<th>Information</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who are your callers?</td>
<td>Your callers have personal attributes that affect how they interact with the application you are designing.</td>
</tr>
<tr>
<td></td>
<td>What language do they speak? If you expect that many callers will have an accent that is different from what the speech recognition package you are using expects, consider offering a bilingual service.</td>
</tr>
<tr>
<td></td>
<td>How old are they? If you expect that a high percentage of callers may be children, you should use simple language and very short menus. Also, since speech recognition works best with deeper voices, for children you may want to use only touchtone input. If you expect that many of your callers are over the age of 65, you might want to allow them more time to respond. Make sure to reduce confusion by making the transaction as simple as possible.</td>
</tr>
<tr>
<td></td>
<td>Are they employees of your company? You may want a script designed for in-house use to appear and perform differently than one designed for your customers. For example, if you provide training classes or user guides, you can probably use shorter and more concise prompts. Jargon and technical terms may be appropriate for employees if you can assume that they know the terms.</td>
</tr>
<tr>
<td></td>
<td>Note any other personal attributes that could affect how callers will interact with your application, such as regional slang, accents, or education level.</td>
</tr>
<tr>
<td>How often do they call?</td>
<td>Some applications are meant to be used only one time, while others can be used more often. Callers might access some applications several times a day.</td>
</tr>
<tr>
<td></td>
<td>If you expect callers to use the application rarely, your prompts and announcements may need to contain explicit information and instructions. Prompts for applications used very often can be made extremely short, since most callers will likely be experienced. If you expect your callers to be mixed in their experience levels, make it possible for the more experienced callers to dial through or barge in during prompts.</td>
</tr>
<tr>
<td>How well do they know the subject matter?</td>
<td>If you expect that most of your callers will be unfamiliar with the subject matter (for example, if you are describing a new service or product), you should take special care in structuring and presenting the information so that your audience can understand it.</td>
</tr>
</tbody>
</table>
How will they feel about responding to requests for sensitive information?

If you are asking your callers to provide sensitive information, such as bank account numbers, credit card information, and the like, many callers are shy about providing this kind of information over the telephone.

If you want callers to provide this kind of information, be sure to use SSL for these parts of your applications—and be sure to let your callers know you are using secure connections for the transmission of this information.

**Note:**
The SSL option is available with Avaya IR Release 1.2.1 or later. It is not available for use with CONVERSANT systems or earlier versions of Avaya IR.

---

**Use simple and natural dialog**

Learning as much as you can about callers will allow you to determine what is "simple and natural" for them. Design uncomplicated, straightforward applications that use terms familiar to the callers.

**Minimize demands on the caller’s memory**

Psychological research has shown that short-term memory can hold no more than five to nine separate pieces of information at one time. For each menu item you present to callers, two different pieces of information must be remembered: the option ("For loan rate information...") and the action required to choose the option ("press 2."). Menus should offer no more than four or five items at once, since callers cannot remember more items.

The fewer items callers have to remember, the more likely it is that they will remember them. Try to make the interaction easy for the callers. Keep menus short, and avoid wordy prompts and announcements.

**Be consistent**

Consistency in the structure of menu choices and in the presentation of information helps prevent callers from becoming confused. By using consistent language and requesting consistent action of callers, you simplify the interaction and reduce demands on memory.

**Provide feedback**

Whenever callers press touch tones or speak in response to a prompt, let them know how the response was interpreted. This clarifies the transaction for callers. For instance, if callers press zero for an attendant, the application could say "Please hold for an attendant." Although the application does not explicitly state that it heard the caller press zero, the interpretation of the key press is implicit in the announcement that follows.
Designing a voice response application

Provide easy exits
To prevent callers from getting trapped in an application script, tell them how to reach an attendant, end a transaction, or return to the main menu. You can give this information at the beginning of the transaction. Provide these types of escapes within your application so that callers can always return to a place they know or can obtain help as necessary. This gives your callers more control over the interaction, and makes them feel as if they are not powerless in the face of automation.

Offer shortcuts
If you expect some callers to call often enough to become experts, provide ways to shorten their interaction with the application. For example, allow them to respond before the end of a prompt so that they can move through the interaction at a pace that is comfortable for them. You may want to consider providing separate scripts (or separate branches of a single script) for novice and expert callers.

Prevent errors
By following the above principles, you can prevent errors from happening. Well designed applications take into account the strengths and weakness of both the callers and the technologies.

Prompts

The prompts used in your application are one of the few forms of system interaction with callers. For this reason, it is very important that you design the application prompts to facilitate a successful interaction. Use the following guidelines to design the prompts for your voice response applications.

Prompt length
Make prompts as short as possible while remaining polite and informative. Keep the prompts brief, but not abrupt. The more quickly callers can do what they called to do, the more satisfied they will be. For example:

Instead of prompting...
To listen to information on travel to California, say 1.
If you are calling to register for classes, press 3.

prompt...
For California travel information, say 1.
To register for classes, press 3.

There are some trade-offs when considering prompt length. If you expect to have a high proportion of one-time or first-time callers, you might want to provide more information in prompts and make the prompts longer. Longer prompts are acceptable if you think your callers will need the extra information. Include only helpful information and express it in a short, direct manner.
Use longer, more informative prompts if you are asking callers to enter information in a way that may be unfamiliar. For instance, consider a caller entering a date. Most people think of months by name. For a touchtone application, callers are asked to press the one-digit or two-digit number for the month. For example, the month of December would be 12. With WholeWord speech recognition, the month of December is represented by the words "one, two" which callers may say as "twelve." Both of these examples require that callers think of the information in a way that is unusual, and so prompts should be clear enough for callers to understand. These longer prompts should include examples, such as:

"Next, enter the month listed on your bill. Say two numbers, such as `one, two' for December, or `zero, five' for May."

**How to word prompts**

Use the following guidelines when determining the wording of prompts:

- **Present the option before the action.**
  
  In menu prompts, always list the option first, and the action second. For example:

  **Instead of prompting...**
  Say 2 to receive the information by fax.

  **Prompt...**
  To receive the information by fax, say 2.

  When the option is listed first, callers need to listen only to the one action that pertains to their choice. When the action is listed first, they must remember it until they hear the option and decide if it is the one they want. Listing the option first will help both experienced and new callers.

- **Use the same structure for each option-action pair in a menu prompt.** This helps the caller to know what to expect. For example:

  **Instead of prompting...**
  To hear sales information, press 1.
  For services, press 2.
  If you would like to speak to an attendant, press 0.

  **Prompt...**
  For sales, press 1.
  For services, press 2.
  For an attendant, press 0.

- **Make each menu choice clearly different from other menu choices.**
  
  Be sure the menu choices are different from each other, so callers can decide which option they want. If callers are confused about which option they want, consider rewording the choices.

- **Avoid jargon and acronyms.**
  
  Assume that callers do not use the same technical words you do. Avoid acronyms and abbreviations that may confuse your callers. Speak out the complete words. Use words that your typical callers will understand. Use technical terms only if your audience is a small group of people who are trained in your business.
Announcements

As with prompts, it is very important that you design the application announcements to facilitate a successful interaction with your callers. Announcements are used to inform the caller. Be selective when deciding whether or not to include announcements in your design, and if you decide to include announcements, how many.

Feedback announcements

Feedback announcements should be short and clear. If you want to include "please" and "thank you," include them only occasionally so that callers can complete the call as quickly as possible.

Avoid referring to the system as *I* or *we*. If callers think the system is a person, they may use more words to make their answers more friendly, thus making speech recognition more difficult. For example:

Instead of prompting... prompt...
Please wait while I locate your records. Please wait while your records are located.

If a request will take more than two or three seconds to fill, assure callers that work is being done. For example:

"Looking for your records. Please wait."
or
"Please wait while your information is located."

Confirmation announcements

There are times where you should read the entry of the caller back, and allow the caller to confirm it. Other times, a feedback announcement (such as "Please wait while your information is located") is more appropriate than a confirmation announcement.

When you need confirmation

In cases where a recognition error or caller touchtone error would have a significant impact (like credit card numbers or call destinations), you should give callers an opportunity to confirm the entry. To do this, repeat the entry and prompt callers to say "yes" or "no" to confirm, as in the following examples:

- With speech recognition:
  "What is your customer number?"
  < 432886 >
  "Customer Number 432886. Is this correct?"
  {pause}
  "Please say yes or no, now."
With touch tones:

"Please enter your customer number."
< 432886 >
"432886. If this is correct, press 1. If not, press 2"

With IVR Designer, use a separate **Prompt and Collect** node to do the confirmation.

**When you do not need confirmation**

In cases where a recognition error or a caller touchtone error would not cause a big problem (that is, if it is easy for callers to return to the point in the application where they really wanted to go), you may choose not to allow callers to confirm an entry. Also skipping the confirmation of nonessential entries can speed the call.

Even if you do not confirm an entry, it is important to let callers know how the recognizer interpreted the input. To do so, include context-relevant information in the next announcement you play. For example:

"What month?"
< August >
"Artists performing at the Palace Theater for August include..."
Designing a voice response application

For Lugano Falls, press 3
For Monroe Bluff, press 4"

In certain cases, it might be acceptable to present more than five menu choices. If all callers are trained in using your voice response system, or if they use the system frequently, having longer menus may save time, as long as you allow callers to dial through or barge-in to interrupt the menu prompt.

Menu choice sequence

To decrease the call length, present the most likely menu choice first, the second most likely second, and so on. If you do not know the preferred order, make a guess and adjust if necessary during application testing.

Numbered menu options

When presenting menu options that a caller can choose by number, present them in numerical order. Avoid skipping numbers.

If you change the sequence of your menu to present the more likely choices sooner, be sure to change the numbers of the menu prompts so that the caller can choose the first option by selecting 1, the second by selecting 2, and so on.

Keep in mind that experienced callers, and those with user guides, probably do not listen to all menu prompts. When you change one or more menus in your application, consider adding an announcement during the greeting, such as "The menu has changed; please listen carefully." After some time, you can remove this message.

Subdivided menu options

If your menu tree is complex, or you think that callers will want to get information from many different menus, give callers a choice to move to a different menu. You could add a choice to allow callers to go back to the previous menu, or a choice that would allow callers to go to the top of the menu tree. For example:

"For parks with camping facilities, press 1  
For other parks, press 2"

< 1 >

"For Stony Ridge, press 1  
For Cantwell Cliffs, press 2  
For Old Man's Cave, press 3  
For the previous menu, press 4"

< 2 >

"Cantwell Cliffs offers 15 tent camping sites, each with a water spigot. Cooking fires are permitted. Toilet and shower facilities are available. Reservations are accepted from April first through November first."
"To make a reservation, press 1
For directions, press 2
For the previous menu, press 3
To start at the beginning, press 4"

Although the menu should automatically repeat if the caller does not respond, you could include an option like "To repeat the menu, press 9."

**Digit input**

Use the following guidelines as you design your applications that use digit input.

**Constant-length digit sequences**

A constant-length digit sequence is a series that always contains the same number of digits, for example a United States social security number has nine digits.

The system recognizes constant-length sequences more accurately than variable-length sequences. Whenever possible, specify the maximum length of the variable-digit sequence to be recognized, to increase recognition accuracy.

**Variable-length digit sequences**

A variable-length digit sequence can contain different numbers of digits, for example a house number in a street address.

If your application must accept a variable-length digit sequence, you can increase recognition accuracy by using a two-step entry process. First ask for the number of digits, then ask for the digits. The recognizer will know how many digits to expect from the caller. For example:

"How many digits are in your house number?"
< 3 >
"Please say the three-digit number now."

If the digit sequence can have more than nine digits, remind the caller to say the number in digit format ("one, two") rather than as a natural number ("twelve").

Different types of credit cards may have different length numbers. If you allow callers to use one of several types of credit cards, use a menu to prompt them for the type of card, then for the numbers. This lets you know how many numbers to expect from the caller, so the recognizer will look for a constant-length number. For example:

"What type of credit card will you be using today? American Express, Discover, MasterCard, Visa, or Universal Card?"
< Visa >
"What are the first four digits on your Visa card?"
Entering digit sequences

For entry of sequences of 10 or fewer digits, the system provides constant-length grammars. A grammar is the set of rules that the speech recognition software uses to understand data, such as caller input. For entry of sequences greater than 10 digits, it is recommended that you use a custom grammar. Otherwise, consider grouping the input. That is, prompt callers to read the series of digits in segments (groups), then confirm each group before going to the next. This technique is especially useful if the digit sequence cannot be validated with a checksum or custom grammar. If the number already has natural groups like in a telephone number, use those groups. For example:

"Please say the first four digits of your 12-digit number."
< 4384 />
"4384. Is that correct?" {pause} "Please say yes or no."
< yes >
"What are the next four digits?"
< 9556 >
"9556. Is that correct?"
< yes >
"And the next four digits?"
< 8833 >
"883. Is that correct?"
< no >
"Please repeat the four digits."
< 8833 >
"8833. Is that correct?"
< yes >

This call is shorter if callers are allowed to barge-in during the prompt to answer the yes/no questions quickly.

Connected-digit versus tone-paced digit entry

WholeWord speech recognition supports the entry of connected-digit sequences, in all available languages. That is, callers can say a series of digits in one long breath, without intentional pauses in between. Many callers find this the most natural way to enter a series of digits.

You may have had experience with voice response systems that do not support connected-digit entry. In order to simulate connected digit input, other systems use pacer tones. When an application uses pacer tones, callers are prompted to speak the first digit, then wait for another tone before speaking the next digit. The tones then continue until all digits have been spoken.

If you prefer tone-paced method of digit entry, you can write your applications to use pacer tones. However, this type of data entry has not been proven to increase recognition accuracy rate for digit sequence. Pacer tones may even result in lower accuracy, since callers may
become confused by the entry method. Avaya recommends using the connected-digit entry method.

**Use examples within a digit entry prompt**

You may need to prompt callers to enter information that is not usually expressed in digit format. For example, dates are usually spoken "June eighteenth" or "six eighteen." However, WholeWord speech recognition recognizes only single digits. June eighteenth must be spoken "six, one, eight" or "zero, six, one, eight." This format is awkward for many callers. In order to make this easier, include an example in the prompt. For example:

"Next, enter the month and day of the month you were born. Use two digits for the month, and two digits for the day. For example, for June eighteenth, you would say 'oh six, one eight.' Please enter the month and day now."

Use an example whenever you think that inexperienced callers will be less confused if you give them some help.

---

**Yes/no questions**

Use the following guidelines as you design your applications that include questions that require a "yes" or "no" response, also known as yes/no questions.

**Touch-tone input for yes/no questions**

In situations where you want callers to respond with touchtone input for "yes" or "no," ask them to press 1 for yes or 2 for no. For example:

"If this is correct, press 1. If not, press 2."

It is best to instruct callers to press 1 and 2, although you may want to use 1 and 0. Even if you prompt callers to press only 1 and 2, you can set up your application to accept additional numbers. For example, 1 for yes and 2 and 0 for no. Whatever numbers you use to represent yes and no, use the same numbers throughout the application. Do not confuse callers by using 2 for no in one place and 0 for no in another.

**Spoken input for yes/no questions**

In situations where you want callers to respond with spoken input, instruct them to say "yes" or "no," rather than a number. Callers find this natural, and make fewer mistakes than if you ask them to say a number meaning yes or no. For example:

Instead of prompting...
If this is correct, say 1, if it is not correct, say 2

Prompt...
Is this correct? Please say yes or no, now.
Designing a voice response application

Write your yes/no questions similarly throughout the application, so that callers will know what to expect.

After asking a yes/no question, pause to give the caller time to respond, then present the possible answers. The prompt will stop playing as soon as the recognizer detects a spoken "yes" or "no" or a touchtone signal. For example:

"You said '64587.' Is this correct?"
{a 1.5 second pause}
"Please say 'yes' or 'no'."

Pace the application

Use the following guidelines to design the pace and speed of your applications.

When callers must wait

If you know callers will have to wait, it is a good idea to let them know. For example, the application could say:

"Please wait while your request is processed."

If you know that the caller may have to wait for longer than a few seconds for the application to continue (because of a database call or host connection, for example), it is a good idea to fill the gap during the wait. You could record a long phrase, perhaps music, and play it to the caller while the information is being located. Or, play a different phrase every seven to ten seconds to let callers know that work is being done. To do this, you must use an external function.

Allow time for caller responses

Pace the interaction so that callers have time to enter the required information and listen to the prompts and announcements. You can determine how comfortable callers are with the pace of the interaction by testing the application.

Do not let the application respond to caller actions too quickly. After callers are asked to make an entry, play a speech phrase consisting of about a half second of silence before the next prompt or error message. Since many telephones have the keypad built into the receiver, this pause leaves time for callers to lift the receiver back to their ears after pressing the keys. If your calling population includes senior citizens, this pause can give them more time to react.

Adjustable pacing

Write your application to allow pauses to be adjusted easily. You can include a pause as mentioned earlier, but you can also change the length of the pause during your application
testing. In the beginning of your application, define a field called `entry_pause`. Assign to `entry_pause` the speech phrase number associated with one of the silence phrases.

Then, throughout the application, speak the field `entry_pause` before the prompts that follow caller entries. If you find during testing that the application responds too quickly, all you need to do is go to where you set the field value and use a longer silence phrase. With this method, you can make many changes in the application by changing only one line. You can do the same thing for other pauses in your application, such as pauses between menu items.

---

**Application errors**

Use the following guidelines when designing your applications to process errors.

**When caller errors occur**

Caller errors occur when callers enter information that the application considers incorrect, or when callers enter nothing at all. When callers make errors, provide them with an informative error message. Tell them what went wrong and how to correct it. To assure that callers hear error messages, do not allow barge-in or dial through during at least the first part of error messages.

Allow callers no more than two or three tries to enter information. After the first error (and second, if allowing three), the application should speak out a prompt that contains more information, so that callers can get a better idea of what is required. After the last try, tell callers to hold for an attendant, then transfer the call.

**When speech recognition errors occur**

Speech recognition errors occur when the recognizer cannot match the caller’s speech to a phrase it is prepared to recognize.

Use the following guidelines for processing speech recognition errors:

- Choose one concise phrase and use it consistently within a single application. For example:

  **Instead of phrases such as...**
  We are sorry, but your speech cannot be understood.
  Please repeat the category you want.

  **Use phrases such as...**
  Sorry. Please repeat.

- When using a word list of five or fewer items, reprompt with a "sorry" phrase, then give a list of the valid options. For example:

  "Sorry. Please say checking, savings, interest rates, loan rates, or operator, now."

- When using a large word list, include some helpful information in the reprompt. For example:
Designing a voice response application

"Department, please."
< Excuse me? >
"Sorry. Please say the name of the department you want, or hold to speak with an attendant."

Effective wording of error messages

To speak out an informative error message using IVR Designer, use the Initial Prompt field on the Response tab within a Prompt and Collect node. Set this field to False for Try Again and True for Reprompt. Be sure to specify the informative error message in the associated Voice Response field. Specify an appropriate error message for each type of error that can occur (Initial Timeout, Too Few Digits, or Not on List), as shown in the following examples for touchtone input:

- If the first prompt is:
  "What time would you like your wake-up call?"
  and the caller does nothing (Initial Timeout error)

  A more informative prompt could be:

  "Please enter the time you would like your wake-up call. Use the telephone keypad to enter two digits for the hour and two digits for the minutes."

- If the first prompt is:
  "Enter the date of the lottery."
  and the caller presses some touch tones, but not enough (Too Few Digits error)

  A more informative prompt could be:

  "Please enter the date of the lottery for which you want to hear winning numbers. Use the telephone keypad to enter two digits for the month, and two digits for the day. If a month or day has only one digit, use a zero before the digit."

Avoid using negative or accusing words when indicating an error. Words such as "invalid," "bad," "wrong," or "incorrect" can make callers feel threatened. "Sorry" is a brief way to say that the input was not recognized or was incorrect, and "Please repeat" or "Please enter again" are brief ways to ask callers to act again.

Touch-tone and speech recognition

Although you might already know how to design good touchtone applications, the information in this section will help you understand and appreciate the differences between touchtone recognition and speech recognition, and how the technologies affect application design.

Touchtone input and spoken input are processed in fundamentally different ways, so the application designer must take care to use each feature in a way that is tailored to how callers will interact with the application.
The following table compares and contrasts touchtone interactions with speech recognition interactions.

<table>
<thead>
<tr>
<th>Possible input</th>
<th>What the system can recognize</th>
<th>How the system processes the input</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>One or more tones</td>
<td>Touchtone signals</td>
<td>Takes touchtone signals and maps to a number on the keypad</td>
<td>Excellent mapping between input and what the recognizer &quot;knows&quot;</td>
</tr>
<tr>
<td>Any noise that a person can make, or any background noise that can be heard over the telephone</td>
<td>A set of word models restricted by grammar</td>
<td>Takes a speech sample and tries to match it to the active word models</td>
<td>Imperfect mapping between input and what the recognizer &quot;knows&quot;</td>
</tr>
</tbody>
</table>

**Application examples**

Suppose that you are working on an application that callers use to report problems with telephone service. Part of your transaction involves asking callers for a telephone number. For a touchtone interaction, you could prompt "Enter your 10-digit telephone number." The caller can do one of three things: press 10 keys, press fewer than 10 keys, or do nothing.

- If the caller presses 10 keys, the application reads back the telephone number for confirmation: "Did you enter 614-860-4001?"

- If the caller presses fewer than 10 keys, the application detects this and reprompts the caller for a valid entry.

- If the caller does nothing, the application receives no touchtones. When this happens, the application can reprompt, providing more information.

For a similar speech recognition interaction, you could prompt, "Say your 10-digit telephone number." The caller can say anything at all in response (words or other sounds) or say nothing. The recognizer listens, then attempts to map the sounds to the appropriate grammar (US_DIG, with minimum and maximum digits set to 10), and come up with a series of digits. No matter what the caller says, the recognizer does the same thing: it listens to the sounds coming through the telephone, and attempts to map 10 digits, or using phrase screening, rejects the utterance.

- If the caller says 10 digits, the recognizer tries to map the speech to 10 digits, then reads back the result for confirmation: "Did you say 614-860-4001?"

- If the caller says fewer than 10 digits, the recognizer attempts to map the speech to 10 digits, then reads back the result for confirmation: "Did you say 618-604-0001?"
Designing a voice response application

- If the caller remains silent and there is background or line noise, the recognizer rejects the utterance. The recognizer returns a null string to the application, and the application can reprompt, giving the caller more information. This is considered a "correct rejection."

- If the caller says "I do not have a phone number yet," the recognizer rejects the utterance.

What can be done to improve the application design in this example? What could increase the chance that callers will say what the recognizer is prepared to accept? Your application could first ask if the caller has the required information, by prompting: "Do you have your telephone number available?" Callers who say no can be transferred to the attendant, while callers who say yes can be prompted for the number.

Touchtone input used with spoken input

You can use WholeWord speech recognition to allow callers without touchtone telephones to use the voice response system by speaking "yes" or "no."

There are three different approaches for planning the application:

- Write two separate applications that are nearly identical, one to process touchtone input and one to process spoken input.
- Write a single application to process both touchtone and spoken input.
- Modify a touchtone input application to process both types of input.

Separate touchtone and speech recognition applications

This approach allows you to use prompts that are best for touchtone entry in one application and those that are best for speech recognition entry in the other.

In many cases, touchtone recognition may be more accurate and cost less than speech recognition. Therefore, you probably want callers with touchtone telephones to use their keypads to interact with the application. To encourage this, start your application with a dial 1 prompt. A dial 1 prompt gives callers a chance to indicate that they have a touchtone telephone, and helps speed the call.
The following figure shows a flowchart for a dial 1 prompt.

**A single touchtone and speech recognition application**

An application that uses touchtone recognition and speech recognition together requires that every prompt be acceptable for both entry methods.

**Note:**
Use this method only if your system resources can process the extra load, since many callers with touchtone telephones may choose to use speech recognition instead. Remember that recognition accuracy is lower for spoken input than for touchtone input.

When using a single application:
Designing a voice response application

1. Structure your prompts so they give callers the choice of using the touchtone keypad or speaking.

2. At the beginning of the application, tell callers that entries may be made with either the keypad or by speaking. For example, the application might say:

"Welcome to XYZ Technologies. During this call, you can use your telephone touchtone keys to make entries, or you can simply speak your responses."

3. Make sure that the prompts are appropriate for either kind of input. There are two ways to do this:

   — Tell callers to "enter" the information. For example:

   "Enter the zip code of the restaurant you visited."

   — Present the prompt as a question. For example:

   "What is the zip code of the restaurant you visited?"

Either method is appropriate. To accomplish this in IVR Designer, write the application the same as you would for speech recognition input. For digits, the application automatically accepts touchtone input in place of the spoken input. For "yes" and "no," you must identify in the application what touchtone keys are associated with the words "yes" and "no," and specify the action associated for each. For IVR Designer applications, insert the appropriate branches on the Prompt and Collect node.

You also must tell callers to press a touchtone key instead of saying "yes" and "no," in case they want to indicate their response with a touchtone key. The prompts to accomplish this could be:

Prompt: "Would you like to leave us a message regarding the service agreement?"
{pause}
"Please enter yes or no."

Reprompt: "Please say `yes' or `no'. Or, for yes, press 1. For no, press 2."

Use touch-tone input when speech recognition fails

If you are using speech recognition, you may increase the number of successful interactions by encouraging callers to use touchtone input when spoken entries have been recognized incorrectly.

Touchtone input may be more accurate than spoken input. You can reprompt callers to indicate their response with touchtone input if they have a touchtone telephone. The implementation is not difficult, since an application written to accept spoken input will also accept touchtone input.
A typical prompt set might look like this:

"Please enter the 6-digit code for the fax information you want."
<"135683">
"You entered 135688. Is this correct?"
<"No">
"Please reenter the code. If you have a touchtone telephone, you may want to enter the code using your keypad, instead."

**Modify a touch-tone-only application to include spoken input**

When you modify a touchtone input application to support spoken input (in addition to or in place of touchtone input), remember to examine the application to see what the prompts say and what action is expected from callers. Think about the capabilities of the speech recognition software you will be using, and where they might fit into the application.

The following table shows the words the wordlist might contain for a homework assignment application:

<table>
<thead>
<tr>
<th>Word</th>
<th>Maps to option</th>
</tr>
</thead>
<tbody>
<tr>
<td>listen to homework</td>
<td>Listen to homework option</td>
</tr>
<tr>
<td>homework</td>
<td>Listen to homework option</td>
</tr>
<tr>
<td>listen</td>
<td>Listen to homework option</td>
</tr>
<tr>
<td>leave a message for the teacher</td>
<td>Leave a message for the teacher option</td>
</tr>
<tr>
<td>leave a message</td>
<td>Leave a message for the teacher option</td>
</tr>
<tr>
<td>teacher</td>
<td>Leave a message for the teacher option</td>
</tr>
<tr>
<td>choose a different class</td>
<td>Choose a different class option</td>
</tr>
<tr>
<td>different class</td>
<td>Choose a different class option</td>
</tr>
<tr>
<td>different</td>
<td>Choose a different class option</td>
</tr>
<tr>
<td>class</td>
<td>Choose a different class option</td>
</tr>
</tbody>
</table>

If you plan to use WholeWord speech recognition, your application will be able to recognize the words "yes" and "no" and connected digits. In places where the touchtone application prompts for digit entry (especially long sequences), you may want to introduce custom grammars, checksums, or grouping to get the highest recognition accuracy possible.
Designing a voice response application

Dial through and barge-in

Use the following guidelines to design your applications to encourage or discourage dial through and barge-in.

Using dial through and barge-in with error messages

For error messages, require callers to listen to at least part of each error message, so the interaction stops and the mistake can be corrected. For touchtone and WholeWord speech recognition input, turn off dial through or barge-in during the first part of each error message. For example:

Turn dial through or barge-in off and prompt:
"Sorry. Please enter..."
Turn dial through or barge-in on.
"...your customer number."

Using dial through and barge-in consistently

Callers expect and appreciate consistency. Therefore, it is a good idea to apply dial through and barge-in consistently in your applications. Allowing callers to barge-in during some prompts and not others in the same application could cause confusion. You may find, however, that your callers may want to barge-in during some prompts, and may be able to accept some inconsistency. During your design testing, determine what will work best for your calling population, and design your applications accordingly.

How to word prompts for dial through and barge-in

The wording, length, and structure of prompts can discourage callers from responding during a prompt. You want to ensure that the caller's responses are consistent with how you have implemented dial through and barge-in.

How to encourage dial through and barge-in

Experienced callers like to shorten the call by responding during the prompt, and thus save time because they know what to say or do. Therefore, if you have the dial through or barge-in capability enabled for a prompt, you should encourage callers to respond when they are ready, even during the prompt. To encourage dial through or barge-in for a menu, leave a pause after each option. Using the Prompt & Collect action, play an option, then play a series of silence phrases equal to 1.5 seconds. Silence is not needed after the final option. For example:

"For sales, say '1' {1.5 second pause}
For service, say '2' {1.5 second pause}
For an attendant, say 'zero'."
Barge-in uses system resources also used by speech recognition. If resources are limited, you could disable barge-in until you have expanded your speech recognition resources.

**How to discourage dial through and barge-in**

If you have the dial through or barge-in capability disabled for a prompt, you want callers to respond only after the prompt is finished to optimize the recognition. To discourage dial through or barge-in for a menu, do not leave a long pause between options. For example:

"For sales, say '1' For service, say '2' For an attendant, say 'zero'."

When barge-in is turned off or unavailable, make sure that your recorded prompts have no silence at the end. Callers may respond during the silence. When this happens, the recognizer will not hear part of the caller's response.

---

**Text-to-Speech in applications**

*Text-to-Speech in Applications* provides guidelines for designing applications that use Text-to-Speech.

Text-to-Speech converts ASCII text to spoken words. This ability is especially useful when it is not practical to prerecord the information to be read, for example if the information is lengthy, changes often, or comes from a database. Text-to-Speech is also useful for testing applications.

---

**Using Text-to-Speech for prompts and announcements**

**Touch-tone input**

When Text-to-Speech is used with touchtone input, callers can interrupt, or dial through, the prompt or announcement, just as they can when the prompt has been prerecorded.

If callers have problems dialing through a prompt or announcement with touchtone input, make sure that dial through is enabled. When using dial through, remember to flush all touchtone input from the buffer whenever the application plays an error message.

**Spoken input**

When Text-to-Speech is used with speech recognition, callers can barge-in during the prompt or announcement. If you have long announcements, you can prompt callers to press a touchtone key or speak when they have heard all of an announcement that they want to hear.
Using both Text-to-Speech and prerecorded speech prompts and announcements

A single application can speak prompts and announcements in either prerecorded speech or Text-to-Speech, or a combination of the two. New callers find Text-to-Speech more difficult to understand than prerecorded speech. Therefore, some application designers choose to speak out unchanging information (including most prompts) with prerecorded speech, unless the number of phrases makes pre-recording unmanageable. The best way to decide between these two alternatives is to test both with some of your callers.

The following example shows how you might use both Text-to-Speech prompts and prerecorded prompts in an application. With a wordlist including 350 names, instead of prerecording all of the names, you could use Text-to-Speech to speak the recognized name for verification. Then use prerecorded speech for the rest of the prompts. For example:

"Name, please." (prerecorded speech)
\textless John Smith \textgreater
"John Smith" (Text-to-Speech)
"Yes or no?" (prerecorded speech)

Getting the most out of Text-to-Speech

Use the following guidelines to improve Text-to-Speech performance in your applications.

Help callers adjust to Text-to-Speech output

Your callers may need to adjust to the computer voice of Text-to-Speech output to understand it well. To help them, you could speak out some less important information before speaking out the information callers need. This gives callers a chance to become familiar with the rhythm and intonation of Text-to-Speech.

Because the sound of Text-to-Speech may be unfamiliar to your callers, consider giving them the option of having the information repeated, or spelled, if necessary. Spelling is especially useful with names, as Text-to-Speech may not pronounce names as the caller would.

Use complete sentences

Text-to-Speech works primarily as a reading machine. It operates under the assumption that the information it is reading is structured in standard sentences (using punctuation, capitalization, subject, object, and verb). In order to make Text-to-Speech output sound most natural, use good grammar, complete sentences, and punctuation in the input text to be spoken.
What if the information you want to speak is not written in complete sentences? Since data fields cannot be punctuated, you may be able to control the output by changing the speaking rate and pauses between the information.

**Pauses**

You can control the pace of the Text-to-Speech output by inserting pauses. If you are speaking text that you control, the easiest way to do this is with punctuation within the words to be read. Remember to punctuate exactly like you would in a sentence (for example, do not leave a space before a period or a comma).

When speaking out a large block of text, you may hear a pause where you do not want a pause. First, check if there is any stray punctuation causing the pause. If not, you can insert a short recorded silence phrase before the sentence during which you heard the pause. This should eliminate the misplaced pause. If the text block is from a remote database, however, this may not be possible.

**Eliminate typographical errors**

Text-to-Speech pronounces exactly what is written, so typographical errors can cause mispronunciations. To make Text-to-Speech output as understandable as possible, look for and listen for typographical errors in the ASCII text, and remove them when you test the application.

**Remember Text-to-Speech pronunciation differences**

Text-to-Speech relies on built-in rules, but cannot account for all exceptions. Therefore, it may mispronounce words, especially some names. If Text-to-Speech mispronounces a word, use phonetic spelling to correct the pronunciation. For example, Text-to-Speech pronounces the name "Bagge" as "baggy," but the correct pronunciation is with a silent e. You can change the spelling of the name to "bag" so that Text-to-Speech pronounces it correctly.

Another way to overcome mispronunciation or misunderstandings is to spell some of the information, especially for names. Design the application to speak the name, then spell it out. Or, you can give callers the option of having a name spelled out.

---

**Testing Text-to-Speech applications**

Writing applications with understandable Text-to-Speech output is a process of trial and error. It is very important to plan the application, then test it with real data to ensure that the speech is understandable. Once you can understand the output well, test the application with some representative callers. Refine and test it again until you and the callers are satisfied with the result.
Designing a voice response application
Creating a voice response application

Creating a voice response application describes how to use the various tools and components of IVR Designer to create a voice response application.

This section includes the following topics:

Working with nodes ................................................................. 61

The Node Inspector ................................................................. 70

Searching for information in applications ............................ 72

Working with call flows ............................................................ 75

The Globals Manager ............................................................... 79

Working with variables ............................................................ 83

Working with prompts ............................................................. 89

Working with phrases .............................................................. 104

Working with VoiceXML grammars ........................................ 121

Using double-byte data .......................................................... 126

VoiceXML database operations .............................................. 128

Working with nodes

Nodes are the basic building blocks used to create an application. You use nodes to build your call flows by dragging node objects or icons from the Node Palette to the design pad.

IVR Designer offers many different nodes, each with its own attributes and behaviors. For detailed information about a specific node, see Node descriptions on page 349.
Creating a voice response application

Placing a node in a call flow

To place a node in a call flow, drag the node's icon from the Node Palette to the design pad. Once you place the node where you want it, the Node Inspector opens. You can use the Node Inspector to view, define, and edit the node's attributes.

Drop the icon onto any open branch. Or, if you do not want to connect the node to the call flow, drop the icon below the last node.

You can also drop the node before an existing node, including at the top of a call flow, by dropping it on top of an existing first-level node. IVR Designer does not let you drop a node on top of a node of any other level.

Dropping a node onto an existing node

While you are dragging a node, the appearance of the cursor changes to let you know whether you can paste the node where the cursor is. When the cursor appears as an arrow, you can drop the node where you want it, either above the node or onto an empty node branch. When the cursor appears as a circle with a slash across it, if you drop the node, IVR Designer places it at the end of the active call flow. IVR Designer also places the node at the end of the active call flow if the cursor is located after the end of the last node in the active call flow.

Cursor change showing invalid drop status

You can also drag and drop nodes (and all subordinate nodes) to move them within a single call flow. When you place your cursor on the node you want to move, the cursor changes as described above to let you know where you can drop it. This is identical to a cut-and-paste operation.
IVR Designer also lets you select multiple nodes within a call flow, but they must be contiguous nodes. That is, they must have the same parent node and be adjacent to each other on the call flow. As with single nodes, all subordinate nodes are copied or moved with the parent nodes.

About node branches

Node branches are indicated by lines leading down and to the right from a node.

Node branch example

The following illustration shows an example of empty and occupied node branches.

You can drag-and-drop a new node icon onto any empty branch. An empty branch is indicated by a horizontal line that does not connect to any node.

Inserting and renaming branches

For some node types, you can insert new branches under them or rename existing default branches.

When you select Insert Branch or Rename Branch from the right-click menu, the results depend on what type of node you are working with. Nodes that you can insert and rename branches include:

- Menu Node
Creating a voice response application

- Prompt and Collect Node
- Time Branch Node
- Call Flow Node
- Set and Test Node

Node editing guidelines

Note:
When you cut or copy a single node, that node and all subordinate nodes attached to it are cut or copied. To cut a single node without cutting its subordinate nodes, you must first disconnect the subordinate nodes, then cut the desired node.

If an open branch is not available or if you attempt to paste the nodes at a location that is not an open branch, IVR Designer places the nodes at the bottom, disconnected from the existing call flow. This is different from many other Windows applications that let you paste a selection over another highlighted selection, thus replacing that selection.

IVR Designer does not let you paste nodes from one application into a different application if the selected language for the two applications is different.

The Paste action can be undone only if it is being done within a single application. If the Paste action is being done between two applications, it cannot be undone.

When attempting to paste, IVR Designer identifies and notifies you of any unresolved issues before either completing or blocking the paste operation. If IVR Designer is able to complete the paste action but finds an error involving a required attribute for a node, it places a message in the Node Notes field for the affected node.

Cut/Copy/Paste operations can be used both within an application and between different applications. When using these operations between applications, you must use the following procedure:

1. Cut or copy the desired nodes in the original application.
2. Save and close the original application.
3. Open the application you want to paste the desired nodes into.
4. Paste the nodes where you want them.

If a resource does not exist in the target application, IVR Designer creates or imports it during the paste operation. If one or more resources do exist, IVR Designer offers the option to overwrite them. If you elect to continue, all existing resources are overwritten. If you elect to cancel the overwrite action, the paste action is completed without overwriting existing resources.
During cut/copy/paste operations, all selected nodes are recreated in their entirety, including attributes and resources, in the target applications. Attributes include:

- Variables
- Phrases
- Prompts
- VoiceXML grammars
- Encapsulated call flows

For practice copying nodes and applications, see Copying nodes and VoiceXML applications on page 219.

**Name conflicts during paste actions**

Node names must be unique within a call flow. When you attempt to paste selected nodes, IVR Designer checks for node name conflicts. If a node name matches one in the call flow where you are attempting to paste it, IVR Designer prompts you for a new name. If you select Cancel, the paste action is canceled.

**Call Flow** nodes are an exception to this name rule. A Call Flow node name is always appended with a number in parentheses indicating how many times it has been used in each call flow. Thus, when a Call Flow node is copied, the number is changed, but the name remains the same. For example, if a call flow containing a Call Flow node named *My_flow(1)* is copied to a different place in the same call flow, the new name is *My_flow(2)*.

**Note:**
The attributes of the copied nodes remain unchanged.

The paste operation uses the resource pathnames in pasting the new application. If the target application already has a resource with the same name as one in the nodes you are trying to paste, IVR Designer prompts you to specify whether you want to keep the original resource on the target system or overwrite it with the new one.

**Selecting multiple nodes for cut, copy, and paste**

IVR Designer lets you cut, copy, or paste multiple nodes within a call flow, but they must be contiguous nodes. That is, they must have the same *parent* node and be adjacent to each other on the call flow. As with single nodes, all subordinate nodes are copied or moved with the parent nodes.
Creating a voice response application

To select multiple nodes for cut, copy, and paste actions:

1. Click the first node that you want to include in the action.
2. Shift-click the last node at the same level that you want to include in the action.

**Note:**
If you try to shift-click a node at a different level, only that node is selected.

---

**Using drag-and-drop to cut, copy, and paste**

Within a single call flow, you can use drag-and-drop techniques to cut, copy, and paste nodes in IVR Designer. If you are in the Multiple Document Interface mode, you *cannot* drag and drop nodes across call flows.

When you drag and drop nodes within a call flow, the appearance of the cursor changes to let you know whether or not you can paste the nodes at the cursor's current location. When you are at a valid drop location, the cursor appears as an arrow with a document attached, as shown in the following figure.

![Valid drop location cursor](image)

When you are at an invalid drop location, the cursor appears as a circle with a slash mark, as shown in the following figure.

![Invalid drop location cursor](image)

If you attempt to drop the selected nodes when the cursor has this appearance, the cut, copy, or paste action is canceled and the nodes return to their original locations. This rule holds true for both single and multiple node cut, copy, and paste actions.

To perform a cut, copy, or paste action on multiple contiguous nodes, you must first make sure you have enough open branches to accommodate the paste action where you want the nodes to go.

If you do not have enough open branches, starting with the target branch and counting those immediately below it, to accommodate the number of nodes you want to paste, and you drop them anyway, the nodes are disconnected and pasted at the end of the call flow.

Select the nodes you want to cut, copy, or paste, then hold the key down while you drag the selected nodes to the desired location.
Editing node attributes

Use the **Node Inspector** to view and edit the attributes and elements of a node.

In general, to enter or edit values for a node’s attributes, you must do one of the following:

- Type the literal value in the appropriate field.
- Make a selection from a drop-down menu.
- Drag and drop a resource from the appropriate resource manager.

For detailed information about a specific node, see **Node descriptions** on page 349.

Right-Click menu options for nodes and empty node branches

The following are general descriptions of the right-click menu options available for nodes and for empty node branches.

Not all options are available for each node type. These are general descriptions only. For the specific details of how these options work with specific types of nodes, see the appropriate node description.

**Right-click menu options for nodes**

The following table describes the menu options available by right-clicking a node.

**Note:** Parentheses ( ) around a menu option name indicate that the option is not available for all node types.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td>Removes the selected node and any subordinate nodes and copies them to the clipboard.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies the selected node and any subordinate nodes to the clipboard.</td>
</tr>
<tr>
<td>Paste</td>
<td>Inserts any nodes on the clipboard at the point where the cursor is.</td>
</tr>
<tr>
<td></td>
<td>This option is available only after you have done a cut or copy action.</td>
</tr>
<tr>
<td>Branch Notes</td>
<td>Opens the <strong>Branch Note</strong> dialog box, where you can enter a short description for the branch. Branch notes are displayed in the main window status bar whenever the cursor is over that branch.</td>
</tr>
<tr>
<td>(Delete Branch)</td>
<td>When available, this option deletes the entire branch from the selected node. <strong>Note:</strong> This action cannot be undone.</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>(Insert Branch)</td>
<td>When available, this option inserts a new branch into the call flow.</td>
</tr>
<tr>
<td>(Rename Branch)</td>
<td>When available, this option lets you rename the branch.</td>
</tr>
<tr>
<td>Delete Node</td>
<td>Removes the selected node and any subordinate nodes from the call flow. <strong>Note:</strong> This action cannot be undone.</td>
</tr>
<tr>
<td>Disconnect Node</td>
<td>Separates the selected node and any subordinate nodes from the call flow.</td>
</tr>
<tr>
<td>(Edit Flow)</td>
<td>(Active only with the Call Flow node) Opens the <em>encapsulated node</em> call flow, making it the active call flow.</td>
</tr>
<tr>
<td>Edit Node</td>
<td>Opens or activates the <strong>Node Inspector</strong>, where you can edit the selected node's attributes.</td>
</tr>
<tr>
<td>(Find Jumped To Node)</td>
<td>(Active only for nodes that have the ➙ icon following the node icon) Finds the node that the selected node jumps to.</td>
</tr>
<tr>
<td>Encapsulate</td>
<td>Converts the selected node to a Call Flow node and encapsulates that and any subordinate nodes, making them into a separate call flow.</td>
</tr>
<tr>
<td>(Expand Node)</td>
<td>Reverses the Encapsulate command, expanding an encapsulated node and moving the components from separate call flow status to incorporate them into the parent call flow.</td>
</tr>
<tr>
<td>Print Node</td>
<td>Prints out a summary of the selected node’s attributes.</td>
</tr>
<tr>
<td>Rename Node</td>
<td>Lets you to rename the node.</td>
</tr>
<tr>
<td>Verify Node</td>
<td>Opens and runs the <strong>Verify Design</strong> tool on only the selected node.</td>
</tr>
<tr>
<td>Print Call Flow</td>
<td>Prints out the active call flow similar to the design pad display of the call flow.</td>
</tr>
<tr>
<td>Verify Call Flow</td>
<td>Opens and runs the <strong>Verify Design</strong> tool on only the active call flow.</td>
</tr>
<tr>
<td>Close</td>
<td>Displays a list of all the currently open call flows, selecting a call flow from the list closes it. <strong>Note:</strong> This option is available only when <strong>Tabbed Interface</strong> is selected on the <strong>Environment</strong> tab of the <strong>Preference Editor</strong>.</td>
</tr>
<tr>
<td>Open</td>
<td>Displays a list of all currently closed call flows, selecting a call flow from the list opens it. <strong>Note:</strong> This option is available only when <strong>Tabbed Interface</strong> is selected on the <strong>Environment</strong> tab of the <strong>Preference Editor</strong>.</td>
</tr>
</tbody>
</table>
Right-Click menu options for an empty branch

The following table describes the menu options available by right-clicking on an empty branch under a node.

**Note:**
Parentheses ( ) around a menu option name indicate that the option is not available for all node types.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td>Grayed out (inactive) on an empty branch.</td>
</tr>
<tr>
<td>Copy</td>
<td>Grayed out (inactive) on an empty branch.</td>
</tr>
<tr>
<td>Paste</td>
<td>Inserts any nodes on the clipboard at the point where the cursor is.</td>
</tr>
<tr>
<td></td>
<td>This option is available only after you have done a cut or copy action.</td>
</tr>
<tr>
<td>Branch Notes</td>
<td>Opens the <strong>Branch Note</strong> dialog box, where you can enter a short description of the branch. Branch notes are displayed in the main window status bar whenever the cursor is over that branch.</td>
</tr>
<tr>
<td>(Delete Branch)</td>
<td>When available, deletes the entire branch from the selected node.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This action cannot be undone.</td>
</tr>
<tr>
<td>(Insert Branch)</td>
<td>When available, inserts a new branch into the call flow.</td>
</tr>
<tr>
<td>(Rename Branch)</td>
<td>When available, lets you rename the branch.</td>
</tr>
<tr>
<td>Connect Node</td>
<td>Opens a dialog box that displays a list of nodes in the call flow that are currently disconnected. Selecting a node from the list connects it to the current call flow at the node where this option was selected.</td>
</tr>
<tr>
<td>Copy Call Flow</td>
<td>Opens a dialog box that displays a list of call flows currently defined for the application. Selecting a call flow from the list copies it to the current branch where this option was selected. <strong>Note:</strong> The [Main] call flow cannot be copied.</td>
</tr>
<tr>
<td>Edit Node</td>
<td>Opens the <strong>Node Inspector</strong>, where you can edit the selected node's attributes.</td>
</tr>
<tr>
<td>Jump to Node</td>
<td>Opens a dialog box that displays a list of nodes in the call flow to which you may jump from the currently selected node.</td>
</tr>
</tbody>
</table>
The Node Inspector

The Node Inspector displays different information and editing options for each node, depending on the node's predetermined attributes. It provides a convenient way for you to edit node attributes. For more information about nodes, see Node descriptions.

Opening the Node Inspector

To open the Node Inspector, select Node Inspector from the Tools menu. IVR Designer displays the Node Inspector window.

Node Inspector window elements

The following table describes the elements of a typical Node Inspector window.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node selection drop-down menu</td>
<td>The drop-down menu at the top of the Node Inspector lets you select which node to edit.</td>
</tr>
<tr>
<td>Node attribute fields</td>
<td>The fields below the Node selection drop-down menu let you set attributes for the selected node.</td>
</tr>
<tr>
<td>Node attribute tabs</td>
<td>When available, the tabs at the bottom of the Node attributes area let you select groups of attributes for editing.</td>
</tr>
</tbody>
</table>
Selecting a node to view or edit attributes

To view or edit a node's attributes:

1. Open the Node Inspector.
2. Select the desired node by doing one of the following:
   - From the active call flow, highlight the node in the design pad.
   - Use the Node Select drop-down menu, located near the top of the Node Inspector window.

   **Note:**
   Only the nodes used in the active call flow are available from this drop-down menu.

   Once this menu is selected, you can view and edit the node's attributes.

   To view additional information or hidden text, resize the Node Inspector window.
   Just above the Node Select drop-down menu, IVR Designer displays the icon and name of the node type you are viewing or editing.

Node Inspector tabs

For most nodes, attributes that are related by function or type are grouped together. If there is more than one set of attributes, each set is designated by a tab near the bottom of the Node Inspector window.

To select the set of attributes you want to view or edit, click the appropriate tab.

Writing node notes

The Node Notes area is located at the bottom of the Node Inspector window, if the Show Node Notes option is checked in the right-click menu.

To write and save notes about a particular node, place the cursor in the text area and type the desired note text.
Creating a voice response application

Node notes stay with the node, for instance, if you copy and paste it, and can be viewed whenever the Node Inspector is active for that node. They also appear in the status bar at the bottom of the main window whenever the cursor rests on the node.

Node notes do not print when you print a report for the node using the Print command.

Right-click menu options for the Node Inspector

The following table describes the right-click menu options for the Node Inspector.

Note: The last four options are available only with nodes for which variable elements are defined. These are the Set and Test node and the Load Variables node.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Hints</td>
<td>Toggles on/off. When checked (toggled on), allows a help window to pop up whenever the cursor rests over a field in the Node Inspector window.</td>
</tr>
<tr>
<td>Show Node Notes</td>
<td>Toggles on/off. When checked (toggled on), displays the Node Notes area of the Node Inspector window and lets you enter, add, or edit comments. When unchecked (toggled off), hides the Node Notes area.</td>
</tr>
<tr>
<td>Stay on Top</td>
<td>Toggles on/off. When checked (toggled on), causes the Node Inspector window to remain on top of other open IVR Designer windows.</td>
</tr>
<tr>
<td>New Element</td>
<td>Lets you add a new element to the variable definition.</td>
</tr>
<tr>
<td>Delete Element</td>
<td>Delete an element from a variable definition.</td>
</tr>
<tr>
<td>Move Element Up</td>
<td>Moves an element upward in the variable definition table.</td>
</tr>
<tr>
<td>Move Element Down</td>
<td>Moves an element downward in the variable definition table.</td>
</tr>
</tbody>
</table>

Searching for information in applications

IVR Designer has powerful and versatile search-and-find capabilities. You can use the Find function on selected items or types of items to locate:

- Text strings
- Text strings in names
- Where items are being used
- Unused items

**Accessing Find**

To use the Find function:

1. Click the **Edit** menu.
2. Click the **Find...** menu option.
3. Click the tab for the search you want to perform.

The following table describes the Find tabs:

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named</td>
<td>Searches for text strings contained within the names of selected items.</td>
</tr>
<tr>
<td>Uses Of</td>
<td>Searches for locations where items are used within the application.</td>
</tr>
<tr>
<td>Unused</td>
<td>Searches for items that have been created, but not used, within the application.</td>
</tr>
<tr>
<td>Find Text</td>
<td>Searches for text strings located anywhere within the application.</td>
</tr>
</tbody>
</table>

**The Named tab**

Use the **Named** tab to search for text strings contained within the names of selected items.

To find a text string contained within the name of an item:

1. Type the text string in the **Find names containing string** field.
2. From the **Search items of type** drop-down menu, select the types of items you want to search in the current application.
3. Click the **Find Now** button.

IVR Designer displays the results of the search in a box below the **Find** window and highlights the item in the location where it finds the text string. If, for instance, the search finds the text string in the name of a node, it highlights the node in the call flow. If the search finds the text string in a phrase, it highlights the phrase in the appropriate phrase table.
Creating a voice response application

In many cases, the search results in multiple finds. To locate other search result items, click the item name in the results box. IVR Designer highlights the appropriate location of that item.

To clear the results of your search and start a new search, click the New Search button.

The Uses Of tab

Use the Uses Of tab to search for locations where items are used within the application. This can be useful in locating all the uses, for instance, of a certain call flow used as a subroutine within the application or locating where a particular external function is used.

To find where selected items are used within the current application:

1. From the Type of item: drop-down menu, select the kind of item you want to search.
2. From the Find uses of item: drop-down menu, select which specific item of that type you want to find.
3. From the Search items of type: drop-down menu, select which kinds of items you want to search through.
   - In most cases, this is either All types or the same as the Type of item: you selected in Step 1.
4. Click the Find Now button.

   IVR Designer displays the results of the search in a box below the Find window and highlights the location of the item. If, for instance, you were searching for a particular phrase, it highlights the phrase in the appropriate phrase table.

   In many cases, the search results in multiple finds. To locate other search result items, click the Item Name in the results box. IVR Designer highlights the appropriate location of that item.

To clear the results of your search and start a new search, click the New Search button.

The Unused tab

Use the Unused tab to search for items that have been created, but not used, within the application. This can be useful in locating unused items that you may want to delete or items you have created but not yet used.

To find items that have been created, but not used, in the current application:

1. From the Search items of type: drop-down menu, select the types of items you want to search in the current application.
2. Click the **Find Now** button.
   
   IVR Designer displays the results of the search in a box below the **Find** window and highlights the item. If, for instance, your search locates an unused call flow, IVR Designer makes that call flow the active call flow and displays it.
   
   To locate other find result entries, click that item name in the results box. IVR Designer displays the appropriate item.
   
   To clear the results of your search and start a new search, click the **New Search** button.

---

### The Find Text tab

Use the **Find Text** tab to search for text strings located anywhere within the application.

To find a text string contained within an item:

1. Type the text string in the **Find this text string**: field.
2. From the **Search items of type**: drop-down menu, select the types of items you want to search in the current application.
3. If it is important to match case (for instance, in searching proper names or titles), check the **Match case** checkbox.
   
   A checkmark appears to confirm that case matching is active.
4. Click the **Find Now** button.
   
   IVR Designer displays the results of the search in a box below the Find window and locates and highlights the item where it finds the text string. If, for instance, the search finds the text string in a node, it highlights the node in the appropriate call flow. If the search finds the text string in a prompt, it highlights the prompt in the appropriate prompt table.
   
   In many cases, the search results in multiple finds. To locate other search result items, click the item name in the results box. IVR Designer highlights the appropriate location of that item.
   
   To clear the results of your search and start a new search, click the **New Search** button.

---

### Working with call flows

A call flow in IVR Designer is a series of nodes or objects that are linked together.

The Call Flow Inventory tool lets you see:

- What call flows exist for your application
Creating a voice response application

- Where each call flow is used within the application
- What other call flows, if any, are used within each call flow.

For practice using call flows, see Using embedded call flows in VoiceXML applications on page 263.

Opening the Call Flow tool

To open the Call Flow Inventory tool, select Call Flow Inventory from the Tools menu.

Note:
To locate any call flow displayed in the Call Flow window, double-click the name of the call flow. IVR Designer locates the call flow, highlights it, and makes it the active window.

Displaying the Uses view

To display the Uses view, click the Uses tab in the Call Flows window.

The following table describes the elements of the Call Flows window in the Uses view.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Folder]</td>
<td>Indicates that the designated call flow is defined but contains no other call flows.</td>
</tr>
<tr>
<td>![Folder]</td>
<td>Indicates that the designated call flow contains other call flows, but that the directory for viewing them is closed.</td>
</tr>
<tr>
<td>![Folder]</td>
<td>Indicates that the designated call flow contains other call flows and that the directory for viewing them is open so that you can see them.</td>
</tr>
<tr>
<td>Uses tab</td>
<td>Lets you view the call flows defined for the current application and see what other call flows they reference (use).</td>
</tr>
<tr>
<td>Used by tab</td>
<td>Lets you view call flows and see which other call flows reference (use) them.</td>
</tr>
</tbody>
</table>

Expanding a call flow

To expand a call flow containing other call flows, double-click a call flow containing the ![Folder] icons.

The call flow expands to display its contents, that is, the subordinate call flows it uses. At the same time, the design pad displays the parent call flow.

To locate any other subordinate call flows in the parent call flow, double-click that subordinate call flow.
Collapsing a call flow

To collapse a call flow containing other call flows, double-click a call flow containing the  
icons.

The call flow collapses to hide its contents, that is, the subordinate call flows it uses. At the same time, the design pad displays the selected call flow.

Displaying the Used By view

To display the Used By view, click the Used By tab on the Call Flows window. The Call Flows window displays the call flows defined for the current application.

The following table describes the various elements of the Call Flows window in the Used By tab view.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display pane</td>
<td>Displays the call flows currently defined for the application.</td>
</tr>
<tr>
<td>☐</td>
<td>At the primary level, signifies that the call flow is defined, but is not used, by any other call flow.</td>
</tr>
<tr>
<td>☐</td>
<td>At a subordinate level, signifies that the call flow uses the call flow in the level above.</td>
</tr>
<tr>
<td>☐</td>
<td>Indicates that the call flow is used by other call flows, but that the call flows using it are hidden (collapsed view).</td>
</tr>
<tr>
<td>☐</td>
<td>Indicates that the call flow is used by other call flows, which are displayed at a secondary level underneath (expanded view)</td>
</tr>
<tr>
<td>Note:</td>
<td>If one or more references to the subordinate call flow are later deleted, IVR Designer does not renumber the remaining references to it.</td>
</tr>
</tbody>
</table>

Expanding a call flow used by other call flows

To expand a call flow used by other call flows, double-click a call flow containing the  ☐  icons.

The call flow expands to display its contents, that is, the parent call flows that use it. At the same time, the design pad displays the used call flow, that is, the one you double-clicked on.

To locate a call flow in the parent call flow, double-click the call flow marked with the  ☐ icon.
Collapsing a call flow used by other call flows

To collapse a call flow used by other call flows, double-click a call flow containing the icons.

The call flow collapses to hide its contents, that is, the parent call flows that use it. At the same time, the design pad displays the selected call flow.

Dragging a call flow to create a copy

You can use the Call Flow Inventory tool to create a copy of a call flow by dragging the call flow onto the design pad.

To create a copy of a call flow by dragging the call flow onto the design pad:

1. Click the name of the call flow you want to copy, and drag it out of the Call Flow Inventory tool window.

2. Place and drop the call flow where you want it on the design pad, either on an open branch or above or below an occupied branch.

   IVR Designer places the call flow much as it would any other node object, with the following exceptions:
   
   — When you drag the call flow over the design pad, the cursor changes to the following:
     
     ![Cursor](image)

     When the cursor has this appearance, you can drop the call flow.

   — When you drag the call flow over an occupied branch or other invalid drop area, the cursor changes to the following:
     
     ![Cursor](image)

     When the cursor has this appearance, you cannot drop the call flow. If you try to drop the call flow, IVR Designer cancels the process without copying the call flow.

Right-click menu options for the Call Flow Inventory tool

The following table describes the right-click menu options for the Call Flow Inventory tool.

Note:
The right-click menu appears only when you right-click the name of a call flow.
The Globals Manager

Globals are alterable resources, such as prompts and variables, that can be accessed from anywhere within an application. For more information about global resources, see Globals on page 171.

By default, all global resources are organized in a single display called the **Globals Manager**. Individual resource managers can, however, either be docked (attached) to or undocked (separated) from the Globals Manager. The following is a list of the resource managers available through the Globals Manager.

- Variables
- Phrases
- Prompts
- VoiceXML Grammars

**Opening the Globals Manager**

To open the Globals Manager, from the **View** menu, select **Globals**.

Each type of global resource has its own window, where the resource elements and their attributes are displayed.

**Globals Manager window elements**

The following table describes the elements on the Global Manager window.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Displays the selected call flow in the design pad, where you can edit it.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the selected call flow from the application.</td>
</tr>
<tr>
<td></td>
<td>You can delete a call flow only if it does not use and is not used by another call flow. The [Main] call flow cannot be deleted.</td>
</tr>
<tr>
<td>Expand</td>
<td>Expands the selected call flow and displays its contents.</td>
</tr>
<tr>
<td>Collapse</td>
<td>Collapses the selected call flow and hides its contents.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Expands all call flows and displays their contents.</td>
</tr>
<tr>
<td>Collapse All</td>
<td>Collapses all call flows and hides their contents.</td>
</tr>
</tbody>
</table>
## Creating a voice response application

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title bar</td>
<td>Located at the top of the window. Contains the buttons for minimizing, maximizing, restoring, and closing the Globals Manager window.</td>
</tr>
<tr>
<td>Control bar</td>
<td>Located immediately below the title bar. Contains the tabstops, Show More button, and Show Less button (see the next three elements).</td>
</tr>
<tr>
<td>Show More icon</td>
<td>Expands your view of the Globals Manager to the next tabstop (see below). If the last tabstop has been reached or exceeded, clicking this icon has no effect.</td>
</tr>
<tr>
<td>Show Less icon</td>
<td>Collapses your view of the Globals Manager to the next tabstop (see below). If the last tabstop has been reached or exceeded, clicking this icon has no effect.</td>
</tr>
<tr>
<td>Tabstop</td>
<td>Controls how far the Globals Manager window expands or collapses when the appropriate button is clicked. Can be dragged to any position in the control bar.</td>
</tr>
<tr>
<td>Resource manager name bar</td>
<td>Used to identify the individual global resource managers. Is distinguishable from other fields in a globals window by its magenta color.</td>
</tr>
<tr>
<td>Resource manager header bar</td>
<td>Located adjacent to the name bar (see above). Identifies the data columns within each global resource manager.</td>
</tr>
<tr>
<td>Data displays</td>
<td>Fields located underneath the name bar and header bars. Used to display the appropriate data for each column.</td>
</tr>
</tbody>
</table>

### Maneuvering in the Globals Manager

You can expand or collapse the entire **Globals Manager** by clicking on the **Show More** or **Show Less** buttons at the top of the window.

There are also tabstops at the top of the Globals Manager window. You can set these tabstops to control how far the Globals Manager expands or collapses. Move these tabstops where you want them by clicking-and-dragging them horizontally within the control bar. For more information about using tabstops, see **More about tabstops** on page 81.

You can open or close a particular global resource manager by double-clicking on the resource manager’s name bar. You can also expand or collapse a resource manager by clicking-and-dragging the resource manager’s name bar up or down.

If a global resource manager is between two other resource managers, you are limited in how far up or down you can move that resource manager’s name bar. You cannot go past the other two resource managers’ name bars, and you cannot move them in tandem.
More about tabstops

With IVR Designer, you can create or delete tabstops and place them where you want them on the Globals Manager control bar. These tabstops control how far the Globals Manager expands or collapses when the appropriate buttons are clicked in the control bar.

To create a tabstop
1. Right-click the control bar somewhere other than on the Show More or Show Less buttons.
2. Click the New Tabstop option.
   IVR Designer displays a new tabstop in the control bar.
3. Drag the tabstop to where you want it.

To delete a tabstop
1. Right-click the tabstop you want to delete.
2. Select the Delete Tabstop option.
   The tabstop disappears.

Using tabstops
The Show More button does not expand the Globals Manager window beyond the last tabstop currently in place. To expand the Globals Manager window beyond the last tabstop:

1. Resize the window by clicking-and-dragging the window edge (as you would any other secondary window).
2. Move an existing tabstop to the desired new location or create another tabstop and place it further out.

No tabstops are visible until you expand the Globals Manager to the last one. To move or set all tabstops, you must do one of the following:

- Expand the Globals Manager as far as you can using the Show More button.
- Maximize the Globals Manager window.

You may want to set the following:
- A first tabstop, so that the Globals Manager is displayed with only the first column with the resource manager name visible
Creating a voice response application

- Other tabstops, so that you can see the desired amount of identifying information for each resource manager
- The last tabstop, so that the Globals Manager expands to the entire width of your screen

Docking and undocking resource managers

Docking a Resource Manager

To dock a resource manager, drag the resource manager's name bar until the name bar is inside the Globals Manager. As you drag the resource manager into the Globals Manager area, the cursor changes to an arrow with a small white box. Release the button. The resource manager reattaches itself to the bottom of the Globals Manager. Therefore, if you want a resource manager to be placed at the top of theGlobals Manager, you must dock it first.

Note:
Do not attempt to dock a resource manager by dragging it by its title bar. It will not dock to the Globals Manager unless you drag it using the name bar.

Undocking a Resource Manager

To undock a resource manager, drag the resource manager's name bar away from the Globals Manager. As you drag the resource manager away from the Globals Manager, a gray box with a solid black outline is displayed. Release the button wherever you want to place the resource manager. You can then treat the resource manager as a separate entity. You can open, close, move, and resize it, regardless of the status of the Globals Manager.

Right-click menu options forGlobals Manager

The right-click menu options that appear in theGlobals Manager depend on where you right-click. The following table describes the menu options available for each location.

<table>
<thead>
<tr>
<th>Location</th>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the control bar</td>
<td>New Tabstop</td>
<td>Creates a new tabstop and places it on the control bar.</td>
</tr>
</tbody>
</table>
Working with variables

Variables in IVR Designer refer to application-specific information holders. They may be used to collect and hold data, act as counters, or make use of system information. Variables are grouped together and accessed through the Variables Manager. Once created, variables can be used as you would use them in any other application-development environment. When variables are set up properly, you can perform arithmetic operations on them, convert them to other types of variables, and compare them with other variables. For more information about variables, see System variable descriptions on page 380. For practice using variables, see Using variables and speech in VoiceXML applications on page 247.

The Variables Manager

The Variables Manager displays all variables defined for the current application, along with their attributes.

From the Variables Manager, you can:

- Use system variables that come as part of the IVR Designer package.
- Create new variables.
- Rename custom variables.
Creating a voice response application

- Set or edit variable attributes.
- Use the right-click menu to perform a variety of other operations on variables.

Opening the Variables Manager

To open the Variables Manager, do one of the following:

- From the View menu, select Variables.
- If the Variables Manager is docked in the the Globals Manager, open the Globals Manager and locate the Variables Manager.

The Variables Manager window elements

The Variables Manager window elements describes the columns on the Variables Manager window and the Spoken As options.

Name column

The Name column is located underneath the Variables Manager name bar. It displays the names of all variables defined for the current application in alphabetic order. Each variable name is enclosed in square brackets [ ].

Type column

The Type chosen for a variable influences several of its other attributes. The six basic variable types, with their default values and accepted input, are listed in the following table. The options in the following table are for US English only.

<table>
<thead>
<tr>
<th>Type</th>
<th>Spoken As</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td></td>
<td></td>
<td>Accepts as input all standard ASCII characters or double-byte data, including special characters and punctuation. (The extended ASCII character set is not supported.)</td>
</tr>
<tr>
<td>Currency</td>
<td></td>
<td></td>
<td>Accepts as input numerals (0-9), a dollar sign, a single leading minus sign, and a decimal point.</td>
</tr>
<tr>
<td>Date</td>
<td>date:ymd</td>
<td>Today's date</td>
<td>Accepts as input numerals (0-9) in one of the following formats:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yyyyymmdd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• yymmdd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• mmdd</td>
</tr>
</tbody>
</table>
### Working with variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Accepts as input numerals (0-9) and a leading minus sign.</td>
</tr>
<tr>
<td>Real</td>
<td>Accepts as input numerals (0-9), a leading minus sign, and a decimal point.</td>
</tr>
<tr>
<td>Time</td>
<td>The current time (24-hour clock)</td>
</tr>
</tbody>
</table>

#### Spoken As column

For system variables (System column displays true), you cannot change the **Spoken As** value. For other variables (System column displays false), you can specify how the contents of a variable is spoken by selecting one of the values in the following table:

<table>
<thead>
<tr>
<th>Spoken As value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(none)</td>
<td>Sets the <strong>Spoken As</strong> column value to null or blank.</td>
</tr>
<tr>
<td>acronym</td>
<td>Speaks the variable as individual characters.</td>
</tr>
<tr>
<td>address</td>
<td>Speaks the variable as a postal address.</td>
</tr>
<tr>
<td>currency</td>
<td>Speaks the variable as currency.</td>
</tr>
<tr>
<td>date: ymd</td>
<td>Speaks the variable as a date using the <code>yyyymmdd</code> format.</td>
</tr>
<tr>
<td>duration: hms</td>
<td>Speaks the variable as a time of day using the <code>hhmmss</code> format.</td>
</tr>
<tr>
<td>duration: hm</td>
<td>Speaks the variable as a time of day using the <code>hhmm</code> format.</td>
</tr>
<tr>
<td>duration: ms</td>
<td>Speaks the variable as a temporal duration using the <code>mmss</code> format.</td>
</tr>
<tr>
<td>duration: h</td>
<td>Speaks the variable as a temporal duration using the <code>hh</code> format.</td>
</tr>
<tr>
<td>duration: m</td>
<td>Speaks the variable as a temporal duration using the <code>mm</code> format.</td>
</tr>
<tr>
<td>duration: s</td>
<td>Speaks the variable as a temporal duration using the <code>ss</code> format.</td>
</tr>
<tr>
<td>measure</td>
<td>Speaks the variable as a measurement, for example, one foot, or two miles.</td>
</tr>
<tr>
<td>name</td>
<td>Speaks the variable as a proper name of a person, organization, etc.</td>
</tr>
<tr>
<td>net: email</td>
<td>Speaks the variable as an internet email handle.</td>
</tr>
<tr>
<td>net: uri</td>
<td>Speaks the variable as an internet Uniform Resource Indicator (URI).</td>
</tr>
<tr>
<td>number</td>
<td>Speaks the variable as a number.</td>
</tr>
<tr>
<td>number: digits</td>
<td>Speaks the variable as individual digits.</td>
</tr>
<tr>
<td>number: ordinal</td>
<td>Speaks the variable as an ordinal.</td>
</tr>
<tr>
<td>telephone</td>
<td>Speaks the variable as a telephone number.</td>
</tr>
<tr>
<td>time: hms</td>
<td>Speaks the variable as a time of day using the <code>hhmmss</code> format (24-hour clock).</td>
</tr>
</tbody>
</table>
Creating a voice response application

Default column
For the default values, the acceptable input is described above. The input can be single-or double-byte text.

System column
If the system displays the word true in this column, the variable is defined as a system variable.

Creating New (Custom) Variables

To create a new (custom) variable:

1. Open the Variables Manager.
2. Right-click the Variables Manager somewhere other than on the name bar.
3. Select New... from the drop-down menu.
   IVR Designer displays a dialog box and prompts you for a name for the new variable.
4. Enter the name you want for the variable in the space provided.

   When naming variables, keep the following in mind:
   — Names can be any length. As a matter of practicality, however, you should try to keep variable names as short as possible.
   — Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the variable name.
   — Variable names are case sensitive. IVR Designer preserves cases as you assign them when naming the variable. The one exception is that the first letter of a name is always converted to upper case, regardless of how you type it in. For example, if you typed in new_Var as the variable name, IVR Designer converts it to New_Var when creating the variable.

   Although IVR Designer preserves cases when saving names and when transferring them to the target system, you cannot have two names with the same letters. If, for example, you have a variable called New_Var, you cannot also create a variable called NEW_VAR.

   — Variable names should begin only with a letter. IVR Designer ignores and deletes all other initial characters when creating the variable name.

   You may use numbers or the underscore ( _ ) symbol after the initial character of a variable name.
5. Click **OK**.
   
The new variable appears in the Variables Manager.

6. Set the attributes for the new variable as desired. For the procedure, see Setting or editing a variable's attributes on page 88.

---

**Renaming custom variables**

**Note:**

You can rename only custom variables. System variables cannot be renamed.

To rename a custom variable:

1. Open the Variables Manager.
2. Right-click the name of the variable whose name you want to change.
3. Select **Rename...** from the menu.
   
   IVR Designer displays a dialog box and prompts you for a new name for the variable.

4. Enter the new name you want for the variable in the space provided.
   
   When naming variables, keep the following in mind:
   
   — Names can be any length. As a matter of practicality, however, you should try to keep variable names as short as possible.
   
   — Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the variable name.
   
   — Variable names are case sensitive. IVR Designer preserves cases as you assign them when naming the variable. The one exception is that the first letter of a name is always converted to upper case, regardless of how you type it in. For example, if you typed in new_Var as the variable name, IVR Designer converts it to New_Var when creating the variable.
   
   Although IVR Designer preserves cases when saving names and when transferring them to the target system, you cannot have two names with the same letters. If, for example, you have a variable called **New_Var**, you cannot also create a variable called **NEW_VAR**.
   
   — Variable names should begin only with a letter. IVR Designer ignores and deletes all other initial characters when creating the variable name.
   
   You may use numbers or the underscore ( _ ) symbol after the initial character of a variable name.

5. Click **OK**.
Creating a voice response application

The name of the variable changes in the Variables Manager and anywhere it is referenced within the application.

---

Setting or editing a variable's attributes

**Note:**
Some system variable attributes (such as **Type**) cannot be edited.

To set or edit a variable’s attributes:

1. Open the **Variables Manager**.
2. Click the attribute you want to set or edit.
3. Do one of the following:
   — Enter the value you want for the attribute.
   — Select the value you want for the attribute from the drop-down menu, where available.

---

Right-click menu options for the Variables Manager

The following table describes the right-click menu options available in the **Variables Manager**.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New...</td>
<td>Lets you create a new (custom) variable.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the selected variable from the Variables Manager.</td>
</tr>
<tr>
<td>Rename...</td>
<td>Lets you rename the selected custom variable without altering its attributes.</td>
</tr>
<tr>
<td>Find Uses of</td>
<td>Opens the <strong>Find</strong> window, searches for uses of the selected variable, and presents the results of the search in the Find window <strong>Results</strong> field.</td>
</tr>
<tr>
<td>Stay on Top</td>
<td>(Available only when the Variables Manager is undocked.) Toggles on and off. When checked (toggled on), causes the Variables Manager window to remain on top of other open IVR Designer windows. If you have more than one resource manager window toggled to <strong>Stay on Top</strong>, the window actually on top will be the last window you clicked. All windows toggled to <strong>Stay on Top</strong>, however, will stay on top of any other IVR Designer windows that are open.</td>
</tr>
</tbody>
</table>
Prompts are used in IVR Designer as components of the Interactive Voice Response (IVR) nodes. Prompts are used to instruct callers, to get callers to select choices, and to obtain information from callers. For practice using prompts, see Prompts, Response, and Confirmation in VoiceXML on page 235 and Editing prompts in VoiceXML on page 259.

Prompts are most commonly composed of phrases. They may also include variables, phrase breaks, or controls.

Prompts are grouped together and displayed in the Prompts Manager.

The Prompts Manager

The Prompts Manager is used to perform many of the actions related to creating and using prompts in IVR Designer applications. From the Prompts Manager, you can:

- Create new prompts
- Rename existing prompts
- Edit existing prompts
- Use the right-click menu to perform a variety of other operations on prompts

The Prompts Manager displays the names and the first line of data for all prompts defined for the current application.

Opening the Prompts Manager

To open the Prompts Manager, do one of the following:

- From the View menu, select Prompts.
- If the Prompts Manager is docked to the Globals Manager, open the Globals Manager and locate the Prompts Manager.

Prompts Manager window elements

The left column (Prompts) of the Prompts Manager window lists all prompts defined for the current application by name in alphabetic order. The right column (Value) displays the phrase text or other elements, such as variables, assigned to the prompts.

Creating new prompts

There are two ways you can create a new prompt in IVR Designer:
Creating a voice response application

- Using the Prompts Manager
- Using the Node Inspector

Using the Prompts Manager to create new prompts

To create a prompt using the Prompts Manager:

1. Open the Prompts Manager.
2. Right-click the Prompts Manager grid anywhere other than on the name bar.
3. Select New... from the menu.
   IVR Designer displays a dialog box and asks you to name the new prompt.
4. Enter a name for the new prompt in the space provided.
   When naming prompts, keep the following in mind:
   — Names can be any length. However, keep prompt names as short as possible.
   — Do not use spaces in prompt names. IVR Designer ignores and deletes spaces when creating the prompt name.
   — Prompt names are case sensitive. IVR Designer preserves cases as you assign them when naming the prompt. The one exception is that the first letter of a name is always converted to upper case, regardless of how you type it in. For example, if you typed in new_Prompt as the prompt name, IVR Designer converts it to New_Prompt when creating the prompt.
   
   **Note:**
   Although IVR Designer preserves cases when saving names and transfers them to the target system with cases preserved, you cannot have two names with the same letters. For example, if you have a prompt called New_Prompt, you cannot also create a prompt called NEW_PROMPT.
   — Begin prompt names with alphanumeric characters only. IVR Designer ignores and deletes all other initial characters when creating the prompt name.
   
   **Note:**
   You can use the underscore (_) symbol any place within the prompt name after the initial character.

5. Click OK.
   IVR Designer displays the Prompt Editor window.

6. Define the prompt attributes as you want in the Prompt Editor.

7. Click OK.
IVR Designer creates the new prompt and places it in the Prompts Manager in alphabetical order by prompt name.

If you typed in text as part of the prompt definition, IVR Designer displays the New Phrase Assignments window.

Using the Node Inspector to create new prompts

To create a prompt using the Node Inspector:

1. Open the Node Inspector.
2. Double-click the line that contains the prompt you want to create.
   - IVR Designer displays a dialog box and asks you to name the new prompt.
3. Enter a name for the new prompt in the space provided.
   - When naming prompts, keep the following in mind:
     - Names can be any length. However, keep prompt names as short as possible.
     - Do not use spaces in prompt names. IVR Designer ignores and deletes spaces when creating the prompt name.
     - Prompt names are case sensitive. IVR Designer preserves cases as you assign them when naming the prompt. The one exception is that the first letter of a name is always converted to upper case, regardless of how you type it in. For example, if you typed in new_Prompt as the prompt name, IVR Designer converts it to New_Prompt when creating the prompt.
       - **Note:** Although IVR Designer preserves cases when saving names and transfers them to the target system with cases preserved, you cannot have two names with the same letters. For example, if you have a prompt called New_Prompt, you cannot also create a prompt called NEW_PROMPT.
     - Begin prompt names with alphanumeric characters only. IVR Designer ignores and deletes all other initial characters when creating the prompt name.
       - **Note:** You can use the underscore (_) symbol any place within the prompt name after the initial character.
4. Click OK.
   - IVR Designer displays the Prompt Editor window.
5. Define the prompt attributes in the Prompt Editor.
6. Click OK.
Creating a voice response application

IVR Designer creates the new prompt and places it in the Prompts Manager in alphabetical order by prompt name.

If you typed in text as part of the prompt definition and if those phrases do not already exist in the Phrases Manager window, IVR Designer displays the New Phrase Assignments window.

Renaming prompts

To rename an existing prompt without altering its contents:

1. Open the Prompts Manager window.
2. Right-click the name of the prompt whose name you want to change.
3. Select Rename... from the menu.
   IVR Designer displays a dialog box and asks you for a new name for the prompt.
4. Enter a new name for the prompt in the space provided.
   When naming prompts, keep the following in mind:
   — Names can be any length. However, keep prompt names as short as possible.
   — Do not use spaces in prompt names. IVR Designer ignores and deletes spaces when creating the prompt name.
   — Prompt names are case sensitive. IVR Designer preserves cases as you assign them when naming the prompt. The one exception is that the first letter of a name is always converted to upper case, regardless of how you type it in. For example, if you typed in new_Prompt as the prompt name, IVR Designer converts it to New_Prompt when creating the prompt.

   Note:
   Although IVR Designer preserves cases when saving names and transfers them to the target system with cases preserved, you cannot have two names with the same letters. For example, if you have a prompt called New_Prompt, you cannot also create a prompt called NEW_PROMPT.
   — Begin prompt names with alphanumerical characters only. IVR Designer ignores and deletes all other initial characters when creating the prompt name.

   Note:
   You can use the underscore (_) symbol any place within the prompt name after the initial character.
5. Click OK.
Working with prompts

The name of the prompt changes in the Prompts Manager window and anywhere the prompt is used or referenced within the application.

Right-click menu options for the Prompts Manager

When you right-click the Prompts Manager window anywhere except the name bar, IVR Designer displays a menu. The following table describes the options available from the Prompts Manager right-click menu.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New...</td>
<td>Lets you create a new prompt.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the selected prompt from the Prompts Manager, if it is not in use.</td>
</tr>
<tr>
<td></td>
<td>If the prompt is used an application, you must first remove it from the application before you can delete it.</td>
</tr>
<tr>
<td>Rename...</td>
<td>Lets you rename an existing prompt.</td>
</tr>
<tr>
<td>Find Uses of</td>
<td>Opens the Find window, searches for uses of the selected prompt, and presents the results of the search in the Find window Results field.</td>
</tr>
<tr>
<td>Stay on Top</td>
<td>Available only when the Prompts Manager is undocked. Toggles on and off.</td>
</tr>
<tr>
<td></td>
<td>When checked (toggled on), it causes the Prompts Manager window to remain on top of other open IVR Designer windows.</td>
</tr>
<tr>
<td></td>
<td>If you have more than one resource manager window toggled to Stay on Top, the window actually on top will be the last window you clicked. All windows toggled to Stay on Top, however, will stay on top of any other IVR Designer windows that are open.</td>
</tr>
<tr>
<td>Edit Prompt...</td>
<td>Opens the Prompt Editor so you can edit the selected prompt.</td>
</tr>
</tbody>
</table>

The Prompt Editor

The Prompt Editor lets you define, edit, and select the attributes and elements of a prompt.

From the Prompt Editor, you can:

- Choose an alternate view of the elements of a prompt
- Create a new phrase within a prompt
- Insert an existing phrase into a prompt
Creating a voice response application

- Edit a phrase in a prompt
- Insert a variable into a prompt
- Insert a phrase break into a prompt
- Insert a control into a prompt
- Delete a prompt element – such as a phrase, variable, or control – from a prompt
- Insert a new line – alternate view only
- Use the right-click menus to perform various other operations on prompt attributes and elements

Opening the Prompt Editor

To open the Prompt Editor:

1. Open the Prompts Manager window.
2. Right-click the name of the prompt you want to edit.
3. Select Edit Prompt... from the menu.

Choosing an Alternate View for the Prompt Text field

When you first open the Prompt Editor window, IVR Designer displays the window like the following example, called here the normal view.
To change to the alternate view of the contents field, click the Change View button.

One advantage of the alternate view is that it tags each element of the prompt. These tags let you see exactly how many elements are used in the prompt and what type they are.

Creating a new phrase within a prompt

To create a new phrase within a prompt, type the new phrase directly in the Prompt Text field.

Note:
You cannot create phrases in the alternate view.

When creating new phrases in the Prompt Editor, keep the following in mind:

- Every time you press the Enter key or select Insert a Phrase Break, you start a new phrase with its own phrase number.
- When you click the Apply or OK button, IVR Designer displays the New Phrase Assignment window and automatically assigns phrase numbers to new phrases.
- Variable names appear enclosed in square brackets ( [ ] ).
- Controls appear enclosed in curly brackets ( { } ).

The New Phrase Assignments window

IVR Designer displays the New Phrase Assignments window when you create a new phrase or edit an existing phrase within a prompt.
Creating a voice response application

When you create a new phrase within a prompt and click either the **Apply** button or the **OK** button, IVR Designer displays the New Phrase Assignments window. The New Phrase Assignments window has four columns. You cannot edit the data in any of these fields. You can only select the phrase number or phrase table you want from the drop-down menus.

**New Phrase Assignments window columns**

The following table describes the columns in the New Phrase Assignments window.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase</td>
<td>From the drop-down menu, select one of the following:</td>
</tr>
<tr>
<td></td>
<td>• [new] – Creates a new phrase and automatically assigns a phrase number to it when you close the window.</td>
</tr>
<tr>
<td></td>
<td>• A phrase number (displayed) – Exchanges any old text with the new or modified text. Any old text is displayed in the <strong>Old Text</strong> column. The text that replaces it is displayed in the <strong>New Text</strong> column.</td>
</tr>
<tr>
<td>Phrase Table</td>
<td>If you are creating a new phrase, select from the drop-down menu the phrase table you want to assign the phrase to.</td>
</tr>
<tr>
<td></td>
<td>To edit or modify an existing phrase, select from the drop-down menu the phrase table that contains the phrase you want to edit or modify. You must do this <em>before</em> selecting the phrase to edit or modify.</td>
</tr>
<tr>
<td>Old Text</td>
<td>If you are editing or modifying an existing phrase, this column displays the text that will be edited or modified.</td>
</tr>
<tr>
<td></td>
<td>If you are editing or modifying a new phrase, this column is empty.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>You cannot alter the contents of this column. This column is for display only.</td>
</tr>
<tr>
<td>New Text</td>
<td>This column displays the text to be assigned to the phrase.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>You cannot alter the contents of this column. This column is for display only.</td>
</tr>
</tbody>
</table>

**Using the New Phrase Assignments window to create a new phrase**

To create the new phrase and assign it a phrase number:

1. Do one of the following:
   - Select **[new]** from the **Phrase** drop-down menu.
   - From the **Phrase** drop-down menu, select the number of the phrase you want to replace.

2. From the **Phrase Table** drop-down menu, select the **Phrase Table** to which you want to assign the new phrase.
3. To confirm your choices and close the dialog box, click the **OK** button.

   IVR Designer displays the new phrase in the Phrases Manager window.

**Inserting an existing phrase into a prompt**

IVR Designer offers two methods for inserting existing phrases into prompts.

**Method A**

1. In the normal view, place the cursor where you want the phrase to go.
2. Do one of the following:
   - Right-click the **Prompt Text** field and select **Insert Phrase**...
   - Press **Control + P**.

   IVR Designer displays the **Insert Phrase** dialog box. This window displays all the phrases that exist for the active phrase table.
3. Highlight the phrase you want from the list.
4. Click **OK**.

   The phrase is inserted into the Prompt Text field at the point where you placed the cursor.

**Method B**

1. Open the **Phrases Manager** window.
2. Do one of the following:
   - Place the cursor where you want the phrase to go in the normal view of the Prompt Editor. Drag-and-drop the selected phrase using the phrase number field to drag from the **Phrases Manager** window to anywhere in the **Prompt Text** field.
   - Select the alternate view of the Prompt Editor. Drag-and-drop the selected phrase, using the phrase number to drag from the **Phrases Manager** window to the line of the Prompt Editor **Prompt Text** field where you want the phrase to go.

   **Note:**
   If you drop the phrase on top of a line already containing an element in the alternate view, the new phrase replaces what is already on that line. To insert a phrase before the last line, first insert a new line in the Prompt Text field where you want the phrase to go. For the procedure, see **Inserting a new line (alternate view on page 102)**.
Creating a voice response application

Editing a phrase in a prompt

You can edit a phrase in IVR Designer much the same way you would edit text in any basic word processor. You can change phrase text only in the normal view of the Prompt Text field.

Select the text you want to edit by highlighting it. Once the text is highlighted, the following functions on the right-click menu become active:

- Cut (Control + X)
- Copy (Control + C)
- Paste (Control + P)
- Delete
- Select All (Control + A)

In addition to typing in new text, you can use any of the standard Windows functions listed above.

If you make changes to phrase text in the Phrases Manager, those changes are automatically reflected in the Prompt Editor.

If you make changes to phrase text in the Prompt Editor, IVR Designer treats the phrase as a new phrase and assigns a new phrase number to it when you click the Apply or OK button. The original phrase and phrase number are not changed or deleted. Finish editing the phrase in the New Phrase Assignments window. If you do not want to keep the original phrase, use the Phrases Manager to delete it.

Inserting a variable into a prompt

Any variable can be used as part of a prompt. Variables can be used to speak stored information to a caller.

IVR Designer offers two methods for inserting a variable into a prompt:

Method A
1. In the normal view, place the cursor where you want the variable to go.
2. Do one of the following:
   - Right-click the Prompt Text field and select Insert Variable...
   - Press Control + B
   - Type a left square bracket character ( [ )
IVR Designer displays the **Insert Variable** dialog box.

3. Highlight the variable you want from the list.

4. Click **OK**.

The variable is inserted into the **Prompt Text** field at the point where you placed the cursor.

**Method B**

1. Open the **Variables Manager** window.

2. Do one of the following:

   — In the normal view of the Prompt Editor, place the cursor where you want the variable to go. Drag-and-drop the selected variable from the **Variables Manager** to anywhere in the **Prompt Text** field.

   — Select the alternate view of the Prompt Editor. Drag-and-drop the selected variable from the **Variables Manager** to the line of the Prompt Editor **Prompt Text** field where you want the variable to go.

   **Note:**

   If you drop the variable on top of a line already containing an element in the alternate view, the variable replaces what is already on that line. To insert a variable before the last line, first, use the right-click menu to insert a new line in the Prompt Text field where you want the variable to go. For the procedure, see **Inserting a new line (alternate view on page 102)**.

**Inserting a phrase break**

A **Phrase Break** signals the end of a phrase. You can use phrase breaks to divide long phrases into shorter phrases or to start a new line in the Prompt Editor to add another component.

To insert a phrase break in the Prompt Editor, do one of the following:

- Place the cursor where you want the break, and then press the **Enter** key.
- Right-click the normal view of the Prompt Editor, and then select **Insert Break**...
- Press **Control + I** in the normal view of the Prompt Editor.

**Inserting a Control into a prompt**

**Controls** are used to:
Creating a voice response application

- Insert a pause (silence) of a specific length
- Specify how Text-to-Speech (TTS) speaks the phrase text

Types of controls
There are two types of prompt controls in IVR Designer:
- Pauses
- Text-to-Speech Options

Pauses
Pauses are programmed intervals of silence, typically used to make TTS (Text-to-Speech) responses sound more natural, though they may also be used with recorded speech. The following pauses are available:

- **break_small** — A small pause in the speech output.
- **break_medium** — A medium pause in the speech output.
- **break_large** — A large pause in the speech output.
- **break_50msec** — A 50 millisecond pause in the speech output.
- **break_100msec** — A 100 millisecond pause in the speech output.
- **break_200msec** — A 200 millisecond pause in the speech output.
- **break_300msec** — A 300 millisecond pause in the speech output.
- **break_400msec** — A 400 millisecond pause in the speech output.
- **break_500msec** — A 500 millisecond pause in the speech output.

Text-to-Speech options
Text-to-Speech offers a variety of options for how it may be used. The following table describes these options.

**Note:**
Text-to-Speech (TTS) is available only if you have purchased it as an option on your CONVERSANT or Avaya IR system. In many cases, you will probably find it preferable to use recorded speech.

<table>
<thead>
<tr>
<th>TTS option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tts_acronym</td>
<td>Speaks the text as individual characters.</td>
</tr>
</tbody>
</table>
Inserting the control

To insert a control into a prompt:

1. In the normal view, place the cursor where you want the control to go.
2. Do one of the following:
   - Right-click the Prompt Text field and select Insert Control...
   - Press Control + L.
   - Type a left curly bracket ( { )

IVR Designer displays the Insert Control dialog box. The options that appear in this window depend upon the language selected for the application. Not all silence or pause options are available for all languages.

3. Highlight the control you want from the list.
4. Click OK.
Creating a voice response application

IVR Designer inserts the control into the Prompt Text field at the point where you placed the cursor.

Deleting a prompt element

IVR Designer offers several options for deleting elements from a prompt:

Note:
Delete actions cannot be undone in the Prompt Editor, and no warning is given.

- In the normal view:
  - Highlight the elements you want to delete, and then press the Backspace key.
  - Backspace over the elements you want to delete.
- In the alternate view:
  - Right-click the line containing the element, and then select Delete Line.
  - Highlight the line containing the element, and then press the Delete key.

Inserting a new line (alternate view)

To insert a new line in the Prompt Text field of the Prompt Editor (alternate view):

1. Highlight the line below where you want the new line to go.
   Note:
   Inserting a line places it just above whatever line is highlighted.

2. Do one of the following:
   - Press the Insert key.
   - Right-click the line and select Insert Line.

Right-click menu options for the Prompt Editor

Which right-click menu you see in the Prompt Editor depends on whether you are in the normal view or the alternate view.

Normal view

The following table describes the right-click menu options for the normal view in the Prompt Editor window.
### Working with prompts

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td>Removes the element from the Prompt Text field and places it on the clipboard. This option is active only if something has already been selected in the Prompt Text field.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies the element to the clipboard, leaving the original where it is. This option is active only if something has already been selected in the Prompt Text field.</td>
</tr>
<tr>
<td>Paste</td>
<td>Pastes whatever is on the clipboard to the cursor's location.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the element from the Prompt Editor. This option is active only if something has already been selected in the Prompt Text field. The Delete action cannot be undone and does not copy the element to the system clipboard.</td>
</tr>
<tr>
<td>Select All</td>
<td>Selects or highlights all elements in the Prompt Text field.</td>
</tr>
<tr>
<td>Insert Variable</td>
<td>Inserts a variable at the cursor's location.</td>
</tr>
<tr>
<td>Insert Phrase</td>
<td>Inserts a phrase at the cursor's location.</td>
</tr>
<tr>
<td>Insert Phrase Break</td>
<td>Inserts a phrase break at the cursor's location.</td>
</tr>
<tr>
<td>Insert Control</td>
<td>Inserts a control at the cursor's location.</td>
</tr>
</tbody>
</table>

### Alternate view

The following table describes the right-click menu options for the alternate view in the Prompt Editor.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Line</td>
<td>Inserts a new line just above the currently selected (highlighted) line.</td>
</tr>
<tr>
<td>Add Line</td>
<td>Adds a line at the end of the element list.</td>
</tr>
<tr>
<td>Delete Line</td>
<td>Deletes the currently selected (highlighted) line. The Delete action cannot be undone and does not copy the element to the system clipboard.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Moves the currently selected line up one line, exchanging places with the line above it.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Moves the currently selected line down one line, exchanging places with the line below it.</td>
</tr>
</tbody>
</table>
Working with phrases

In IVR Designer, phrases are bits of spoken or written speech, often used to instruct customers or prompt them for information. Individual phrases are grouped together in phrase tables and used in prompts.

By default, IVR Designer includes a custom phrase table, labeled simply PhraseTable.

You can create a maximum of eight other phrase tables per application.

The Phrases Manager

The Phrases Manager displays the contents of one phrase table at a time.

From the Phrase Manager you can:

- Select which phrase table to display
- Create new phrases
- Edit existing phrases
- Use the right-click menu to perform many other operations on phrases and phrase tables

For practice using the Phrases Manager, see Editing phrases in VoiceXML on page 256.

Opening the Phrases Manager

To open the Phrases Manager, do one of the following:

- From the View menu, select Phrases.
- If the Phrases Manager is docked to the Globals Manager, open the Globals Manager and locate the Phrases Manager.

Phrases Manager window elements

The following table describes the columns in the Phrases Manager window.

<table>
<thead>
<tr>
<th>Column heading</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase number</td>
<td>Displays the phrase numbers of all phrases defined for the current application.</td>
</tr>
<tr>
<td>Recorded</td>
<td>An X in this column indicates that an audio file exists for the corresponding phrase (on the PC).</td>
</tr>
</tbody>
</table>
Outdated | An X in this column indicates that the phrase text has been altered since a recording was made for it or that no recording has been made for the phrase (default).
---|---
Phrase Text | Displays the text assigned to each phrase. This text (the first 50 characters) becomes the phrase tag, which is used by the target system to identify the phrase.

### Selecting a phrase table to display

The Phrases Manager displays the phrases of only one phrase table at a time. If you want to view the contents of another phrase table, you must first select it from the right-click menu.

To select which phrase table to display:

1. Open the **Phrases Manager**.
2. Right-click anywhere in the Phrases Manager other than the name bar.

IVR Designer displays a right-click menu. Each phrase table for the current application is listed at the bottom enclosed in square brackets ( [ ] ). The currently displayed phrase table is designated by a checkmark.

To select another phrase table for display, click the name of the phrase table.

### Creating new phrases

Most of the time, you want to create your own custom phrases for each application you develop in IVR Designer. This is usually done from the **Phrases Manager**.

When you create phrase text for a phrase, you do not necessarily create the audio file to play it with. Although the audio file is not required, the phrase text is required as the first 50 characters become the phrase tag used to identify the phrase on the target system. You can also create the audio files for your phrases in IVR Designer.

To create a new phrase:

1. Open the **Phrases Manager**.
2. Right-click the Phrases Manager somewhere other than on the name bar.
3. Select **New**...

IVR Designer displays a new line in the Phrases Manager and opens the **Phrase Editor**. A phrase number is automatically assigned to the new phrase.

4. In the **Phrase Text** field, enter the phrase text you want for the phrase.
5. (Optional) Record the phrase using the **Recording** box controls, or import a prerecorded *.wav file using the **Import** button.
Creating a voice response application

To record and play back recorded speech, you must have a sound card installed on your computer. You must also have speakers or headphones and a microphone.

At present, IVR Designer supports only the *.wav format for audio files.
To edit the *.wav file, click the Edit button.

6. Click OK.

Editing existing phrases

To edit an existing phrase:

1. Open the Phrases Manager.

2. Do one of the following:
   — Double-click the row containing the phrase you wish to edit.
   — Right-click the row containing the phrase you wish to edit, and then select Edit Phrase...

   IVR Designer displays the Phrase Editor window.

3. Edit the phrase text in the Phrase Text field.

4. (Optional) Record the phrase using the Recording box controls, or import a prerecorded *.wav file using the Import button.

   To record and play back recorded speech, you must have a sound card installed on your computer. You must also have speakers or headphones and a microphone.

   At present, IVR Designer supports only the *.wav format for audio files.
   To edit the *.wav file, click the Edit button.

5. Click OK.

Right-click menu options for the Phrases Manager

When you right-click the Phrases Manager window anywhere except the name bar, IVR Designer displays a menu.

The following table describes the menu options on the Phrases Manager right-click menu.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New…</td>
<td>Creates a new phrase, automatically assigns a phrase number to it, and opens the <strong>Phrase Editor</strong> window.*</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the contents of the selected phrase from the table (both the phrase text and, if one exists, the *.wav file, but <strong>not</strong> the phrase number).</td>
</tr>
<tr>
<td>Rename…</td>
<td>Unavailable option in the Phrases Manager.</td>
</tr>
<tr>
<td>Find Uses of</td>
<td>Opens the <strong>Find</strong> window, searches for Uses Of the selected phrase, and presents the results of the search in the Find window <strong>Results</strong> field.</td>
</tr>
<tr>
<td>Stay on Top</td>
<td>(Available only when the Phrases Manager is undocked.) Toggles on and off. When checked (toggled on), it causes the Phrases Manager window to remain on top of other open IVR Designer windows. If you have more than one resource manager window toggled to <strong>Stay on Top</strong>, the window actually on top will be the last window you clicked. All windows toggled to <strong>Stay on Top</strong>, however, will stay on top of any other IVR Designer windows that are open.</td>
</tr>
<tr>
<td>Edit Phrase…</td>
<td>Opens the <strong>Phrase Editor</strong> so you can edit the selected phrase.</td>
</tr>
<tr>
<td>Display Alphabetically by Tag</td>
<td>Toggles on and off. When checked (toggled on), it lists all phrases in the table alphabetically by phrase tags.</td>
</tr>
<tr>
<td>Fill Phrases with Default…</td>
<td>Opens the <strong>Create Default Phrases</strong> dialog box, which lets you create a user-specified number of new phrases. It also fills each new phrase with user-defined default phrase text.</td>
</tr>
<tr>
<td>Import Phrases from Text…</td>
<td>Opens the <strong>Phrase Text Import</strong> dialog box, which lets you select and import predefined text from sources outside IVR Designer.</td>
</tr>
<tr>
<td>Export Phrases for Spell Check…</td>
<td>Inactive in the current release of IVR Designer (grayed out).</td>
</tr>
<tr>
<td>New Phrase Table…</td>
<td>Opens the <strong>Phrase Table Editor</strong>, which lets you create a new phrase table and define its basic parameters.</td>
</tr>
<tr>
<td>Edit Phrase Table…</td>
<td>Opens the <strong>Phrase Table Editor</strong> and populates it with information about the currently open phrase table, which you can then edit.</td>
</tr>
<tr>
<td>Delete Phrase Table…</td>
<td>Deletes the currently selected phrase table. If any phrases from the table are being referenced by the current application, IVR Designer warns you and instructs you to find and remove the references before you delete the table. <strong>Note:</strong> The Delete Phrase Table action cannot be undone.</td>
</tr>
<tr>
<td>[Phrase Table Names]</td>
<td>Lists the names of all phrase tables currently defined for the application. The currently active phrase table is designated by a checkmark. To select a different table as the active table, click that table’s name.</td>
</tr>
</tbody>
</table>
The Phrase Editor

The Phrase Editor is used to:

- Write or edit phrase text
- Make an audio recording of the phrase
- Import an audio file
- Edit an audio file

Opening the Phrase Editor

To open the Phrase Editor:

1. Open the Phrases Manager.
2. Do one of the following:
   - Double-click any phrase table line entry.
   - Right-click anywhere in the Phrases Manager window other than the name bar; then from the menu, do one of the following:
     - Select New...
     - Select Edit Phrase...

IVR Designer displays the Phrase Editor window.

Phrase Editor window elements

The following table describes the elements of the Phrase Editor window.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
</table>
| Phrase Text field| Type or import the text for the phrase you are creating here. Text in this field can be edited much like text in any basic word processor.  
Text created here appears in the Phrase Text column of the Phrases Manager. This text (the first 50 characters) becomes the phrase tag, which is used by the target system to identify the phrase. |
| Play button      | Plays digitally recorded audio files. These audio files must be in *.wav format.                                                             |
| Pause button     | Pauses the playback of a recorded phrase.                                                                                                   
The Pause button is currently operable only in Playback mode. |
### Working with phrases

<table>
<thead>
<tr>
<th>Button Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stop button</strong></td>
<td>Stops a recording or playback in progress and resets the recording to its beginning.</td>
</tr>
<tr>
<td><strong>Record button</strong></td>
<td>Digitally records a phrase.</td>
</tr>
<tr>
<td><strong>Import button</strong></td>
<td>Imports a prerecorded audio file from another source. Audio files must be in <em>.</em>.wav format to playback in IVR Designer. Other audio formats are not currently supported.</td>
</tr>
<tr>
<td><strong>Edit button</strong></td>
<td>Edit a recorded phrase. Clicking on this button opens the default audio editing application on your machine. This is typically the Microsoft Sound Recorder, unless you have specified another application. This button is only active when there is a recorded phrase present, either as part of the application or in the clipboard buffer.</td>
</tr>
<tr>
<td><strong>Erase button</strong></td>
<td>Deletes the recording on the system clipboard and resets the Length and Size indicators to 0. This action cannot be undone and does not warn you that you are permanently erasing the recording.</td>
</tr>
<tr>
<td><strong>Out of Date checkbox</strong></td>
<td>When checked, this signifies that changes have been made to the written text since a recording was made for the phrase. If no recording has been made, this box is checked by default. You can override the <strong>Outdated</strong> status.</td>
</tr>
<tr>
<td><strong>Length box</strong></td>
<td>Displays the running time of the recording to the nearest tenth of a second.</td>
</tr>
<tr>
<td><strong>Size box</strong></td>
<td>Displays the size of the recording in numbers of bytes.</td>
</tr>
<tr>
<td><strong>Use the following URI checkbox</strong></td>
<td>When checked, you can enter a URI on page 404 that specifies the location of the phrase. For more information, see Assigning URIs to phrases and phrase tables.</td>
</tr>
<tr>
<td><strong>Previous phrase (Prev) button</strong></td>
<td>Selects the previous phrase in the phrase table for editing without having to close and reopen the Phrase Editor.</td>
</tr>
<tr>
<td><strong>Next phrase (Next) button</strong></td>
<td>Selects the next phrase in the phrase table for editing without having to close and reopen the Phrase Editor.</td>
</tr>
<tr>
<td><strong>OK button</strong></td>
<td>Accepts the edits made to the selected phrases and makes the changes in the application.</td>
</tr>
<tr>
<td><strong>Cancel button</strong></td>
<td>Cancels any changes made to the phrase and closes the Phrase Editor.</td>
</tr>
</tbody>
</table>
Creating a voice response application

Using the Phrase Editor

Writing and editing phrase text
To write or edit phrase text:

1. Click anywhere inside the Phrase Text field.
2. Add, change, or delete the phrase text as desired.
3. (Optional) Click the Use the following URI checkbox, and enter a Uniform Resource Identifier that specifies the location of the phrase. For information about assigning URIs, see Assigning URIs to Phrases and Phrase Tables.
4. Click OK.

Recording phrases
To make an audio recording of a phrase:

   Note:
   To make or edit audio recordings, you must have a sound card, speakers or headphones, and a microphone properly connected to and configured on your computer.

1. Click the Record button.
2. Use the microphone to record the phrase.
3. Click the Stop button when you are finished recording.

Importing a phrase recording
To import a prerecorded audio file:

1. Make sure the audio file you want to import is in *.wav format and has been recorded at the proper sampling rate. The *.wav format is currently the only audio format supported by IVR Designer.
2. Click the Import button.
   If an audio file already exists for the phrase, IVR Designer displays a warning message asking if you want to overwrite the existing file.
3. Select the *.wav file you want from the Open dialog box.
4. Click Open.

Editing phrase recordings
To edit an audio (*.wav) file:
1. Click the **Edit** button.
   The default audio editing application for your computer opens.
2. Use the audio editing application to edit the file.
3. Make sure you save the edited file using the *.wav file format.
4. Close the audio editing application.

**Assigning URIs**

**Assigning a URI for a specific phrase**

When creating a new phrase or editing an existing phrase, you can enter a Uniform Resource Identifier (URI on page 404) that specifies the location of the phrase. If you have assigned a base URI, you can use either an absolute or a relative URI. If you have not assigned a base URI, you should always use an absolute URI.

An absolute URI has one of the following formats:

- http://folder_name/filename.vis
- https://folder_name/filename.vis

**Note:**
Use this format only if you are using SSL with the URI. For more information about using SSL in VoiceXML applications, see "Using SSL in VoiceXML applications" in the Avaya IR system help. Note also that you cannot use this format for applications used with CONVERSANT systems or Avaya IR systems earlier than Release 2.0.

- ftp://folder_name/audiofilename.wav
- file:///folder_name/recordedspeech.wav

A relative URI has the format /folder_name/audio.vis.

Use the following procedure to assign a URI for a specific phrase:

1. Use the **Phrase Editor** to create a new phrase, or to open an existing phrase.
2. In the **Audio URI** area, make sure the **Use the following URI** check box is checked, and enter the absolute or relative URI.

**Assigning a base URL to a Phrase Table**

The following procedure assigns a base URL to a IVR Designer phrase table.

The URI should have one of the following formats:
Creating a voice response application

- http://folder_name/folder_name
- https://folder_name/folder_name

**Note:**
Use this format only if you are using SSL with the URI. For more information about using SSL in VoiceXML applications, see "Using SSL in VoiceXML applications" in the Avaya IR system help. Note also that you cannot use this format for applications used with CONVERSANT systems or Avaya IR systems earlier than Release 2.0.

- ftp://folder_name/folder_name
- file:///folder_name/folder_name

The base URI that you specify will be prepended to any relative URI that is specified for a specific phrase. For example, if you enter http://audio as the base URI, and you enter /welcome.wav as a relative URI for phrase 5001, the absolute URI for phrase 5001 is http://audio/welcome.wav.

Use the following procedure to specify the base URI for a phrase table:

1. Open the **Phrases Manager**.
2. Right click the Phrases Manager anywhere other than on the name bar, and select **Edit Phrase Table**.
3. In the Phrase Table Editor, select **Recorded Phrases** in the **For Code Generation, Phrases Use** field.
4. In the **For Code Generation, Phrases Use** field, enter a base URI in the text box at the bottom of the field.

**Using https:// in URIs**

When using SSL within a VoiceXML application, make sure that your Uniform Resource Identifier (URI) references begin with "https://..." (rather than simply http://...).

You can use https://... for some or all of the URI references in your application. But because using secure URIs (https://...) for references can impair system performance, Avaya recommends that you use it only for those URIs where secure transmission of data is required. For other URIs, it is alright to use http://...

You can also define your XML tags in such a way that only certain tags make use of the https://... URI. For example, you might require that only data passed to a sub-dialog tag be transported using SSL. In this case, the use of the sub-dialog tag might look like the following:

```
<subdialog src="https://...">
```
This would cause any data passed to the <subdialog> tag to use https://... references. All other URI references in the application, then, would use http://...

Audio recording tips

Keep the following in mind when recording phrases:

• When you are finished recording your phrase, check your recording by playing it back. To play a recorded phrase, click the Play button.

• In most cases, you will also want to edit unwanted dead space at the beginning and ending of your recording. This is known as trimming silence.

• To get the best results from your recording efforts, Avaya recommends that you use a microphone of professional quality. This is especially important if you want to convert your recordings to a format that the target voice response system can use.

• The target system plays only specially formatted files sampled at 8 kHz. When you use the recording application included with IVR Designer, it automatically records at an 8 kHz sampling rate and converts the recording during the code generation process to a form the target system can use. If you import *.wav files into your application, record them at the same rate to avoid conversion problems.

The Phrase Table Editor

The Phrase Table Editor is used to:

• Create new phrase tables

• Edit existing phrase tables
Phrase Table Editor window

The following figure shows the Phrase Table Editor window.

![Phrase Table Editor](image)

Window elements

The following table describes the elements of the Phrase Table Editor window.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase Table Name field</td>
<td>Type a new phrase table name or accept the existing name. Phrase table names must be eight or fewer characters long. IVR Designer truncates names with more than eight characters to the first eight characters. <strong>Note:</strong> Do not put square brackets [ ] around a new name. IVR Designer automatically inserts the brackets when it creates a new table.</td>
</tr>
<tr>
<td>Starting Phrase Number field</td>
<td>Enter a starting phrase number for this phrase table. By default, IVR Designer assigns the first unallocated free phrase number.</td>
</tr>
<tr>
<td>Number of Phrases field</td>
<td>Enter the number of phrases you want to allocate for this phrase table.</td>
</tr>
<tr>
<td>For Code Generation, Phrases Use field</td>
<td>Select whether phrases should be generated for playback on the target system using recorded speech or Text-to-Speech.</td>
</tr>
</tbody>
</table>
### Working with phrases

| Use the following base URI checkbox and textbox | When checked, you can enter a base URI for a Phrase Table. For more information, see Assigning URIs to phrases and phrase tables. |
| Select TTS Language dropdown list box | From the dropdown list, select one of the available languages for this phrase table. The selected language applies to phrases that are spoken using Text-to-Speech. |

### Opening the Phrase Table Editor

To open the Phrase Table Editor:

1. Open the **Phrases Manager**.
2. Right-click anywhere in the Phrases Manager window *other than* the name bar. Then, from the menu, do one of the following:
   - Select **New Phrase Table...**
   - Select **Edit Phrase Table...**

   IVR Designer displays the **Phrase Table Editor** window.

### Creating Phrase Tables

To create a new phrase table:

1. Open the **Phrase Table Editor**.
2. Type a name for the new phrase table in the **Phrase Table Name** field.

   **Note:**
   Phrase table names must be eight or fewer characters long. IVR Designer truncates names with more than eight characters to the first eight characters.

3. Enter a **Starting Phrase Number** for the phrase table.
   By default, IVR Designer places the next unallocated phrase number in this field.

4. Enter the **Number of Phrases** you want to allocate to this phrase table.
   The default for new phrase tables is 100 phrases.

5. Select whether you want to use **Recorded Phrases** or **Text-to-Speech** for the phrase table during the code-generation process.

   IVR Designer defaults to Recorded Speech for code generation.

   **Note:**
   Text-to-Speech is available only if you have purchased it as an option on your CONVERSANT or Avaya IR system. In many cases, you will find it preferable
Creating a voice response application

to use recorded speech. To assign Text-to-Speech, select the **Text-to-Speech** option in the Phrase Table Editor window.

6. Click **OK**.

**Editing Phrase Tables**

To edit existing phrase tables:

1. Open the **Phrase Table Editor**.
2. Edit the phrase table fields as desired.
3. Click **OK**.

**Importing text**

A phrase table record consists of two fields: the phrase number and the phrase text. When creating text files to import as phrases into IVR Designer, you may create them with or without phrase numbers.

- If you include phrase numbers, there should be a one-to-one correspondence between the phrase numbers and the phrase text when you construct your text file.

- If you do not include phrase numbers, you must construct your files so as to separate your phrases clearly.

When importing files, keep the following considerations in mind:

- If you do not include phrase numbers in your source text, IVR Designer assigns phrase numbers automatically, beginning with the first unallocated phrase number.

- If you include phrase numbers in your source text:
  - Do not use numbers less than 5000. IVR Designer defaults to phrase numbers equal to or greater than 5000.

  **Caution:**
  
  *You can* use a number less than 5000, but if you do so, be careful. Some Enhanced Basic Speech tables use a high number of phrases, and by choosing a number less than 5000, you run the risk of overwriting phrases in that table.

  — Use a starting phrase number that is as close as possible to the last defined phrase number in the table. This is because IVR Designer creates blank phrases to fill in the missing numbers.

  For example, suppose that the last phrase number in the table is 5005, and you select a starting phrase number of 6000. IVR Designer then creates blank phrases
numbered 5506-5999 before creating the phrases you want and populating them with the imported phrase text.

— Decide ahead of time what kind of separator you will use and use that separator consistently. If you use more than one kind of separator, IVR Designer is not able to correctly create and populate the new phrases.

One advantage of using phrase numbers in your source text file is that you can replace noncontiguous phrases in a phrase table.

For example, suppose you have a list of eleven phrases, numbered 5000-5010, and you want to replace only three of them, 5003, 5006, and 5010, with new phrases. By creating a text file containing only those three phrase numbers and their corresponding phrase text, you can import that text file and replace only those three phrases.

Using the Phrase Text Import dialog box

You can use IVR Designer to import predefined text as a phrase from outside IVR Designer into the Phrases Manager. Choose from the following options to import text:

- From the clipboard to an existing phrase
- From the clipboard to a new phrase
- From a file to an existing phrase
- From a file to a new phrase

Importing text from the clipboard into an existing phrase

To import text from the clipboard to an existing phrase:

1. Copy the phrase you want to import to the clipboard using the Copy command.
2. Double-click the phrase you want to replace in the Phrases Manager.
   
   IVR Designer displays the Phrase Editor window.
3. Highlight the text in the Phrase Text field.
4. Press Control + v.

Importing text from the clipboard into a new phrase

To import text from the clipboard to a new phrase:

1. Copy the phrase you want to import to the clipboard using the Copy command.
2. Right-click the Phrases Manager window, anywhere other than the name bar, and select Import Phrases from Text...
Creating a voice response application

IVR Designer displays the **Phrase Text Import** dialog box.

3. Select **Clipboard** in the **Import From:*** box.

4. If the text you are importing contains phrase numbers, check the **Input contains phrase numbers** checkbox. If it does not, make sure this box is unchecked.

5. If you checked the **Input contains phrase numbers** checkbox, select the type of phrase number/phrase text separator you used from the **Phrase numbers are separated from phrase text by:*** drop-down menu. Choices include:
   - **Auto-Detect** – Checks for all types of separators, such as white space, commas, or tabs, and places the separation at the appropriate places
   - **White Space** – Looks for spaces in text files between numbers and letters and places the separation where it finds the spaces
   - **Commas** – Looks for commas in text files and places the separation where it finds them
   - **Tabs** – Looks for tab markers between numbers and letters and places the separation where it finds them

6. (Optional) To preview the phrases as they will appear in the Phrases Manager, click the **Preview** button.

7. Click **OK**.

**Importing text from a file to an existing phrase**

To import text from a file to an existing phrase:

1. Use a text editor to create the phrase text you want to import, and save the file.
   Be sure to include the phrase numbers for the phrases you want to replace.

2. Right-click the **Phrases Manager** window anywhere other than the name bar, and select **Import Phrases from Text...**

   IVR Designer displays the **Phrase Text Import** dialog box.

3. Select **File** in the **Import From:** box.

4. Do one of the following:
   - Use the **Browse Files** button to locate and select the file you want to import.
   - Type the full pathname of the file you want to import.

5. Check the **Input contains phrase numbers** checkbox.

6. Select the type of phrase number/phrase text separator you used from the **Phrase numbers are separated from phrase text by:** drop-down menu. Choices include:
Working with phrases

— **Auto-Detect** – Checks for all types of separators, such as white space, commas, or tabs, and places the separation at the appropriate places

— **White Space** – Looks for spaces in text files between numbers and letters and places the separation where it finds the spaces

— **Commas** – Looks for commas in text files and places the separation where it finds them

— **Tabs** – Looks for tab markers between numbers and letters and places the separation where it finds them

7. (Optional) To preview the phrases as they will appear in the Phrases Manager, click the **Preview** button.

8. Click **OK**.

   IVR Designer displays a warning box and asks whether you want to start numbering at 5000, the default that IVR Designer expects, or continue with your choice.

9. Select **Yes**.

   IVR Designer displays a warning box for each phrase you are trying to overwrite, and asks if you want to overwrite the existing phrases.

   — To overwrite all targeted phrases, select **All**.

   — To overwrite only selected phrases, answer **Yes** only to those phrases you want to overwrite.

   — If there are any additional phrases at the end, IVR Designer automatically assigns them new phrase numbers and populates the table with them.

### Importing text from a file to a new phrase

To import text from a file to a new phrase:

1. Use a text editor to create the phrase text you want to import, and save the file.

2. Right-click the **Phrases Manager** window anywhere other than the name bar, and select **Import Phrases from Text...**

   IVR Designer displays the **Phrase Text Import** dialog box.

3. Select **File** in the **Import From:** box.

4. Do one of the following:

   — Use the **Browse Files** button to locate and select the file you want to import.

   — Type the full pathname of the file you want to import.
Creating a voice response application

5. If the text you are importing contains phrase numbers, check the Input contains phrase numbers checkbox. If it does not, make sure this box is unchecked.

6. If you checked the Input contains phrase numbers checkbox, select the type of phrase number or phrase text separator you used from the Phrase numbers are separated from phrase text by: drop-down menu. Choices include:
   
   — Auto-Detect – Checks for all types of separators, such as white space, commas, or tabs, and places the separation at the appropriate places
   
   — White Space – Looks for spaces in text files between numbers and letters and places the separation where it finds the spaces
   
   — Commas – Looks for commas in text files and places the separation where it finds them
   
   — Tabs – Looks for tab markers between numbers and letters and places the separation where it finds them

7. (Optional) To preview the phrases as they will appear in the Phrases Manager, click the Preview button.

8. Click OK.

IVR Designer automatically assigns the imported phrases new phrase numbers and populates the table with them.

Using the Create Default Phrases dialog box

To create default phrases and use them to populate phrase entries in the Phrases Manager:

1. Right-click the Phrases Manager anywhere other than on the name bar.

2. Select Fill Phrases with Default...

IVR Designer displays the Create Default Phrases window.

3. Enter the phrase number you want to start with in the Starting Phrase Number: field.

IVR Designer defaults to the first unallocated phrase number. If you want to start with a different number, delete the current number and replace it with the number you want.

If you have designated a phrase number that is already in use and then click OK, the following message is displayed:

This will overwrite existing phrases. OK to continue?

— To continue overwriting existing phrases, click Yes.

— To return to the Create Default Phrases dialog box and change the starting number, click No or Cancel.
4. Enter the number of new phrases you want to create in the **Number of Phrases:** field.
5. Click **OK**.

IVR Designer creates the number of new phrases you requested. Each phrase is populated with its phrase number as the default text.

---

**Working with VoiceXML grammars**

A grammar specifies what to listen for after a caller is prompted to enter information. IVR Designer offers built-in and custom grammars for VoiceXML applications.

**Built-in VoiceXML grammars**

Built-in grammars support both DTMF, which listens for touchtone input (the keys 0 through 9, *, and #), and voice entry. The following table describes the built-in grammar types.

<table>
<thead>
<tr>
<th>Built-in grammar</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Valid spoken or DTMF inputs include affirmative and negative phrases appropriate to the locale. DTMF 1 is <strong>true</strong> and DTMF 2 is <strong>false</strong>. If the field value is subsequently used in a prompt, it is spoken as an affirmative or negative phrase appropriate for the current locale.</td>
</tr>
<tr>
<td>date</td>
<td>Valid spoken inputs include phrases that specify a date, including a month, day, and year. DTMF inputs are: four digits for the year, followed by two digits for the month, followed by two digits for the day. The result is a fixed-length date string with the format <strong>yyyymmdd</strong>, for example 20020308.</td>
</tr>
<tr>
<td>digits</td>
<td>Valid spoken or DTMF inputs include one or more digits, 0 through 9. The result is a string of digits. If the field value is subsequently used in a prompt, it will be spoken as a sequence of digits.</td>
</tr>
<tr>
<td>currency</td>
<td>Valid spoken inputs include phrases that specify a currency amount. For DTMF input, the * key is the decimal point. The result is a string with the format <strong>UUUmm.nn</strong>, where <strong>UUU</strong> is the three character currency indicator according to ISO standard 4217:1995. If the field is subsequently used in a prompt, it will be spoken as a currency amount appropriate to the current locale.</td>
</tr>
<tr>
<td>number</td>
<td>Valid spoken inputs include phrases that specify numbers, such as one hundred twenty-three or five point three. Valid DTMF input includes positive numbers entered using digits and the * key to represent a decimal point. The result is a string of digits from 0 to 9, which may include a decimal point.</td>
</tr>
</tbody>
</table>
Valid spoken inputs include phrases that specify a phone number. The result is a string containing a telephone number consisting of digits and optionally containing the character x to indicate a phone number with an extension. For North America, one possible result is: 8005551234x789.

Valid spoken inputs include phrases that specify a time, including hours and minutes. The result is a five character string in the format hhmmx, where x can have one of the following values: a for AM, p for PM, h to indicate a time specified using the 24-hour clock, or ? to indicate an ambiguous time. Because there is no DTMF convention for specifying AM/PM, the result from DTMF input will always end with h or ?. If the field is subsequently used in a prompt, the value will be spoken as a time appropriate to the current locale.

Custom VoiceXML grammars

You can define custom VoiceXML grammars using the VXML Grammars Manager. The grammar can be located on the target voice response system, or on a remote web server. For information about creating a custom VoiceXML grammar that resides on the target voice response system, see Creating a local VoiceXML grammar on page 122. For information about creating a custom VoiceXML grammar that resides on a remote web server, see Creating an external VoiceXML grammar. For practice creating a custom VoiceXML grammar, see Using voice input in VoiceXML applications on page 304.

The VXML Grammars Manager

The VXML Grammars Manager displays the contents of all VoiceXML grammars, which are word lists that are defined for the application.

Opening the VXML Grammars Manager

To open the VXML Grammars Manager, do one of the following:

- From the View menu, select VXML Grammars.
- If the VXML Grammars Manager is docked in the the Globals Manager, open the Globals Manager and locate the VXML Grammars Manager.

Creating a local VoiceXML grammar

IVR Designer makes it easy for you to create new VoiceXML grammars (also called word lists) to use in your applications. Once created, these grammars become part of the application. Use the following procedure to create a VoiceXML grammar that will reside on the target voice response system. For more information about using a VoiceXML grammar in an application, see Creating custom VoiceXML grammars on page 304.

To create a local VoiceXML grammar:
1. Open the **VXML Grammars Manager**.
2. Right-click the VXML Grammars Manager somewhere other than on the name bar.
3. Select **New...**
   IVR Designer displays a dialog box and asks you to name the new VoiceXML grammar.
4. Enter a name for the VoiceXML grammar in the space provided.
5. Click **OK**.
   IVR Designer displays the **VXML Grammar Editor** window.
6. Enter the words you want in your VoiceXML grammar in the **Word List** field. Single-byte or double-byte text is allowed in this field.
7. When you are finished adding words, click **OK**.
   IVR Designer displays the new VoiceXML grammar in the VXML Grammars Manager.

### Creating an external VoiceXML grammar

IVR Designer makes it easy to specify an external VoiceXML grammar that was not created using IVR Designer. For information about creating an external VoiceXML grammar, refer to *Speech Recognition Grammar Specification Version 1.0*, W3C Working Draft, August 2001 at the following URL: [http://www.w3.org/](http://www.w3.org/)

In the following procedures, the words you enter for the VoiceXML grammar are used in application development. For example, a word from a VoiceXML grammar could be used to specify the branch of a Prompt and Collect node to take if a caller speaks that word in response to a prompt.

**External VoiceXML grammar on target system**

To specify an external VoiceXML grammar that resides on the Avaya Interactive Response or CONVERSANT system:

1. Open the **VXML Grammars Manager**.
2. Right-click the VXML Grammars Manager somewhere other than on the name bar.
3. Select **New...**
   IVR Designer displays a dialog box and asks you to name the new VoiceXML grammar.
4. Enter a name for the VoiceXML grammar in the space provided.
5. Click **OK**.
   IVR Designer displays the **VXML Grammar Editor** window.
6. Enter the words you want in the VoiceXML grammar in the **Word List** field. Single-byte or double-byte text is allowed in this field.

   **Note:**
   IVR Designer does not verify that the words you enter correspond to values in the external VoiceXML grammar that you specify in step 8. Therefore, make sure that you enter words that match values in the specified external VoiceXML grammar.

7. Select the **Use External Grammar** checkbox.

8. In the **Source:** field, enter a URL that specifies the location of the external VoiceXML grammar. Single-byte or double-byte text is allowed in this field.

   **Note:**
   If the VoiceXML grammar resides in the same directory as the application’s generated VoiceXML file, specify only the grammar file name, for example, `grammarFile.gram`.

9. When you are finished adding words, click **OK**.

   IVR Designer displays the new VoiceXML grammar in the VXML Grammars Manager.

---

**External VoiceXML grammar on external web server**

To specify an external VoiceXML grammar that resides on an external web server:

1. Open the **VXML Grammars Manager**.

2. Right-click the VXML Grammars Manager somewhere other than on the name bar.

3. Select **New...**

   IVR Designer displays a dialog box and asks you to name the new VoiceXML grammar.

4. Enter a name for the VoiceXML grammar in the space provided.

5. Click **OK**.

   IVR Designer displays the **VXML Grammar Editor** window.

6. Enter the words you want in the VoiceXML grammar in the **Word List** field. Single-byte or double-byte text is allowed in this field.

   **Note:**
   IVR Designer does not verify that the words you enter correspond to values in the external VoiceXML grammar that you specify in step 8. Therefore, make sure that you enter words that match values in the specified external VoiceXML grammar.

7. Select the **Use External Grammar** checkbox.

8. In the **Source:** field, enter a URI that specifies the location of the external VoiceXML grammar. Single-byte or double-byte text is allowed in this field.
Note:
The external VoiceXML grammar must be in XML format.

9. In the **Type**: field, select **xml Grammar**.

10. When you are finished adding words, click **OK**.
    
    IVR Designer displays the new VoiceXML grammar in the VXML Grammars Manager.

---

### Renaming a VoiceXML grammar

To rename a VoiceXML grammar:

1. Open the **VXML Grammars Manager**.
2. Right-click the name of the VoiceXML grammar you want to rename.
3. Select **Rename...**
    
    IVR Designer displays a dialog box and prompts you for a new name for the grammar.
4. Enter the new name you want for the VoiceXML grammar in the space provided.
5. Click **OK**.
    
    The name of the VoiceXML grammar changes in the VXML Grammars window.

---

### Editing a VoiceXML grammar

To edit a VoiceXML grammar:

1. Open the **VXML Grammars Manager**.
2. Right-click the name of the VoiceXML grammar you want to edit.
3. Select **Edit VXML Grammar...**
    
    IVR Designer displays the **VXML Editor** window.
4. Edit your VoiceXML grammar in the **Word List** field.
5. When you are finished editing the grammar, click **OK**.
    
    The VXML Editor disappears, and the edited VoiceXML grammar appears in the **VXML Grammars Manager** window.
Right-click menu options for the VXML Grammars Manager

The following table describes the options available from the VXML Grammars Manager right-click menu.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New...</td>
<td>Lets you create a new VoiceXML grammar.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the VoiceXML grammar from the VXML Grammars Manager, if it is not in use in the application.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you have used the VoiceXML grammar in your application, you must first remove its use from the application before you can delete it.</td>
</tr>
<tr>
<td>Rename...</td>
<td>Lets you rename the selected VoiceXML grammar.</td>
</tr>
<tr>
<td>Find Uses of</td>
<td>Opens the Find window, searches for Uses Of the selected VoiceXML grammar, and presents the results of the search in the Find window results field.</td>
</tr>
<tr>
<td>Stay on Top</td>
<td>(Available only when the VXML Grammars Manager is undocked) Toggles on and off. When checked (toggled on), it causes the VXML Grammars Manager window to remain on top of other open IVR Designer windows.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you have more than one resource manager window toggled to &quot;Stay on Top&quot;, the window actually on top will be the last window you clicked in. All windows toggled to &quot;Stay on Top&quot;, however, will stay on top of any other IVR Designer windows that are open.</td>
</tr>
<tr>
<td>Edit VXML Grammar</td>
<td>Lets you edit a VoiceXML grammar.</td>
</tr>
</tbody>
</table>

Using double-byte data

This topic applies only to VXML applications. For TAS applications, double-byte data is only allowed in the prompt and phrase editors. See Internationalization and globalization on page 13 for additional information that pertains to both VXML and TAS applications.

Operating system considerations

IVR Designer generates VXML applications using the native font of the operating system (OS). If the OS font doesn't match the language of the VXML application you are generating, some characters generate incorrectly. One of the following situations may occur:
Using double-byte data

- If the OS system is not double-byte but the VXML application is double-byte, the double-byte characters are generated incorrectly.
- If the OS system is double-byte but the VXML application is Latin-1 (not double-byte), some non-alphanumeric characters are generated incorrectly.

Use one of the following approaches to create the VXML application:

- If you have access to a PC with an operating system that matches the VXML format, use it to generate the VXML application.
- Use IVR Designer to generate and save the VXML application. Use a text editor to copy and paste the changes into the VXML application and save the VXML file.

Fields that support double-byte data

The following table lists the fields that accept double-byte data and provide links to more detailed descriptions of each window or node.

<table>
<thead>
<tr>
<th>Window or Node</th>
<th>Field accepts double-byte characters</th>
<th>For more information, see:</th>
</tr>
</thead>
<tbody>
<tr>
<td>VXML Grammar Editor</td>
<td>Source URL, word list</td>
<td>Creating a local VoiceXML grammar on page 122 or Creating an external VoiceXML grammar on page 123</td>
</tr>
<tr>
<td>Prompt and Collect node</td>
<td>ASR Language (drop-down menu with language choices)</td>
<td>Prompt and Collect node attributes on page 364</td>
</tr>
<tr>
<td>Inline Code node</td>
<td>Text Editor</td>
<td>Inline Code node on page 356</td>
</tr>
<tr>
<td>Variables Manager</td>
<td>Default column</td>
<td>The Variables Manager window on page 84 elements</td>
</tr>
<tr>
<td>Log/Trace node</td>
<td>Text</td>
<td>Log/Trace node on page 358</td>
</tr>
<tr>
<td>Set Test Expression Editor</td>
<td></td>
<td>Set Test Expression Editor on page 372</td>
</tr>
<tr>
<td>• Assign</td>
<td>Source Value</td>
<td></td>
</tr>
<tr>
<td>• Concat</td>
<td>Source String 1, Source String 2</td>
<td></td>
</tr>
<tr>
<td>• ItemCount</td>
<td>Source List</td>
<td></td>
</tr>
<tr>
<td>• ItemOf</td>
<td>Source List</td>
<td></td>
</tr>
<tr>
<td>• Length</td>
<td>Source String</td>
<td></td>
</tr>
<tr>
<td>• Parse</td>
<td>Source String, Parsing Separator</td>
<td></td>
</tr>
<tr>
<td>• Substring</td>
<td>Source Substring</td>
<td></td>
</tr>
<tr>
<td>Set Test Branch Condition Editor, Branch Conditions tab</td>
<td>Value fields</td>
<td>Set Test Branch Condition Editor on page 376</td>
</tr>
</tbody>
</table>
VoiceXML database operations

This section describes the various database operations that can be used in VoiceXML applications to access remote databases like MSSQL, DB2, and Sybase.

Avaya IVR Designer Release 5.1 or later has a sample VoiceXML application called JDBC_SP. This application plays a welcome announcement and then presents the user with a menu for selecting the type of database operation to perform. The topics in this section describe database operations (menu options) that are used in the JDBC_SP sample application.

For more information about and practice using VoiceXML database operations and stored procedures, see Using database operations in VoiceXML applications on page 313 and Using stored procedures in VoiceXML applications on page 322.

Select operation

Option one of the JDBC_SP sample application performs the Select operation. The Inline Code node for the Select operation uses the following VMXL code:

```xml
<var name="CHANNEL" expr="Channel"/>

<subdialog name="MyServlet"
src="http://localhost/servlet/JdbcServlet" method="get"
namelist="CHANNEL CMD TNAME CRITERIA GETCOLUMNS DIPNAME">

  <filled>
    <if cond="MyServlet.retcod&el;='0'">
      <prompt>Got a negative error code</prompt>
      <goto expr='"#d3f8ult3x1tF0rm'"/>
    </if>

    <assign name="document.RetFieldNum" expr="MyServlet.EMPNUM"/>
    <assign name="document.RetField2" expr="MyServlet.PAY"/>

  </filled>

</subdialog>
```
As shown in the code above, the VXML subdialog tag is used to invoke the servlet from the VXML code. In this example, the subdialog is named MyServlet. JdbcServlet is the servlet provided by Avaya for JDBC access and is specified in the src attribute. The data is sent over to the servlet by the get method. The namelist attribute specifies the arguments to be passed to the servlet, which are in turn used to perform the database operation. All the fields specified for the namelist attribute need to exist as application variables in the IVR Designer application. The following table describes the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANNEL</td>
<td>Specifies the channel number of the call in progress. IVR Designer provides a system variable called Channel in every application for this use. As can be seen from the first line in the VXML code, the system variable Channel is copied into the local CHANNEL variable.</td>
</tr>
<tr>
<td>CMD</td>
<td>Specifies the database command to perform. In this application, CMD has been declared as a character variable in the application and has been initialized to the string SELECT in the SetSelectParams node.</td>
</tr>
<tr>
<td>TNAME</td>
<td>Specifies the Table Name to perform the database operation on. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string JDBC_TBL in the SetSelectParams node.</td>
</tr>
<tr>
<td>CRITERIA</td>
<td>Specifies the Criteria for the select operation. In this application, CRITERIA has been declared as a character variable in the application and has been initialized to the string EMPNUM=12345 in the SetSelectParams node. The Criteria field should only be assigned a comma-separated list of strings that are in the format: &quot;&lt;DB_FLDNAME=&lt;VALUE&gt;&gt;&quot;. The &quot;=&quot; operation can be replaced by any of the operators: '&lt;', '&lt;=', '&gt;', '&gt;=', '!='. An example of specifying multiple criteria in this application would need CRITERIA to be set to the following: &quot;EMPNUM=12345,PAY='100000'&quot;</td>
</tr>
<tr>
<td>GETCOLUMNS</td>
<td>Specifies the Columns or database field names to return from the query operation. In this application, GETCOLUMNS has been declared as a character variable in the application and has been initialized to the string EMPNUM,PA in the SetSelectParams node. PAY and WIFE are column names in the JDBC_TBL table being used in this example.</td>
</tr>
<tr>
<td>DIPNAME</td>
<td>Specifies the DBDIP on the Avaya IR platform that should be used for accessing the database. One of the DIPs on the IR platform should be configured using the Web Administration interface to be able to access the required database. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string DBDIP2 in the Variables Manager.</td>
</tr>
</tbody>
</table>

As shown in the VXML code above, a retcode value is always returned from the servlet indicating the success or failure of the servlet and database operation. The retcode field will
Creating a voice response application

contain a positive value if the operation was successful. In this example, an error prompt is being played if the retcode contains a value less than or equal to zero. Other values returned from the servlet can be assigned to application variables. In this case, the servlet returns EMPNUM and PAY, since they were identified in the GETCOLUMNS parameter as fields to be returned.

Next operation

Option two in the JDBC_SP sample application performs the Next operation. The Inline Code node for the Next operation uses the following VXML code:

```xml
<var name="CHANNEL" expr="Channel"/>

<subdialog name="MyServlet"
src="http://localhost/servlet/JdbcServlet" method="get"
namelist="CHANNEL CMD DIPNAME">
    <filled>
        <if cond="MyServlet.retcode&lt;='0'">
            <prompt> Got a negative error code </prompt>
            <goto expr="'#d3f8ult3x1tF0rm'"/>
        </if>
        <assign name="document.RetField1" expr="MyServlet.PAY"/>
        <assign name="document.RetField2" expr="MyServlet.WIFE"/>
    </filled>
</subdialog>

<catch event="error.badfetch">
    <goto expr="'#d3f8ult3x1tF0rm'"/>
</catch>
```

The namelist attribute specifies the arguments to be passed to the servlet, which are in turn used to perform the database operation. The following table describes the fields:
Field | Description
---|---
CHANNEL | Specifies the channel number of the call in progress. IVR Designer provides a system variable called Channel in every application for this use. As can be seen from the first line in the VXML code, the system variable Channel is copied into the local CHANNEL variable.
CMD | Specifies the database command to perform. In this application, CMD has been declared as a character variable in the application and has been initialized to the string NEXT in the SetSelectParams node.
DIPNAME | Specifies the DBDIP on the Avaya IR platform that should be used for accessing the database. One of the DIPs on the IR platform should be configured using the Web Administration interface to be able to access the required database. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string DBDIP2 in the Variables Manager.

As shown in the VXML code, a retcode value is always returned from the servlet indicating the success or failure of the servlet and database operation. The retcode field will contain a positive value if the operation was successful and will have a value of zero if there are no more records to be returned. In this example, an error prompt is being played if the retcode contains a value less than or equal to zero.

### Insert operation

Option three of the JDBC_SP sample application performs the Insert operation. For the insert operation, the user is prompted for an employee number and the user input is used to insert a record into the database table. The Inline Code node for the Insert operation uses the following VXML code:

```xml
<var name="CHANNEL" expr="Channel"/>

<subdialog name="MyServlet"
src="http://localhost/servlet/JdbcServlet" method="get"
namelist="CHANNEL CMD TNAME SETCOLUMNS DIPNAME">
  <filled>
    <if cond="MyServlet.retcode&lt;'0'">
      <prompt> Got a negative error code </prompt>
      <goto expr="'#d3f8ult3x1tF0rm'"/>
    </if>
  </filled>
  <assign name="document.RetCode" expr="MyServlet.retcode"/>
</subdialog>
```
Creating a voice response application

The namelist attribute specifies the arguments to be passed to the servlet, which are in turn used to perform the database operation. The following table describes the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANNEL</td>
<td>Specifies the channel number of the call in progress. IVR Designer provides a system variable called Channel in every application for this use. As can be seen from the first line in the VXML code, the system variable Channel is copied into the local CHANNEL variable.</td>
</tr>
<tr>
<td>CMD</td>
<td>Specifies the database command to perform. In this application, CMD has been declared as a character variable in the application and has been initialized to the string INSERT in the SetSelectParams node.</td>
</tr>
<tr>
<td>TNAME</td>
<td>Specifies the Table Name to perform the database operation on. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string JDBC_TBL in the SetSelectParams node.</td>
</tr>
<tr>
<td>SETCOLUMNS</td>
<td>Specifies the Criteria for the delete operation. In this application, CRITERIA has been declared as a character variable in the application and has been initialized to the string EMPNUM=&lt;USER_INPUT&gt; in the SetDeleteParams node. The Criteria field should only be assigned a comma-separated list of strings that are in the format: &quot;&lt;DB_FLDNAME=&lt;VALUE&gt;&gt;&quot;. The &quot;=&quot; operation can be replaced by any of the operators: '&lt;', '&lt;=', '&gt;', '==', '!='. An example of specifying multiple criteria in this application would need CRITERIA to be set to the following: &quot;EMPNUM=12345,PAY='100000'&quot;</td>
</tr>
<tr>
<td>DIPNAME</td>
<td>Specifies the DBDIP on the Avaya IR platform that should be used for accessing the database. One of the DIPs on the IR platform should be configured using the Web Administration interface to be able to access the required database. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string DBDIP2 in the Variables Manager.</td>
</tr>
</tbody>
</table>

As shown in the VXML code, a retcode value is always returned from the servlet indicating the success or failure of the servlet and database operation. The retcode field will contain a positive value if the operation was successful. In this example, an error prompt is being played if the retcode contains a value less than or equal to zero.
Delete operation

Option four of the JDBC_SP sample application performs the **Delete** operation. For the delete operation, the user is prompted for an employee number and the user input is used to identify the record to delete from the database table. The Inline Code node for the Delete operation uses the following VXML code:

```vxml
<var name="CHANNEL" expr="Channel"/>

<subdialog name="MyServlet"
src="http://localhost/servlet/JdbcServlet" method="get"
namelist="CHANNEL CMD TNAME CRITERIA DIPNAME">
  <filled>
    <if cond="MyServlet.retcode&lt;='0'">
      <prompt> Got a negative error code </prompt>
      <goto expr="'#d3f8ult3xitF0rm'"/>
    </if>
    <assign name="document.RetCode" expr="MyServlet.retcode"/>
  </filled>
</subdialog>

<catch event="error.badfetch">
  <goto expr="'#d3f8ult3xitF0rm'"/>
</catch>
```

The namelist attribute specifies the arguments to be passed to the servlet, which are in turn used to perform the database operation. The following table describes the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANNEL</td>
<td>Specifies the channel number of the call in progress. IVR Designer provides a system variable called Channel in every application for this use. As can be seen from the first line in the code snippet, the system variable Channel is copied into the local CHANNEL variable.</td>
</tr>
<tr>
<td>CMD</td>
<td>Specifies the database command to perform. In this application, CMD has been declared as a character variable in the application and has been initialized to the string DELETE in the SetDeleteParams node.</td>
</tr>
</tbody>
</table>
Creating a voice response application

<table>
<thead>
<tr>
<th>TNAME</th>
<th>Specifies the Table Name to perform the database operation on. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string JDBC_TBL in the SetDeleteParams node.</th>
</tr>
</thead>
</table>
| CRITERIA | Specifies the Columns or database field names and corresponding values to insert into the specified database table. In this application, SETCOLUMNS has been declared as a character variable in the application and has been initialized in the SetInsertParams node. The SETCOLUMNS field should only be assigned a comma-separated list of strings that are in the format: 
`"<DB_FLD1NAME>,<VALUE>,<DB_FLD2NAME>,<VALUE>,........."` |
| DIPNAME  | Specifies the DBDIP on the Avaya IR platform that should be used for accessing the database. One of the DIPs on the IR platform should be configured using the Web Administration interface to be able to access the required database. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string DBDIP2 in the Variables Manager. |

As shown in the VXML code, a *recode* value is always returned from the servlet indicating the success or failure of the servlet and database operation. The *recode* field will contain a positive value if the operation was successful. In this example, an error prompt is being played if the *recode* contains a value less than or equal to zero. Other values returned from the servlet can be assigned to application variables. In this case, the servlet returns PAY and WIFE assuming that these were the fields specified in the most recent query operation.

**Update operation**

Option five of the JDBC_SP sample application performs the **Update** operation. For the update operation, the user is prompted for an employee number and the user input is used to identify the record to update in the database table. The Inline Code node for the Update operation uses the following VXML code:

```xml
<var name="CHANNEL" expr="Channel"/>

<subdialog name="MyServlet"
src="http://localhost/servlet/JdbcServlet" method="get"
namelist="CHANNEL CMD TNAME CRITERIA SETCOLUMNS DIPNAME">
  <filled>
    <if cond="MyServlet.recode&lte;'0'">
      <prompt> Got a negative error code </prompt>
      <goto expr="'#3f8ult3x1tF0rm'"/>
    </if>
  </filled>
</subdialog>
```
The namelist attribute specifies the arguments to be passed to the servlet, which are in turn used to perform the database operation. The following table describes the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANNEL</td>
<td>Specifies the channel number of the call in progress. IVR Designer provides a system variable called Channel in every application for this use. As can be seen from the first line in the VXML code, the system variable Channel is copied into the local CHANNEL variable.</td>
</tr>
<tr>
<td>CMD</td>
<td>Specifies the database command to perform. In this application, CMD has been declared as a character variable in the application and has been initialized to the string UPDATE in the SetSelectParams node.</td>
</tr>
<tr>
<td>TNAME</td>
<td>Specifies the Table Name to perform the database operation on. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string JDBC_TBL in the SetUpdateParams node.</td>
</tr>
<tr>
<td>CRITERIA</td>
<td>Specifies the Columns or database field names and corresponding values to insert into the specified database table. In this application, SETCOLUMNS has been declared as a character variable in the application and has been initialized in the SetUpdateParams node. The SETCOLUMNS field should only be assigned a comma-separated list of strings that are in the format: &quot;&lt;DB_FLD1NAME&gt;,&lt;VALUE&gt;,&lt;DB_FLD2NAME&gt;,&lt;VALUE&gt;,...........&quot;</td>
</tr>
</tbody>
</table>
| SETCOLUMNS  | Specifies the Criteria for the delete operation. In this application, CRITERIA has been declared as a character variable in the application and has been initialized to the string EMPNUM=<USER_INPUT> in the SetUpdateParams node. The Criteria field should only be assigned a comma-separated list of strings that are in the format: "<DB_FLDNAME=<VALUE>". The "=" operation can be replaced by any of the operators: '<', '<=', '>', '>=', '!='. An example of specifying multiple criteria in this application would need CRITERIA to be set to the following: "EMPNUM=12345,PAY='100000"
Creating a voice response application

| DIPNAME | Specifies the DBDIP on the Avaya IR platform that should be used for accessing the database. One of the DIPs on the IR platform should be configured using the Web Administration interface to be able to access the required database. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string DBDIP2 in the Variables Manager. |

As shown in the VXML code, a `retcode` value is always returned from the servlet indicating the success or failure of the servlet and database operation. The `retcode` field will contain a positive value if the operation was successful. In this example, an error prompt is being played if the `retcode` contains a value less than or equal to zero.

### Stored Procedure

Option six of the JDBC_SP sample application performs the **Stored Procedure** operation. The Inline Code node for the Stored Procedure operation uses the following VXML code:

```xml
<var name="CHANNEL" expr="Channel"/>

<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get" namelist="CHANNEL CMD SPNAME PARAMS SPRETCODE DIPNAME">
    <filled>
        <if cond="MyServlet.retcode\lt;='0'">
            <prompt> Got a negative error code </prompt>
            <goto expr="'#d3f8ult3xtF0rm'"/>
        </if>

        <assign name="document.RetCode" expr="MyServlet.retcode"/>
        <assign name="document.RetCode_SP" expr="MyServlet.spretcode"/>
        <assign name="document.RetParam1" expr="MyServlet.param1"/>
    </filled>
</subdialog>

<catch event="error.badfetch">
```

136 Avaya IVR Designer Help for VoiceXML Applications
The namelist attribute specifies the arguments to be passed to the servlet, which are in turn used to perform the database operation. The following table describes the fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANNEL</td>
<td>Specifies the channel number of the call in progress. IVR Designer provides a system variable called Channel in every application for this use. As can be seen from the first line in the VXML code, the system variable Channel is copied into the local CHANNEL variable.</td>
</tr>
<tr>
<td>CMD</td>
<td>Specifies the database command to perform. In this application, CMD has been declared as a character variable in the application and has been initialized to the string STOREDPROC in the SetSelectParams node.</td>
</tr>
<tr>
<td>SPNAME</td>
<td>Specifies the Store procedure name to call on the database. In this application, SPNAME has been declared as a character variable in the application and has been initialized to the string SP1 in the SetSPParams node.</td>
</tr>
<tr>
<td>PARAMS</td>
<td>Specifies the parameters to pass to and return from the stored procedure. In this example, PARAMS is assigned the value &quot;STRING,699999,IN,STRING,null,OUT&quot; in the SetSPParams node. This is because the stored procedure SP1 expects an input string parameter and returns an output string parameter. The PARAMS field should only be assigned a comma-separated list of strings that are in the format: &quot;&lt;TYPE&gt;,&lt;VALUE&gt;,&lt;IN/OUT/INOUT&gt;,&lt;TYPE&gt;,&lt;VALUE&gt;,&lt;IN/OUT/INOUT&gt;.....&quot; The values that can be passed for &lt;TYPE&gt; are: STRING, NUMBER, DATE, TIME, or TIMESTAMP. When using a null STRING input parameter, the value of the parameter should be empty. For example, the PARAMS for a stored procedure with one STRING input parameter should be set to &quot;STRING,,IN&quot; when the value of the string input is null. Note: Applications can execute stored procedures created on a database and retrieve the resultset/values returned by the procedure. However, testing has revealed that a timestamp (date and time) parameter may fail to pass to the stored procedure.</td>
</tr>
<tr>
<td>SPRETCODE</td>
<td>Specifies if the stored procedure returns a value other than the parameters specified as OUT. Note that this feature is not supported by all databases. In this application, SPRETCODE has been declared as a character variable in the application and has been initialized to the string YES in the Variables Manager.</td>
</tr>
</tbody>
</table>
Creating a voice response application

| DIPNAME | Specifies the DBDIP on the Avaya IR platform that should be used for accessing the database. One of the DIPs on the IR platform should be configured using the Web Administration interface to be able to access the required database. In this application, TNAME has been declared as a character variable in the application and has been initialized to the string DBDIP2 in the Variables Manager. |

As shown in the VXML code, a **retcode** value is always returned from the servlet indicating the success or failure of the servlet and database operation. The **retcode** field will contain a positive value if the operation was successful. In this example, an error prompt is being played if the **retcode** contains a value less than or equal to zero. Since the SPRETCODE field has been set to YES, the stored procedure also returns a value that can be obtained by accessing **MyServlet.spretcode**. Other values returned by the stored procedure and originally specified as OUT parameters, can be accessed by **MyServlet.param1**, **MyServlet.param2** etc. based on the number of parameters returned by the stored procedure. If the stored procedure returns a resultset or recordset, only one row is returned at a time to the application. In such a case, the individual fields returned can be obtained by accessing **MyServlet.<DB_FLDNAME>**. Successive rows can be accessed by using the Next operation.
Verifying and testing an application

Verifying and testing an application describes how to use the Verify Design tool to check your application for possible errors or omissions, and how to use the Simulation tool to run your application in a simulated environment.

For more information about and practice using the Verify Design tool and the Simulation tool, see Testing a VoiceXML application on page 199.

This section includes the following topics:

Checking an application for errors ................................................ 139
Simulating an application .............................................................. 141

Checking an application for errors

You can use the Verify Design tool to check your IVR Designer application for possible errors or omissions. When you open the Verify Design tool, it automatically searches through each call flow of your application. The results are displayed in the Verification Results window. For practice verifying a VoiceXML application, see Verifying a VoiceXML application on page 199.

Opening and running the Verify Design tool

The Verify Design tool runs automatically when you open it.

To open the Verify Design tool, select Verify Design from the Tools menu. IVR Designer performs the design verification actions and displays the results in the Verification Results window.

Verification Results window elements

The following table describes the elements of the Verification Results window.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results display pane</td>
<td>Displays the results of the verification</td>
</tr>
</tbody>
</table>
Verifying and testing an application

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Indicates that possible errors at this node are hidden. \nTo view the possible errors in this node, double-click the icon to expand it.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indicates that possible errors at this node are displayed below the node name.</td>
</tr>
<tr>
<td>✔</td>
<td>Indicates a possible error or omission. The node where the possible error or omission is located is listed just above this description.</td>
</tr>
<tr>
<td></td>
<td>Call flow tabs \nUsed to select the call flow for which you want to view verification results. The results for each call flow are contained in a separate display pane.</td>
</tr>
<tr>
<td></td>
<td>Lets you scroll to different call flows to view possible errors.</td>
</tr>
</tbody>
</table>

If the Verify Design tool finds no errors or omissions, the following message is displayed:

Verification discovered no errors.

Using verification results

When you run the Verify Design tool, IVR Designer displays the results of the search in the Verification Results window.

Note:

The results of the search indicate only possible errors or omissions. You must make the final determination whether a "flagged" node really is an error or omission.

- The + and + icons designate the names of nodes where the Verify Design tool found possible errors or omissions.
- If the directory symbol is open (default), the lines immediately below marked with the ✔ icon describe the possible errors or omissions.

To locate the node within its call flow, double-click the name or the description of the node.

To view another call flow's search results, click that call flow's tab at the bottom of the window.

Right-click menu options for the Verification Results window

The following table describes the right-click menu options available from the Verification Results window.

Note:

The right-click menu is displayed only when you right-click an entry in the Verification Results window.
Simulating an application

The **Simulation** tool runs your application in a simulated environment. The Simulation tool is often helpful in finding problems within the application, thus reducing the amount of time required to test your application on the target system. For practice simulating an application, see *Simulating a VoiceXML application* on page 200.

**Opening the Simulation tool**

To open and start the Simulation tool, select **Simulation** from the **Tools** menu. IVR Designer displays the **Application Simulation** window, and the simulation immediately starts running from the currently highlighted node.

**Note:**

The Simulation tool will not work if you attempt to start it from an empty branch. If an empty branch is highlighted and you try to run the Simulation tool, IVR Designer displays the following message:

> Start simulator from the beginning of the application?

**Application Simulation window elements**

The following table describes the elements of the **Application Simulation** window.

<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Locates the node in the design pad.</td>
</tr>
<tr>
<td>Verify Node</td>
<td>Verifies only the selected node for possible errors or omissions.</td>
</tr>
<tr>
<td>Verify Call Flow</td>
<td>Verifies only the current call flow.</td>
</tr>
<tr>
<td>Verify Design</td>
<td>Verifies the entire application design.</td>
</tr>
<tr>
<td>Expand</td>
<td>Expands a collapsed node entry, indicated by the icons.</td>
</tr>
<tr>
<td>Collapse</td>
<td>Collapses an expanded node entry, indicated by the icons.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Expands all node entries in the current call flow.</td>
</tr>
<tr>
<td>Collapse All</td>
<td>Collapses all node entries in the current call flow.</td>
</tr>
</tbody>
</table>
### Verifying and testing an application

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop button</td>
<td>Stops the simulation.</td>
</tr>
<tr>
<td>Pause button</td>
<td>Causes the simulation to pause.</td>
</tr>
<tr>
<td>Play button</td>
<td>Resumes running a simulation that has been paused or stopped, or runs a simulation that has ended.</td>
</tr>
<tr>
<td>Reset Variables button</td>
<td>Resets all variables to their default values.</td>
</tr>
<tr>
<td>Step-Through button</td>
<td>Steps through a simulation, one node at a time.</td>
</tr>
<tr>
<td>Simulation script field</td>
<td>Displays a script of all simulated events while the simulation is running.</td>
</tr>
<tr>
<td>Time Remaining field</td>
<td>Displays a &quot;countdown&quot; for each node that uses a timed delay for responses.</td>
</tr>
<tr>
<td>Collected Digits field</td>
<td>Contains any digits that have been collected (using touchtone input or data from a table) and not used.</td>
</tr>
<tr>
<td>Touchtone keypad</td>
<td>Simulates caller input from a touchtone telephone.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You may also enter alphanumeric touchtone input from your computer keypad or keyboard. You cannot enter touchtone input from your telephone for the Simulation tool.</td>
</tr>
<tr>
<td>Clear button</td>
<td>Clears any digits in the Collected Digits field.</td>
</tr>
<tr>
<td>ASR button</td>
<td>Simulates a spoken response from the caller.</td>
</tr>
<tr>
<td>Asynchronous Event button</td>
<td>Simulates an asynchronous event.</td>
</tr>
<tr>
<td></td>
<td>To simulate an asynchronous event (such as a customer hanging up in the middle of a node), press the Asynchronous Event button during a simulation.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you have defined a routine to handle asynchronous events, pressing this button calls that routine.</td>
</tr>
<tr>
<td>Force Timeout button</td>
<td>Enables you to skip the countdown, or delay, that is programmed into many nodes, and go on to the next node.</td>
</tr>
<tr>
<td>DPR button</td>
<td>Simulates a Dial Pulse Recognition (DPR) response from the caller.</td>
</tr>
</tbody>
</table>

---

### Running an application simulation

To run a simulation of an application or part of an application:
1. On the design pad, highlight the node or object where you want the simulation to start.

2. Open the Simulation tool.

   When you open the Simulation tool, the simulation starts running immediately.

   **Note:**
   You can use the Set Breakpoint option (from the right-click menu for each node) to step through a simulation or temporarily stop the simulation at points where you might want to look at what is going on within the application, check variable values, and so forth.

You do not have to start a simulation at the very beginning of your application. If you only want to test part of the application, you can choose any node within the application from which to start.

While the simulation is running, the script field displays a script that logs all simulated events.

If the application uses recorded phrases, the recorded phrases reside on your PC, and your PC has the appropriate hardware, the recorded phrases play at the appropriate place during the simulation.

If the application uses Text-to-Speech phrases, the script field displays the phrase text at the appropriate place during the simulation.

The Time Remaining: field displays the amount of time left for the current node, where there is a built-in time limit.

To force a timeout for a node, press the Force Timeout button. Forcing a timeout enables you to skip the countdown, or delay, that is programmed into many nodes.

Using the Force Timeout button can lessen the time it takes to move through a simulation.

3. Respond to any prompts according to what you want to test.

   Often, the response a caller is prompted for involves pressing a button on a telephone keypad. Use the touchtone keypad in the Application Simulation window to simulate this input. Use the mouse to click the desired buttons.

   To simulate a response from a dial pulse telephone (that is, from a caller who does not have a touchtone telephone), click the DPR button when you encounter a prompt in which a touchtone or dial pulse response is required from the caller.

   — To cause the simulation to pause, click the Pause button.
   
   — To stop the simulation, click the Stop button.
   
   — To resume running the simulation, click the Play button.
Verifying and testing an application

Note:
When you resume a simulation, it starts at the beginning of the node where you stopped or paused.

To step through the simulation one node at a time, click the Step-Through button.

When the simulation is finished, the message End of simulation is displayed in the script window.

4. (Optional) Do one of the following:
   - To reset the variables to their defaults, for example so that you can run the simulation again, press the Reset Variables button.
   - To clear Collected Digits, press the Clear button.
   - To run the simulation again, select the node you want to start from, and then click the Play button.

Examining the simulation script

While a simulation is running, the script field displays a script that logs all simulated events. This script is kept until you close the Application Simulation window.

To view parts of the script that have scrolled up in the field and are no longer visible, use the scrollbar at the right.

There are times when it is helpful to see the entire script at once. To do this, you can print the script.

To print a copy of the script:
1. Right-click the script window and select Select All.
   IVR Designer highlights the script text.
2. Right-click the script window again, and select Copy.
3. Paste the text into a document in a text processor such as Notepad or Wordpad.
4. Use the Print command in the text processor to print out the script.

Right-click menu options for the Simulation tool

The following table describes the right-click menu options for the Simulation tool.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Inactive in the current version.</td>
</tr>
<tr>
<td>Cut</td>
<td>Removes the highlighted text and copies it to the clipboard.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies the highlighted text to the clipboard, leaving the original where it is.</td>
</tr>
<tr>
<td>Paste</td>
<td>Pastes a selection previously copied to the clipboard to the current cursor location.</td>
</tr>
<tr>
<td>Delete</td>
<td>Removes the highlighted text. The deleted text is not copied to the clipboard.</td>
</tr>
<tr>
<td>Select All</td>
<td>Highlights all text in the script window so you can cut, copy, or delete it.</td>
</tr>
</tbody>
</table>
Generating and transferring an application

Generating and transferring an application describes how to use the Code Generation tool to generate the files necessary to use the application on the target voice response system, and how to use the Application Transfer tool to transfer the application files to the target voice response system.

The Application Transfer tool lets you transfer the generated code files using either a network connection or floppy disks. You can select all or only some of the generated code files for application transfer.

For more information about and practice using the Code Generation tool and the Application Transfer tool, see Generating and assigning VoiceXML applications on page 202.

This section includes the following topics:

Generating a VoiceXML application .............................................. 146
Transferring the application........................................................... 150
Transferring files using a modem.................................................. 155

Generating a VoiceXML application

The Code Generation tool generates the VoiceXML application (also called a VoiceXML document) that you transfer to the target voice response system. The code generated for VoiceXML applications uses the little-endian Unicode-16 file format.

Opening the Code Generation tool

To open the Code Generation tool, select Code Generation/Application Transfer from the Tools menu.

The appearance of the Code Generation/Application Transfer window depends on the following conditions:

- If errors exist in files already generated for the application, the file types appear in red text in the Target Files to Generate box.
- The checkboxes in the Target Files to Generate box are checked by default whenever:
— Target files of that type have not already been generated
— The application source files for that type have been changed since the last time code was generated

• File types are inactive whenever no items for that type have been defined within the application.

In IVR Designer Release 5.3, you have the option of selecting the output VXML document type, as shown below.

![Code Generation / Application Transfer: ASYNC](image)

• Click the VXML 2.0 Non-compliant checkbox, if you want to generate VXML 2.0 non-compliant documents.
• Click the VXML 2.0 compliant checkbox, if you want to generate VXML 2.0 compliant documents.

**Note:** VoiceXML 2.0 complaint documents are compatible only with an Avaya IR Release 2.0 system. VoiceXML 2.0 complaint documents can be made compatible with an Avaya IR Release 1.3 system, by installing the AVvoicxml2-0 package. Voice XML 2.0 compliant documents are not compatible with versions earlier than Avaya IR R1.3. VoiceXML 2.0 non-complaint documents are
Generating and transferring an application

incompatible with an Avaya IR Release 2.0 system. Click the **VXML 2.0 compliant** checkbox, if you want to generate VXML 2.0 compliant documents.

---

**Generating the VoiceXML application files**

To generate the source code files for the target system:

1. Open the **Code Generation/Application Transfer** tool window.

2. If you have changed the name of your application since the last build, select **Remove from the Build** menu.

   This option removes all files previously generated for the application. Avaya recommends that, to avoid confusion later on, you select this option whenever you change your application's name.

3. (Optional) Select the **Show Warnings** option if you want to see warnings for possible application errors or missing files, such as unrecorded speech files.

4. Select the files you want to generate.

   — To generate all files possible, select **Build All** from the **Build** menu.

   — To generate specific types of files, check the appropriate boxes on the list in the **Target Files to Generate** box.

5. Do one of the following:

   — Click the **Apply** button.

   — Select **Build All** from the **Build** menu.

If there are any files existing from a previous build, IVR Designer asks whether you want to overwrite them.

IVR Designer generates the files required for Application Transfer.

---

**The code generation process**

During the code generation process, the following actions occur:

- File type entries in the **Target Files to Generate** box are displayed in green text as they are being generated.

- Each file type checkbox is cleared as the Code Generation process is completed on the selected types of files.
• Any file type that generates an error is flagged with red text.

If IVR Designer encounters errors during the Code Generation process, it displays an error dialog box, and asks you if you want to see the errors. To continue without viewing the error log, click the **No** button. To view the error log, click the **Yes** button.

### Menus in the Code Generation window

The following table describes the menus in the **Code Generation** window and their options.

<table>
<thead>
<tr>
<th>Menu</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Open</td>
<td>Not available from the Code Generation tab view.</td>
</tr>
<tr>
<td>File</td>
<td>Exit</td>
<td>Closes the Code Generation/Application Transfer dialog box immediately.</td>
</tr>
<tr>
<td>Target</td>
<td>[Generated files list]</td>
<td>Lists all files that have been generated for the application. The Target Menu is inactive and is grayed out on the menu bar if no files have been generated for the application. To view the contents of a particular file from the list, click that file's name.</td>
</tr>
<tr>
<td>Build</td>
<td>Build All</td>
<td>Includes all possible files in the build and overwrites any files that have been generated previously. If previously generated files exist, IVR Designer displays the message: All existing target files will be removed. Do you want to continue? To continue with the build, click <strong>Yes</strong>. To cancel the operation, click <strong>No</strong>.</td>
</tr>
<tr>
<td>Build</td>
<td>Remove</td>
<td>Removes all previously generated files from the application. If none exist, this menu item is inactive. <strong>Note:</strong> Avaya recommends that you select this option whenever you change your application's name to avoid confusion later on. To remove all previously generated files, click <strong>Yes</strong>. To cancel the operation, click <strong>No</strong>.</td>
</tr>
<tr>
<td>Build</td>
<td>Errors</td>
<td>Displays the error log, if one exists. If no error log exists, this menu item is inactive.</td>
</tr>
</tbody>
</table>
Transferring the application

Use the Application Transfer tool to transfer the application files to the voice response system on which the application will be installed and run. This tool lets you transfer the generated code files using either a network connection or floppy disks. You can select all or some of the generated code files for application transfer.

Opening the Application Transfer window

To open the Application Transfer window from within IVR Designer, click the Tools Menu, and then select Code Generation/Application Transfer.

To open the Application Transfer window from outside IVR Designer (the standalone version):

1. Locate the shortcut for the IVR Designer Application Transfer standalone application. This file should reside in the directory in which you have installed your IVR Designer application files.
2. Click the IVR Designer Application Transfer option.

Note: The standalone version of the Application Transfer tool does not have or display any of the Code Generation tool options.

Setting application transfer options

The Application Transfer Options dialog box lets you select and set options for Application Transfer. You can choose to transfer your application files using floppy disks, or over a network directly to the target system. You can also elect to have IVR Designer notify you before overwriting files at the transfer destination and when the transfer is complete. Finally, you can elect whether to use your settings as the default settings for future applications.

To set the Application Transfer options:

1. Open the Code Generation/Application Transfer tool window.
2. Click the Application Transfer tab.
3. From the Options menu, select Application Transfer.
   IVR Designer displays the Application Transfer Options dialog box.
4. In the Type of Transfer box, select the method for transferring your application from your PC to the target system:
   — Network lets you transfer the application using a TCP/IP network connection.
     Selecting this option activates the Network area on the window.
Note:
It is possible to use the Network option to transfer your application using a
modem connection. In order to transfer files using a modem connection, you
must first set up the necessary PPP connections between your PC and the
target system. For information on setting up the necessary PPP connections,
see Transferring files using a modem on page 155.

— **Floppy** lets you transfer the application files to floppy disks, which can then be used to
transfer the files to the target system. Selecting this option activates the **Floppy** area
on the window.

5. If you want IVR Designer to notify you before overwriting any files with matching
filenames, click the **Files exist** checkbox.

    — If you click the **Files exist** checkbox, IVR Designer checks the destination system to
    see if files using the same application name exist. If they do, you get a warning when
    you try to transfer the application. If you choose to proceed with the transfer, IVR
    Designer overwrites any files with matching filenames.

    — If you do not click the **Files exist** checkbox, IVR Designer overwrites any files at the
destination with matching filenames without notifying you that it will do so.

6. If you want IVR Designer to notify you when the transfer process has been completed,
   click the **Transfer is complete** checkbox.

    — If you click the **Transfer is complete** checkbox, IVR Designer opens an alert box after
    the transfer is finished. A message states that the transfer is complete.

    — If you do not click the **Transfer is complete** checkbox, IVR Designer does not notify
    you when the transfer is complete.

7. In IVR Designer Release 5.3, you have the option of selecting the transfer mode, as
   shown below.
Generating and transferring an application

— If you click the **Normal** checkbox, IVR Designer transfers the application using File Transfer Protocol (FTP) service.

— If you click the **Secure (SSH/ SFTP)** option, IVR Designer transfers the application using the Secure File Transfer Protocol (SFTP) service. Secure Shell (SSH) is the underlying protocol over which the SFTP service is provided. SSH provides the authentication and security functions for SFTP. SSH (Secure Shell) is a program that includes capabilities for logging into another computer over a network, executing commands on a remote machine, and moving files from one machine to another. It provides strong authentication and secure communication over untrusted networks. SSH provides a more secure means of accessing remote systems than protocols such as telnet and FTP. Unlike FTP and telnet, SSH allows users to connect to remote hosts over an encrypted link. This protects against interception of clear text logins and passwords.

**Note:**
On IR systems using Solaris 10 as the operating system, SSH is provided by default. On IR systems using Solaris 8 as the operating system, SSH can be installed using the openSSH package. (http://www.sunfreeware.com). On Conversant V8 and V9, SFTP service is not supported. You will have to transfer the application using the FTP service.

8. If you want IVR Designer to transfer application source files along with the generated code, check the **Transfer Source** checkbox.

9. If you are transferring over a network, enter the following information about the target system in the **Network information** fields:
   a) Enter the name or the IP address of the destination host in the **Host:** field.

      If you have a Hosts file defined in your \Winnt\system32\drivers\etc\ directory (for Windows NT 4.0), the Host drop-down menu displays the names of the hosts listed in that file.

      Otherwise, the drop-down menu is blank, and you must provide the name of the host you want.

   b) Type the user login ID in the **Login:** field.

   c) Type the user password in the **Password:** field.

      For security reasons, only asterisks (*) appear in the **Password:** field as you type the password.

      If you are transferring to floppy disks, select the drive you want from the **Floppy** drop-down menu.

10. If you want IVR Designer to transfer application source files along with the generated code, check the **Transfer Source** checkbox.
11. If you want to make the options you have selected the default settings for future application transfers, click the Save As Default button.

   **Note:**
   If you do not click the Save As Default button, clicking OK sets the options for only the current application. The next application reverts to the previous defaults.

12. Click OK.

---

**Transferring files from floppy disks to the target system**

After you have transferred the application files to a floppy disk, you must transfer them to the target system. To transfer the files to the target system, you must have the Service Creation Installation (SCI) tools installed on the target system.

   **Note:**
   If you select Network as the Type of Transfer in the Application Transfer Options window, you do not need to perform this procedure.

**Procedure**

To transfer files from floppy disk to the target system:

1. Log in to the target system as root.
2. Insert the floppy disk containing the application files into the floppy disk drive.
3. At the system prompt, enter some form of the following command:

   `scat [-o] application_name`

**Description**

- `-o` is an option. This option overwrites any files with matching filenames without warning you
- `application_name` is the name of the application

**Example**

To transfer the files for an application named acct_bal from floppy disks to the target system and overwrite files with matching filenames without warning you, enter:

```
scat -o acct_bal
```
Transferring application source files

IVR Designer provides the option to transfer your applications' source files to the target system. Even though the files cannot be used by the target system, this can be helpful should you need assistance from technical support personnel in troubleshooting your applications.

To transfer your application source files to the target system:

1. Open the Code Generation/Application Transfer window.
2. Click the Application Transfer tab.
3. From the Options menu, select Application Transfer.
   IVR Designer displays the Application Transfer Options window.
4. Check the Transfer Source checkbox.
5. Click OK.
   The Application Transfer Options window closes and the Application Transfer window's appearance changes. Notice that in addition to the Local Files and Remote Files panes, there is now a Folders pane.
6. Click a folder to select the type of source files you want to transfer.
   You can transfer the contents of only one folder at a time.
   The contents of the folder are displayed in the Local Files pane.
7. In the Local Files pane, select the files you want to transfer.
   Note: You cannot transfer application source files by simply selecting the folder. Once you select the folder, you must also select the files you want to transfer from that folder.
8. Click the Transfer button.
9. Repeat steps 6 through 8 for each additional set of application source files you want to transfer.

Application Transfer window menus

The Code Generation/Application Transfer tool window has its own set of menus. The following table describes the Application Transfer menus and their options.
Transferring files using a modem

Transferring files using a modem provides information about configuring PPP connections on PCs using the Windows 2000 operating system.

There may be times when you need to transfer application files to a target system, but you do not have a LAN connection and it is impractical to accomplish the transfer using floppy disks. In such cases, you can establish a network connection using a modem and then transfer the files using the same application transfer procedures you would use for a LAN transfer.

Configuring your system for PPP connections

To configure a PC using the Windows 2000 operating system for PPP connections, see Windows 2000 PPP configuration on page 156. You can establish network connections between machines and transfer application files using Network transfer options. For information about transferring application files over a network, see Transferring an application to the target system.

Note:
The modem network network connections described here may be used to conduct any type of TCP/IP activities you want. You are not limited just to transferring IVR Designer application files. You can, for example, use telnet,
Generating and transferring an application

FTP, or SFTP between the two computers using this connection. On IR systems using Solaris 10 as the operating system, SSH is provided by default. On IR systems using Solaris 8 as the operating system, SSH can be installed using the OpenSSH package. (http://www.sunfreeware.com). On Conversant V8 and V9, SFTP service is not supported. You will have to transfer the application using the FTP service.

Sample configuration scenario

The procedures in the following sections are based on the following example scenario:

Note:
This scenario is an example only. When establishing this connection on your own system, change the information to match your configuration needs. Before beginning the PPP configuration procedures, you need to know the IP addresses for both the PC and the target system.

- Windows-based PC being configured:
  
  Name: WinPC
  IP address: 192.0.0.1

- system to connect to:
  
  Name: ICex
  IP address: 192.0.0.2
  Phone number: 555-1212
  PPP login/password: mylogin / mypass
  Serial port used: tty01 (COM2)

Windows 2000 PPP configuration

To configure a PC running Windows 2000 for PPP connections.

Configure the TCP/IP settings

To configure the TCP/IP settings:

1. Open the Control Panel.
2. Double-click the Network and Dial-up Connections icon.
3. Double-click the Network connection you want to configure.
4. Verify that the TCP/IP protocol is installed. If it is not, click the Install button and then select TCP/IP Protocol.

   **Note:**
   You must have administrative privileges on your Windows 2000 machine before you can install this protocol.

5. Click OK.

**Create the Dial-Up Networking connection**

To create the Dial-Up Networking connection:

1. Open the Control Panel.
2. Double-click the Network and Dial-up Connections icon.
3. Click the Make New Connection button.
   The system displays the Network Connection Wizard dialog box.
4. Click Next.
   Select the Dial-up to Private Network connection type.
5. Click Next.
   Select the device to make the connection.
6. Enter the phone number of the Avaya IR machine to which you want to connect.
   For this example, the phone number is **555-1212**.
7. Click Next.
   The system displays the Connection Availability dialog box.
8. Select the Only for myself option.
   The system displays the Completing the Network Connection dialog box.
9. Enter a name for this connection.
10. Click Finish.
   The system saves this connection and adds it to the Network and Dial-up Connections and displays the Connect Dial-up Connection dialog box to modify the settings.

**Modify the Dial-Up Networking connection**

Before using the Dial-Up Networking connection, you must modify its settings.

To modify the Dial-Up Networking connection:

1. Open the Control Panel.
Generating and transferring an application

2. Double-click the Network and Dial-up Connections icon.
   The system displays the Network and Dial-up Connections dialog box.
3. Select the connection you just created in Create the Dial-Up Networking Connection.
   For this example, select ICex.
4. Click the Properties button.
5. Click the Networking tab.
6. Verify that the Type of dial-up server I am calling: field is set to PPP: Windows 95/98/NT/2000, Internet.
7. Verify that the Internet Protocol (TCP/IP) check box is checked.
8. If the File and Printer Sharing for Microsoft Networks check box is selected, clear it.
9. Verify that the Client for Microsoft Networks check box is checked.
10. Click the Properties button.
   The system displays the Internet Protocol (TCP/IP) properties dialog box.
11. Verify that the Obtain an IP address automatically and Obtain DNS server address automatically options are selected.
12. Click the Advanced... button.
13. Verify that the Use IP header compression check box is checked.
14. If your PC has an ethernet connection with a local network interface card (NIC) card, and you want to access systems on that network while you are connected using this PPP connection, make sure the Use default gateway on remote network check box is not checked.
15. Click OK (in the Advanced TCP/IP Settings dialog box).
17. Click the Security tab.
18. Verify that the Show terminal window check box is checked.
19. Click OK.

Open the PPP connection
To open and use the PPP connection:
1. Open the Control Panel.
2. Double-click the Network and Dial-up Connections icon.
   The system displays the Network and Dial-up Connections dialog box.
Transferring files using a modem

3. Select the connection you just created.
   For this example, select ICex.

4. Click the Dial button.
   The system dials the target system. When the target system responds, the system displays the Post.Dial Terminal Screen and prompts you to log in.

5. Enter the login for the target system. For this example, the login is mylogin.
   The system prompts you for the login password.

6. Enter the login password for the target system. For this example, the login password is mypass.
   The system prompts you for your termtype.

7. Enter PPP.
   The system displays a series of unreadable characters in the Post-Dial Terminal Screen window.

8. Click Continue.
   The system finishes establishing the network connection and closes the Post-Dial Terminal Screen window.

---

Transferring an application to the target system

Before you can transfer your application to the target system, you must first:

- Generate the application files using the Code Generation tool.
- Set the Application Transfer Options.

To transfer your application to the target voice response system:

1. Open the Code Generation/Application Transfer window.
2. Click the Application Transfer tab.
   The Code Generation/Application Transfer window displays the files available for transfer on the PC, and the destination folder on the target system.
3. Select the files you want to transfer.
   You can choose to look at or select only certain types of files. Use the drop-down menu below the Local Files pane to select the file types.
Generating and transferring an application

To select:

— A single file, click the filename in the Local Files pane.

— Multiple contiguous files, click the first filename, and shift-click the last filename.

— Multiple discontiguous files, click the first topic you wish to select, and control-click each other filenames you wish to select.

— All displayed files, click the Edit menu, and then click Select All.

4. If you want to view the files in the destination area before transferring your application, click the Display Files on Remote System or Floppy button on the Toolbar.

— If you are transferring over a network, IVR Designer displays all filenames on the remote host.

— If you are transferring to floppy disks, IVR Designer displays all filenames on the current floppy disk.

5. Click the Transfer button in the lower right corner of the window.

If you are transferring to floppy disks and you selected the Display Message When Files exist option in the Application Transfer Options window, IVR Designer displays the filenames of any files on the current floppy disk and then displays a warning dialog box with the message: Application files currently exist on A. Overwrite files?

Do one of the following:

— To cancel the Application Transfer operation, select No.

— To have IVR Designer write the files to the floppy disk, overwriting any files that have matching filenames, select Yes.

Note:

If you are transferring an application to a floppy disk, transfer only one application per floppy disk. You must still transfer the files from the floppy disk to the target system.

If you are transferring over a network and you selected the Display Message When Files exist option in the Application Transfer Options window, IVR Designer displays the filenames of all files on the target system and checks to see if the application name of the application you wish to transfer already exists there. If the name:

— Does not exist, IVR Designer completes the Application Transfer by writing the files to the target system.

— Exists, IVR Designer displays a warning message.
If you selected the **Display Message When Transfer is complete** option in the Application Transfer Options window, IVR Designer displays a Transfer complete message when the transfer is finished.
After you have created, generated, and transferred your IVR Designer application to the target system, you must assign the application to channels or telephone numbers before you can use it. For more information about and practice assigning a VoiceXML application from IVR Designer, see Assigning VoiceXML applications to channels or numbers on page 207. For information about assigning a VoiceXML application to channels or numbers from the target system, refer to the product documentation for the target voice response system.

This section includes the following topics:

Assigning a VoiceXML application to channels or numbers………162

Assigning a VoiceXML application to channels or numbers

After you have transferred a VoiceXML application to the target system, you can use IVR Designer to assign it to one or more channels or numbers on the target system.

Note:
Currently, IVR Designer can be used only to assign VoiceXML applications to channels or numbers on the Avaya Interactive Response system, or to display assignments.

Assign a VoiceXML application to channels

To assign a VoiceXML application to one or more channels on the target system:

1. Open the Code Generation/Application Transfer tool window.
2. Click the Application Assign tab.
3. (Optional, but recommended) Check to see if the channels to which you want to assign the application are available (do not have a service already assigned to them) on the target system. To check channels:
   a) Select Channel Services if it is not already selected.
   b) Enter the channel numbers.
Assigning a VoiceXML application to channels or numbers

c) Click **Display**.

The **Output** field displays information about the specified channels.

4. In the **Channel Number(s)** field, enter the numbers of the channels to which you want to assign the application.

5. Click **Assign**.

The system assigns the application to the specified channels and the **Output** field displays a message confirming the assignment.

**Assign a VoiceXML application to called or calling numbers**

To assign a VoiceXML application to called or calling numbers on the target system:

1. Open the **Code Generation/Application Transfer** tool window.

2. Click the **Application Assign** tab.

3. (Optional, but recommended) Check to see if the numbers to which you want to assign the application are available (do not have a service already assigned to them) on the target system. To check numbers:

   a) Select **Number Services** if it is not already selected.

   b) Enter the called or calling numbers.

   c) Click **Display**.

   The **Output** field displays information about the specified numbers.

4. Specify the numbers to which to assign the application by doing one of the following:

   — In the **Called Number(s)** field, enter the DNIS numbers to which to assign the application.

   — In the **Calling Number(s)** field, enter the ANI numbers to which to assign the application.

**Note:**

To assign an application to called or calling numbers, Dialed Number Identification Service (DNIS) or Automatic Number Identification (ANI) service must be available on the switch.

5. Click **Assign**.

The system assigns the application to the specified numbers and the **Output** field displays a message confirming the assignment.
VoiceXML application development tours

The VoiceXML application development tours provide detailed examples to help you learn how to use Avaya IVR Designer features in your VoiceXML applications.

- Basic VoiceXML application development tours cover some of the most common features that are used in many applications. These tours provide step-by-step instructions to walk you through completing the applications.

- Advanced VoiceXML application development tours cover some of the features that are used in more complex applications. These tours provide sample call flows and general instructions for completing the applications.

To continue, see the Basic VoiceXML application development tours on page 165 tours.

This section includes the following topics:

Basic VoiceXML application development tours ........................... 165

Advanced VoiceXML application development tours ....................... 272
Basic VoiceXML application development tours

The Basic VoiceXML application development tours walk you through some of the key tasks that you perform in creating Avaya IVR Designer VoiceXML applications. It is recommended that you complete these tours in sequence, if possible.

In some of these tours, you build on applications from other tours or on VoiceXML tour applications you loaded with IVR Designer. The tour applications are contained in the Program Files/Avaya/Avaya IVR Designer/Tour Applications/VoiceXML folder.

You can view and print the entire Basic VoiceXML application development tours section in PDF format.

To continue, see VoiceXML IVR Designer Interface on page 166.

This section includes the following topics:

VoiceXML IVR Designer Interface................................................. 166
Configuring the VoiceXML IVR Designer desktop......................... 173
Creating a basic VoiceXML application....................................... 181
Testing a VoiceXML application .................................................. 199
Generating and assigning VoiceXML applications ......................... 202
Branching based on time, day, or date in a VoiceXML application 212
Copying nodes and VoiceXML applications .................................. 219
Presenting menus in VoiceXML applications .............................. 224
Collecting information in VoiceXML applications ......................... 233
Using calculations in VoiceXML applications .............................. 240
Using variables and speech in VoiceXML applications .................. 247
Using embedded call flows in VoiceXML applications .................. 263
VoiceXML application development tours

VoiceXML IVR Designer Interface

The VoiceXML IVR Designer tour introduces you to the IVR Designer interface. It provides an overview of the key elements of the interface used to develop VoiceXML applications. Throughout the tour, you may click links to go to the Avaya IVR Designer Help, which provides detailed information.

To continue, see VoiceXML menus on page 166.

VoiceXML menus

The IVR Designer menu bar, shown in the following figure, provides access to the menus that you use to develop applications.

File Edit View Window Tools Help

The menu bar is always visible on the interface. Most functions on the menu bar are also accessible through the Toolbar.

File Menu

Use the File menu options, shown in the following figure, to create and manage application files.

<table>
<thead>
<tr>
<th>Option</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td></td>
</tr>
<tr>
<td>Open…</td>
<td></td>
</tr>
<tr>
<td>Open Script Builder Application</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Ctrl+S</td>
</tr>
<tr>
<td>Save</td>
<td></td>
</tr>
<tr>
<td>Save As…</td>
<td></td>
</tr>
<tr>
<td>Print…</td>
<td></td>
</tr>
<tr>
<td>1 C:\PROGRAM FILES\VOICE@WORK\APPS\XML\TEST1\TEST1.gdp</td>
<td></td>
</tr>
<tr>
<td>2 C:\PROGRAM FILES\VOICE@WORK\APPS\MENU\MENU.gdp</td>
<td></td>
</tr>
<tr>
<td>3 C:\PROGRAM FILES\VOICE@WORK\APPS\XML\VOICEIN\VOICEIN.gdp</td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td></td>
</tr>
</tbody>
</table>

Note the following when using the File menu:

- Only one IVR Designer application may be open at a time.

Edit Menu

Use the Edit menu options, shown in the following figure, to:
• Make changes to files, the Toolbar, and palettes.

• Find information in an application based on name, text, or use.

<table>
<thead>
<tr>
<th>Undo</th>
<th>Ctrl+Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redo</td>
<td>Ctrl+Y</td>
</tr>
<tr>
<td>Cut</td>
<td>Ctrl+X</td>
</tr>
<tr>
<td>Copy</td>
<td>Ctrl+C</td>
</tr>
<tr>
<td>Paste</td>
<td>Ctrl+V</td>
</tr>
<tr>
<td>Configure Toolbar...</td>
<td></td>
</tr>
<tr>
<td>Configure Node Palette...</td>
<td></td>
</tr>
<tr>
<td>Configure External Function Palette...</td>
<td></td>
</tr>
<tr>
<td>Find...</td>
<td></td>
</tr>
</tbody>
</table>

Note the following about the Edit menu:

• **Find** searches across all call flows in the current application. It does not search applications that are closed.

• To configure the **Toolbar** and **Node Palette**, you make choices about which icons to include and their order. You also may move and re-size the Toolbar and other palettes.

**View Menu**

The **View** menu, shown in the following figure, controls which elements of the interface are displayed.

| ✓ Toolbar |
| ✓ Node Palette |
| ☐ Globals |
| ☐ Variables |
| ☐ Phrases |
| ☐ Prompts |
| ☐ VXML Grammars |

Note the following about using the View menu:

• Clicking menu items for the Toolbar or another palette displays or hides the palette. Check marks indicate that the palette is displayed.

• Clicking **Globals** displays or hides the **Globals Manager**.

• Clicking a resource manager (for example, Variables or Prompts) displays or hides the resource manager, if it is undocked. Managers must be undocked before you can display them individually.

**Window Menu**
VoiceXML application development tours

Use the **Window** menu, shown in the following figure, to set up how call flow windows are displayed.

<table>
<thead>
<tr>
<th>Cascade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile</td>
</tr>
<tr>
<td>Arrange Icons</td>
</tr>
<tr>
<td>Minimize All</td>
</tr>
</tbody>
</table>

Note the following about using the Window menu:

- The menu appears only if you have selected the **Multiple Document Interface** option in the **Preference Editor**.
- When you display call flow windows using an option in this menu, you can switch back and forth between the windows easily.

**Tools Menu**

Use the **Tools** menu, shown in the following figure, to display tools that you use to develop, review, test, generate, and transfer applications.

<table>
<thead>
<tr>
<th>Call Flow Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Generation / Application Transfer</td>
</tr>
<tr>
<td>Node Inspector</td>
</tr>
<tr>
<td>Preference Editor</td>
</tr>
<tr>
<td>Simulation</td>
</tr>
<tr>
<td>Verify Design</td>
</tr>
</tbody>
</table>

Note the following about the Tools menu:

- Tools listed here are specific to the type of application you are developing and its state. You will become very familiar with these tools.
- Most of these tools are also available as icons on the Toolbar.

**Help Menu**

The **Help** menu, shown in the following figure, provides access to the **Avaya IVR Designer Help** topics.

| Contents... F1 |
| Index |
| About... |

To continue, see **VoiceXML Toolbar** on page 169.
VoiceXML Toolbar

Icons on the **Toolbar**, shown in the following figure, provide quick access to functions that you use to develop applications, such as opening and closing application files, editing, running simulations and so forth. You can also access most Toolbar functions through the IVR Designer Menu bar. See **Toolbar icons** on page 342 for explanations of each Toolbar icon.

![Toolbar Icons](image1.png)

**Toolbar configuration**

You can:

- Undock the Toolbar from its default location (under the Menu bar), move it to another location, size it, and hide it.
- Dock the Toolbar to its default location.
- Select the icons you want to include on the Toolbar and select the order in which they are arranged.

[Configuring the VoiceXML IVR Designer desktop on page 173](#) provides practice in configuring the Toolbar.

To continue, see **VoiceXML Node Palette** on page 169.

VoiceXML Node Palette

Icons on the **Node Palette**, shown in the following figure, represent the node objects that you use to build call flows for your applications. Examples of node objects include Disconnect Call and Announcement.

![Node Palette](image2.png)

To add a node object, drag it onto the design pad. Node objects save time because the programming for the function represented by the icon is done for you.
Node Palette configuration

You can:

- Undock the Node Palette from its default location under the Menu bar, move it to another location, size it, and hide it. (The Node Palette shown above is undocked.)
- Dock the Node Palette to its default location.
- Select the icons you want to include on the Node Palette and select the order in which they are arranged.

Configuring the VoiceXML IVR Designer desktop on page 173 provides practice in configuring the Node Palette.

To continue, see VoiceXML Globals on page 171.
VoiceXML Globals

**Globals** are a set of resources that you use when developing applications. Each resource has values that you can customize. For instance, the Variables Manager has values for system variables and custom variables. On the IVR Designer interface, Global resources may be listed together in their own window, as shown below. This window, shown in the following figure, is called the **Globals Manager**.

---

**Globals configuration**

You can:

- Reduce or enlarge the Globals Manager.
- Adjust and add tab stops to the Globals Manager.
- Undock individual resource managers from the Globals Manager, move them to other locations, size them, set them to remain on top of other windows, and hide them.
- Dock resource managers to theGlobals Manager.

Configuring the VoiceXML IVR Designer desktop on page 173 provides practice in configuring both the **Globals Manager** and individual resource managers.
### Variables Manager

Variables are grouped together in the **Variables Manager**, shown in the following figure. You use variables to collect and hold data, act as counters, and use system information.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Type</th>
<th>Spoken As</th>
<th>Default</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ANI]</td>
<td>character</td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>[Attempts]</td>
<td>number</td>
<td>number</td>
<td>0</td>
<td>true</td>
</tr>
<tr>
<td>[Channel]</td>
<td>number</td>
<td>number</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>[CollectedDigits]</td>
<td>character</td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>[Date]</td>
<td>date</td>
<td>date:ymd</td>
<td>20020718</td>
<td>true</td>
</tr>
<tr>
<td>[DeliveryTime]</td>
<td>character</td>
<td>duration:m</td>
<td>30</td>
<td>false</td>
</tr>
<tr>
<td>[DNIS]</td>
<td>character</td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>[HOURS_CLOSED]</td>
<td>number</td>
<td>number</td>
<td>0</td>
<td>true</td>
</tr>
<tr>
<td>[idigits]</td>
<td>number</td>
<td>number</td>
<td>0</td>
<td>true</td>
</tr>
</tbody>
</table>

Note the following about using the **Variables Manager**:

- You can create new variables and change the values and settings of system variables.
- System variables are predefined and are included as part of the IVR Designer package.

### Prompts Manager

In IVR Designer, prompts are components of node objects. You use prompts to instruct callers and get information from them. The **Prompts Manager**, shown in the following figure, groups and displays all defined prompts.

With the **Prompts Manager**, you can:

- Create new prompts.
- Rename and revise existing prompts.
Phrases Manager

You use the **Phrases Manager** to work with the phrase tables that store standard and custom phrases for use in IVR Designer applications. The **Phrases Manager**, shown in the following figure, displays one phrase table at a time.

With the **Phrases Manager**, you can:

- Create new phrases and edit existing phrases.
- Record phrases or import previously recorded phrases.

VXML Grammars Manager

With **VXML Grammars**, shown in the following figure, you can specify words to be recognized by the application.

With the **VXML Grammars Manager**, you can:

- Create, modify, and rename grammars.

To continue, see **Configuring the VoiceXML IVR Designer desktop** on page 173.

---

Configuring the VoiceXML IVR Designer desktop

When you create applications in IVR Designer, you can configure visual elements of the interface to create a desktop, as you do with other graphical user interfaces. You configure these elements to suit:
VoiceXML application development tours

- Your individual preferences
- Application and project requirements

This tour gives you practice in configuring the IVR Designer desktop.

To configure the desktop, you:

- Size, move, hide, and display elements of the interface.
- Establish application parameters and other environmental preferences using the Preference Editor.

To continue, see Configuring VoiceXML interface elements on page 174.

Configuring VoiceXML interface elements

IVR Designer provides great flexibility for configuring the visual elements of the interface, called the desktop. You can move, size, hide, and display elements according to your preferences and current requirements.

Some of the configurable elements are displayed automatically when you start IVR Designer. To display others, you must open a new or existing application and select the items you want to display from the Toolbar. Certain elements may be docked within other elements or undocked and moved. Before you begin configuring these elements, it is helpful to understand their characteristics.

Sizing

As with any Windows interface, you may minimize, maximize, and close elements by clicking the appropriate buttons. You size and shape windows by clicking and dragging their borders.

Docking and undocking

Interface elements that are docked within another element may be undocked to another location. When undocked, these elements are treated as separate entities. You may open them, close them, and move them freely about the screen. To undock and dock elements, click them in the appropriate points, which vary depending on the type of element.

- To undock a resource manager in the Globals Manager, click its name bar, shown in the following figure, and drag the resource to another location. To dock a resource manager, click the name bar again and drag the resource back to the Globals Manager. If you click the title bar of a resource manager, shown in the following figure, and drag the window, the resource manager moves but does not dock. When you dock a resource manager, it goes to the end of the Globals Manager list.
To undock the Toolbar, click a blank area or an inactive icon and drag the Toolbar away. To dock the Toolbar, click it as in undocking, drag the Toolbar over the menu bar, and release it. If you do not place the Toolbar over the menu bar, it does not dock. You dock and undock the Node Palette in the same way.

**Displaying and hiding**

Except for the menu bar, you may hide elements on the IVR Designer interface. To hide elements, click the close box on the window. In the case of docked resource managers, you must first undock them. To display a hidden element, select it from the View menu or click the associated icon.
Desktop example

The example IVR Designer desktop, shown in the following figure, is set up to meet the requirements of a particular user. The figure shows how elements of the workspace can be adjusted.

A. You can click \( \rightarrow \) to enlarge the **Globals Manager** and click \( \leftarrow \) to reduce it.

B. You can minimize, maximize, and close the **Globals Manager**.

C. The **Toolbar** may be docked (shown above) or undocked and moved to another location.

D. Individual resource managers may also be undocked and treated as separate entities.

E. If you are using **Multiple Document Interface** mode, you can move individual call flow windows within the design pad. The design pad shown in this figure uses the tabbed interface.
F. The **Node Palette** may be undocked and treated as a separate entity. You may find it convenient to place it in the lower-right corner of the design pad, as shown.

**Desktop Practice**

Practice configuring the IVR Designer desktop:

1. Open a sample IVR Designer application:
   a) Click **Start > Programs > Avaya > Avaya IVR Designer > SampleApps > VoiceXML Applications**.
   b) Select a folder for any sample application and double-click it to open it.
   c) Double-click the icon next to the sample application file to launch IVR Designer and the sample application.

2. If you do not see the **Globals Manager** window, click the **Globals Manager** icon to display it.

3. Adjust the **Globals Manager**:
   a) Click [enlarge] and [reduce] to change the size of the Globals Manager.
   b) Undock a resource on the Globals Manager by clicking its name bar and dragging it to another location. Then dock the resource by clicking its name bar and dragging it back.
   c) Maximize the Globals Manager. Then adjust the column widths for a resource manager by clicking and dragging the lines on either side.
   d) Click the title bar of the Globals Manager and move it to the side of the screen.
   e) Close the Globals Manager and then redisplay it by clicking the Globals Manager icon.

4. Adjust the **Toolbar**:
   a) Click a blank area on the the Toolbar to undock it. Then drag it to another location.
   b) Click and drag the borders of the Toolbar to adjust its size.
   c) Dock the Toolbar by dragging it over the menu bar and releasing it.

5. Adjust the **Node Palette**:
   a) If the Node Palette is not displayed, click **View > Node Palette**.
   b) If the Node Palette is docked, undock it by clicking it on a blank area and dragging it to another location.
c) Click and drag the borders of the Node Palette to adjust its size.

d) Close the Node Palette and redisplay it by clicking View > Node Palette.

e) Dock the Node Palette.

6. Display and adjust the Node Inspector:

a) Click the Node Inspector icon unless the Node Inspector is already displayed.

b) Move the Node Inspector by clicking its name bar and dragging it to another location.

c) Click and drag the borders of the Node Inspector to adjust its size.

d) Close the Node Inspector.

7. Continue configuring the desktop until it looks the way you want.

— When you work on applications, you can configure the desktop according to your current requirements. For instance, you can hide elements you are not using, enlarge elements that you need, or set certain windows to stay on top.

— To save the desktop so that it looks the same when you return to the application or when you start IVR Designer the next time, select the appropriate Save option in the Environment tab of the Preference Editor.

To continue, see Setting VXML preferences on page 178 for practice in setting up the Preference Editor.

Setting VoiceXML preferences

The Setting VoiceXML Preferences topic introduces you the the Preference Editor and the options it provides.

1. Click the Preference Editor icon on the Toolbar.
Basic VoiceXML application development tours

IVR Designer displays the **Preference Editor**, shown in the following figure.

![Preference Editor](image)

2. Review options in the **Application Parameters** tab and make any desired changes.
   - You may wish to save your applications to a directory other than the one shown.
   - The default language typically is set at installation and not changed.
   - You can choose to display warnings for features that are unsupported on the Avaya IR platform or on CONVERSANT platforms.
3. Click and review the **Environment** tab, shown in the following figure, and make any desired changes.

---

The **Opening/Saving** options relate to saving the current workspace configuration. If you like the way the workspace is configured, select the option to save it on exit.

---

You probably want to save your applications at regular intervals, such as every 10 minutes.

---

Spell checking requires that Microsoft Word with spell checking is installed on your PC. Language defaults to the Word spell checker language.

---

You use **Call Flow Editor** options to adjust the appearance of the interface and the options it presents for call flows.

4. Click and review the **Printing** tab.

---

Fields in this tab set printing options for application components, such as call flows and nodes.

---

For printing to occur, the appropriate option must be selected and the component must be present in the application.
If you do not choose to send application output to a default printer, it goes to a document file.

To continue, see Creating a basic VoiceXML application on page 181.

Creating a basic VoiceXML application

The topics in Creating a basic VoiceXML application provide a step-by-step example of how to design and build a simple VoiceXML application using IVR Designer.

To continue, see VoiceXML application design overview on page 181.

VoiceXML application design overview

IVR Designer can be used to develop VoiceXML applications that can be run on the CONVERSANT R9 or Avaya IR system. The recommended workflow for producing applications using IVR Designer is:

1. Determine application requirements.
2. Design the application on paper, identifying all of the requirements and planning for all contingencies.
3. Develop the application according to specifications, using IVR Designer.
4. Verify the application.
5. Generate the source code file to transfer to the target Avaya IR system.
6. Transfer the application from your PC to the target system.
7. Install the application on the target system.
8. Dial in and test the application on the target system.
9. Modify the design specification or the application and repeat code generation through testing if necessary.

When you complete all these steps, you have a working voice application that meets your client’s needs.

Scripting an application

You may not be the person who develops the requirements for the application. Instead, you may receive the requirements from someone else. When you receive application requirements from someone else, you may need to refine the requirements. Take some time to think about the details of how the application should work. Add any missing requirements and make existing requirements clearer, if necessary.
Specifying a voice application has one advantage over many types of requirements statements. Because a voice application automates a conversation, you can imagine the script of a telephone call between a caller and a live agent, write down the steps, and transpose them into a logical analysis that assigns roles or tasks to the caller and the agent.

Whether you receive an application requirements statement from someone or develop it yourself, you start by interpreting the requirements as a scripted conversation. Imagine, and note, all probable communication between the agent and the caller. Try listening to live agents and noting what they say and do. Listening to agents is especially helpful if the application must automate a type of interaction you’ve never heard.

Suppose that one of the departments in your bank has asked you to develop a simple application. It greets the caller and presents two options:

- Provide information on auto loan interest rates.
- Transfer the caller to someone for assistance in obtaining an auto loan.

Translate these requirements into a typical call:

- Agent: Hello, Big Bank. How may I help you?
  **Announce greeting, prompt for information.**

- Caller: What’s your current interest rate on auto loans?
  **State request for information.**

- Agent: That’s 7.9%. May I help you with anything else?
  **Announce requested information, prompt for information.**

- Caller: Well, I’d like to speak to somebody about applying for an auto loan.
  **State request for information.**

- Agent: One moment and I’ll transfer you. (Transfers call then hangs up.)
  **Announce transfer, transfer call appropriately, disconnect, and quit.**

Now let's look at our sample banking call again. This time, we analyze the actions that are taking place. These actions form the basis of the script.

The action steps are noted in **bold**.

- Agent: Hello, Big Bank. How may I help you?
  **Announce greeting, prompt for information.**

- Caller: What’s your current interest rate on auto loans?
  **State request for information.**

- Agent: That’s 7.9%. May I help you with anything else?
  **Announce requested information, prompt for information.**

- Caller: Well, I’d like to speak to somebody about applying for a loan.
  **State request for information.**

- Agent: One moment and I’ll transfer you. (Transfers call then hangs up.)
  **Announce transfer, transfer call appropriately, disconnect, and quit.**
1. Agent makes announcement
2. Agent prompts the caller for information.
3. Caller requests information.
4. Agent reports the information to the caller.
5. Agent repeats steps 1 through 4 until the caller's request requires call transfer.
6. Agent announces transfer
7. Agent transfers the call
8. Agent hangs up.

On the highest level, these are the requirements for an IVR Designer application, and they translate into action steps.

On a more detailed level, you would note the specific types of information the agent must prompt for and specific actions he or she must take. In another tour you get practice in working with more detailed requirements statements, and developing more complex applications.

**Developing a step analysis**

To determine the voice response capabilities required by your application, you should outline each possible step in the call transaction, indicating whether the Agent or Caller takes that step.

Here is a logical step analysis of our sample conversation between a bank agent and a caller:

<table>
<thead>
<tr>
<th>STEP</th>
<th>Agent</th>
<th>Caller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Announce greeting.</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Prompt for information.</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>State request for information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Announce requested information.</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Prompt for information.</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>State request for information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Announce transfer.</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Transfer call appropriately.</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Disconnect.</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

This logical step analysis describes the flow of the call transaction. When you break a call down by speaker, it is easy to include all of the interactions between the caller and the agent.
Creating a flowchart for an application

Because an IVR Designer VoiceXML application must describe the flow of an automated call transaction to the Avaya IR or CONVERSANT system, a step analysis is a useful tool for application developers, particularly when refined into graphic form as a flowchart.

You may be familiar with the standard flowcharting symbols, shown in the following figure. They are:

- **Process**
- **Decision**
- **Pre-defined Process**
- **Terminator**
- **Connector**
- **Off-page connector (match the numbers)**

Modified versions that can be used to specify IVR Designer call flows are shown in the following figure:

- **Node Type**
- **Decision: branch, based on condition, input, value of variable**
- **Encapsulated call flow**
- **Title of call flow, default terminator of call flow, end of application**
- **Jump to node name**
- **Off-page connector (match the numbers)**
With these modified symbols, the flowchart for the step analysis would look like this:

![Flowchart Image]

You may use these or other symbols when you create a flowchart. In fact, you may choose not to make a flowchart at all. Instead, you may develop the application directly from the step analysis.

What is important is that you follow an organized process for planning an application. A good design process helps prevent confusion during development of a complicated application. With a good plan, you are less likely to omit important details. These details are important because they make your application operate smoothly.

To continue, see Creating a VoiceXML application on page 185.

Creating a VoiceXML application

The application you develop in Creating a VoiceXML application does the following:
VoiceXML application development tours

- Plays a greeting announcement to the caller.
- Transfers the call to another number and announces the transfer to the caller.
- Before disconnecting the call, notifies the caller if the call cannot be transferred successfully.
- Disconnects and terminates the call.

You begin your application development by adding nodes.

Creating a new VoiceXML application

Before you can begin building your first application, you must create it. To create a new VoiceXML application:

1. Click the New icon on the Toolbar.

   IVR Designer displays the New Avaya IVR Designer Application window, shown in the following figure. Notice that IVR Designer suggests a default name.

2. Replace the default name with vfirst.
   
   If you are creating both TAS and VoiceXML applications, you can create a separate VXML folder under the Apps folder.

3. Click OK.
IVR Designer creates a new application called \texttt{vfirst} and displays the Select Application Type window, shown in the following figure.

![Select Application Type](image)

This window is used to select a TAS or VoiceXML application type.

4. Click \texttt{VXML Application} and click \texttt{OK}.

5. Configure your workspace so you can view the design pad and the node palette.

   If necessary, select View > Node Palette to bring the node palette into view.

You are now ready to begin developing your application. The most basic task in developing an application is building one or more call flows by dragging the appropriate nodes onto the design pad.

Most TAS applications start with a node that answers the call. With VoiceXML, the call is answered before the application is started.

**Adding an Announcement node**

To add an Announcement node:

1. Locate the Announcement node on the Node Palette and drag it onto the design pad.

   IVR Designer displays the node naming dialog box, shown in the following figure. If the Automatically name nodes check box is selected in the Preferences Editor, IVR Designer suggests a name for the node.

   ![Name the Announcement node](image)

2. Type \texttt{say hello} in the name field.
When the node naming dialog box opens, the suggested name is highlighted. To replace it, simply type the new name. If the suggested name is not highlighted, double-click it to highlight it. Then replace the suggested name by typing the new name.

3. Click OK.

IVR Designer adds the new node to the design pad, as shown in the following figure, and may open the Node Inspector window.

Notice that IVR Designer uses upper case for the initial character of each word and removes the spaces between words.

Many nodes require further definition after they have been placed in the call flow. You can choose to complete these definitions either when you place the node in the call flow, or later on, after placing all the nodes in the call flow. This is a matter of personal style. For the purpose of this example, you define the nodes after placing all nodes in the call flow.

**Adding a Transfer Call node**

To add a **Transfer Call** node:

1. Locate the Transfer Call node on the Node Palette and drag it onto the empty branch of the Announcement node.

2. Type `transfer call` in the name field.

3. Click OK.

IVR Designer adds the new node to the call flow, shown in the following figure, and changes the display in the Node Inspector window.

Check carefully the appearance of the call flow with this new node added. If you did not drop the Transfer Call node onto the empty branch of the **SayHello** node, the **TransferCall** node is not connected. If you accidentally drop the node in the wrong place, you can drag and drop it again to place it correctly.
If the nodes in a call flow are not connected, the application works only as far as the disconnected node. Any nodes after the disconnected node do not execute.

Notice that the TransferCall node has two empty branches. The first branch is the pass branch, which is the branch that is taken if the node is executed successfully. In this case, it means that the call is transferred successfully.

The second branch, with the X symbol, is the fail branch. This is the branch that is taken if the call cannot be successfully transferred.

**Adding a Disconnect Call node**

To add a Disconnect Call node:

1. Locate the Disconnect Call node on the Node Palette and drag it onto the empty pass branch of the Transfer Call node.
   
The pass branch is the branch indicated by the arrow.
2. Type **hangup** in the name field.
3. Click **OK**.

IVR Designer adds the new node to the call flow, shown in the following figure, and changes the display in the Node Inspector window.

---

**Adding a Return node**

To add a Return node:

1. Locate the Return node on the Node Palette and drag it onto the empty branch of the Disconnect Call node.
2. Type **quit** in the name field.
3. Click **OK**.
IVR Designer adds the new node to the call flow, shown in the following figure, and changes the display in the Node Inspector window.

![Diagram showing a call flow with nodes: SayHello, TransferCall, Hangup, Quit, and TransferFail.]

The purpose of the Return node is not as obvious as the purpose of other nodes you have used. In the [Main] call flow, the Return node terminates the application script and allows the target system to reset itself for another caller. Notice that the Return node has no branches beneath it.

**Adding a Transfer Fail Announcement node**

To add another Announcement node:

1. Drag an Announcement node to the empty fail branch of the Transfer Call node.
   - The fail branch is indicated by the X symbol.
2. Type **transfer fail** in the name field.
3. Click **OK**.

IVR Designer adds the new node to the call flow, shown in the following figure, and changes the display in the Node Inspector window.

![Diagram showing the updated call flow with the TransferFail node.]

This node is executed only if the call cannot be transferred. The purpose of the announcement is to notify the caller that the call could not be transferred before disconnecting the call.
Using the Jump to Node function

If the call fails to transfer, the application should disconnect the call and end the script, much as it did for the successful call transfer.

You could add another Disconnect Call node and another Return. However, since you already have those nodes in place, you can simply instruct the application to jump to the Disconnect Call node and continue from there, whenever the fail branch of the Transfer Call node is executed.

To direct the system to hang up the call if the call transfer cannot be completed:

1. Right-click the empty branch of the **TransferFail** Announcement node.

   IVR Designer displays a pop-up menu of options, shown in the following figure.

   | Cut  
   | Copy  
   | Paste  
   | Branch Notes  
   | Connect Node  
   | Edit Node  
   | Jump To Node  
   | Print Call Flow  
   | Verify Call Flow  
   | Close  
   | Open  

   One of IVR Designer’s powerful features is the extensive use of right-click pop-up menus. Most windows, nodes, and tools have them.

2. Click the **Jump To Node** option.
IVR Designer displays the Nodes dialog box, shown in the following figure.

This dialog box displays all the nodes used in the current call flow.

3. Click the **Hangup** node.

4. Click **OK**.

IVR Designer changes the appearance of the call flow, shown in the following figure, to show that the empty branch now points to the Disconnect Call node.

The ➔ symbol indicates that the node branch jumps to another node.

To continue, see **Specifying parameters for a VoiceXML application** on page 193.
Specifying parameters for a VoiceXML application

Many nodes require definition before they are fully functional in a call flow. This definition can take the form of defining prompts, selecting variables, or defining argument values. These tasks are accomplished using the Node Inspector.

In this application, a continuation of the application from Creating a VoiceXML application on page 185, the following node edits are required:

- Define the initial announcement prompt to be played to the caller when the call is answered.
- Set the telephone number transfer options for the Transfer Call node.
- Define the announcement to be played to the caller if the call cannot be transferred.

Defining an Announcement node prompt

As it stands now, the application is set to play an announcement to the caller. What to say in that announcement has not been defined. You must define the Initial Prompt that contains the announcement to play to the caller.

To define the Initial Prompt:

1. Click the **SayHello** Announcement node.

   IVR Designer highlights the Announcement node. The Node Inspector moves to the background. Depending on how you have configured your workspace, it may now be either partially or totally hidden by the main window.
2. Open the **Node Inspector** window, shown in the following figure.

If the Node Inspector is totally hidden by the main window, you must bring it to the front. To avoid this extra step, position the Node Inspector so that its window is visible outside the main window. Then you simply click the Node Inspector window to bring it to the front. Resize the window if needed to view all the fields.

If you need to bring the Node Inspector to the from, perform one of these actions:

- Click the Node Inspector icon on the Node Palette.
- Select **Node Inspector** from the **Tools** menu.
- Right-click the node you want to edit and select **Edit Node** from the pop-up menu.

3. Double-click the **Initial Prompt** field.

IVR Designer displays the **Name Prompt** dialog box, shown in the following figure.

4. Accept the suggested name in the prompt name field by clicking **OK**.
5. In the **Prompt Text** field, type the following text:

   **Hello, world!**

   This becomes a phrase and is a part of the custom phrase table for this application.

   Unless you have Text-to-Speech installed on your target voice response system, the text you type here does not get played back or synthesized into audible speech in IVR Designer. This text is primarily for phrase identification purposes. To use recorded speech files to play back your phrase text during Simulation and on the target system, you must provide separate speech sound files.

6. Click **OK**.

   IVR Designer displays the **New Phrase Assignments** window, shown in the following figure.

   ![New Phrase Assignments Window](image)

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Phrase Table</th>
<th>Old Text</th>
<th>New Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>[new]</td>
<td>[PhraseTable]</td>
<td></td>
<td>Hello, world!</td>
</tr>
</tbody>
</table>

7. Click **OK to** accept the defaults.

   IVR Designer performs the following actions:
VoiceXML application development tours

— Creates a new prompt named [SayHelloInitialPrompt].
— Places the name of the prompt in the Node Inspector Initial Prompt field.
— Creates a new phrase and automatically assigns it a phrase number.
— Opens the Globals Manager, shown in the following figure, and displays the new prompt and new phrase in their respective global resource manager windows.

You will see later how this prompt and phrase text are used in verifying and simulating your application.

Setting Transfer Call node parameters

After playing the announcement, the application is set to transfer the call to another number. You must define how to handle the transfer by setting the Transfer Call node parameters.

To set the Transfer Call node parameters:

1. Click the TransferCall node on the design pad.
2. Click the **Node Inspector** window to bring it to the front.

3. If you plan to test this application on a working Avaya IR or CONVERSANT system, click the **Phone Number** field of the Node Inspector and enter **1234**, as shown in the following figure, or any valid telephone number.

![Node Inspector window](image)

Notice that the **Transfer Type** field says **Blind**. This means that the call is transferred without the system checking for call completion.

Notice also that this node has two sets of options represented by two tabs located near the bottom of the Node Inspector window (**Transfer Options** and **Failure Treatment**). For now, accept the defaults.

**Defining a Transfer Call node failure announcement**

The application is now set to play a simple announcement and transfer the call to another number. But what if the call cannot be transferred?

When a call cannot be transferred, the application directs the call to the fail branch of the Transfer Call node, where you placed the **TransferFail** Announcement node. You must define the announcement to play when this fail branch is taken. Before it hangs up, you want the system to play an announcement warning that the call transfer could not be completed.
VoiceXML application development tours

The procedure for defining this announcement is very similar to what you did to define the announcement you created earlier.

To define the failed transfer announcement:

1. Click the TransferFail Announcement node.
2. Click the Node Inspector to bring it to the front.
3. Double-click the Initial Prompt field.
4. Click OK to accept the suggested name.
   IVR Designer displays the Prompt Editor window.
5. In the Prompt Text field, type the following text:
   
   I am sorry, your call could not be transferred. Please try again later.
   
   If you press Return after the first line, IVR Designer creates two phrases.
6. Click OK.
   IVR Designer displays the New Phrase Assignments window.
7. Click OK to accept the defaults, as you did before.
   IVR Designer performs the following actions:
   — Creates a new prompt named [TransferFailInitialPrompt].
   — Places the name of the prompt in the Node Inspector Initial Prompt field.
   — Creates a new phrase and automatically assigns it a phrase number.
   — Displays the new prompt and new phrase in their respective global resource manager windows in the Globals Manager.

You have just completed building your first application call flow. You should now have an idea of how to build your own call flows.

Once the application has been built, you must test and debug the application and generate the code that allows the application to run on the target voice response system. You must also transfer the generated code to the target voice response system and install it.

To continue and follow this example through the testing and debugging phase, see Testing a VoiceXML application on page 199.
Testing a VoiceXML application

Before generating code and transferring the application to the target system, you should always test the application using the **Verify Design** and Simulation tools included with IVR Designer.

The topics in *Testing a VoiceXML application* provide examples of how to verify and simulate an application.

**Note:**
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see *Verifying a VoiceXML application* on page 199.

Verifying a VoiceXML application

In *Verifying a VoiceXML application*, you continue to use the `vfirst` application that you created. If you did not complete this application, use the `svfirst` VoiceXML tour application provided with IVR Designer.

The **Verify Design** tool checks the application for possible errors or omissions in design.

1. Open the `vfirst` or `svfirst` application.

2. To verify the design using the **Verify Design** tool, click its icon on the Toolbar.

   The Verify Design tool automatically checks the application call flow and displays the results in the Verification Results window, shown in the following figure.

![Verification Results](image)

Check marks indicate possible errors or omissions that the Verify Design tool found.

Since the Verify Design tool identifies only possible errors or omissions, you must decide whether the flagged items really are errors or omissions. In some cases, the code
VoiceXML application development tours

generation for the application will not complete successfully unless the problem is corrected. In this case, you want to specify a message saying that the call is being transferred to another number. Otherwise, the caller might be confused by hearing clicks and silence.

Adding a Message to Caller
To correct this omission and define a message to play before transferring the call:
1. Double-click the red check mark, designated **Message to Caller is not specified**.
   IVR Designer locates the omission in the **TransferCall** node and highlights the node. At the same time, it locates the required field in the Node Inspector.
2. To close the **Verification Results** window, click **Close (X)**.
3. Double-click the **Message to Caller** field in the Node Inspector.
4. Accept the default in the **Prompt Name** field.
   IVR Designer displays the **Prompt Editor** window.
5. Type the following text in the **Prompt Text** field:
   **Please hold while your call is transferred.**
6. Click **OK**.
   IVR Designer displays the **New Phrase Assignments** window.
7. Click **OK**.
   The new prompt and new phrase are displayed in the Globals Manager.

To continue, see Simulating a VoiceXML application on page 200.

Simulating a VoiceXML application

Using the **Simulation** tool, you can simulate how an application will behave when installed on the target system. This tool can be very useful in finding application design problems.

To simulate the **vfirst** application using the Simulation tool:
1. Click the **SayHello** node to highlight it.
   A simulation always starts running from whatever node is highlighted at the time the Simulation tool is selected. Highlighting this node starts the simulation from the beginning of the call flow.
2. Click the Simulation icon on the Toolbar.
The Simulation tool starts running automatically. You can monitor the progress of the simulation in the Application Simulation window, shown in the following figure.

When the simulation reaches the Transfer Call node, the Simulation tool asks how to simulate the result of the attempted call transfer. If you select the Answer option, the Simulation tool simulates a successful transfer. If you select the Error... option, the Simulation tool simulates an unsuccessful transfer and takes the fail branch of the Transfer Call node.

3. Click Answer.

The Simulation tool completes the simulation of a successful call transfer.

4. To thoroughly test the application, always simulate as many possibilities as you can. In this case, there is only one other possibility: that the call transfer attempt was not successful. To test this, run the simulation from the AnswerCall node again, this time selecting the Error... option when the simulation reaches the Transfer Call node.

5. To close the Application Simulation window, click Close (X).
To continue and follow this example through generating the code, and transferring and assigning the application, see Generating Code and Transferring and Assigning VoiceXML Applications on page 202.

### Generating and assigning VoiceXML applications

After you have debugged an application and corrected any errors, you must generate code, transfer the application to the Avaya IR or CONVERSANT system, and assign the application to channels or numbers on the target system.

The topics in Generating and assigning VoiceXML applications show you how to complete these tasks.

**Note:**

This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Generating VoiceXML code on page 202.

### Generating VoiceXML code

Now that you have designed and tested the vfirst application, you can generate the code to be transferred to the target system.

To generate code:

1. Open the vfirst you created or the svfirst1 VoiceXML tour application.

2. Click the Code Generation / Application Transfer icon.

   If you have not saved the application, IVR Designer displays a dialog box asking whether you want to save it. Respond to this message.
IVR Designer displays the Code Generation tab, shown in the following figure, of the Code Generation / Application Transfer window.

This window displays the kinds of files you can generate from the current application. Notice that because you have no phrase sound files defined for this application, this option is inactive.

3. Click **Apply**.

IVR Designer automatically generates the code needed to make this application run on the target system and notifies you of any errors encountered during the code generation process. If the code is error-free, the system displays the message shown in the following figure.

4. Click **OK**.

To continue, see **Transferring VoiceXML applications** on page 204.
Transferring VoiceXML applications

Once the VoiceXML code is generated, transfer the application to the target system where it runs.

⚠️ Caution
If you do not have access to a separate development system, you may want to read Transferring VoiceXML applications without transferring the application to your system. Transferring training applications to your production system could affect performance.

IVR Designer provides two options for transferring your applications to the target system:

- On diskettes
- Over a network

Only network transfer is available on Avaya IR systems.

Setting the application transfer options

To set the application transfer options:

1. Open the **Code Generation / Application Transfer** tool window.
   
   You can click the Code Generation icon or select this option from the **Tools** menu.

2. Click the **Application Transfer** tab.
IVR Designer displays the **Application Transfer** tab, shown in the following figure. Before you transfer the application, define how the transfer should be done by specifying the transfer options.

3. From the **Options** menu, select **Application Transfer**.

IVR Designer displays the **Application Transfer Options** dialog box, shown in the following figure. You complete the information in this window before transferring the application.
4. In the **Type of Transfer** box, select the method for transferring the application from your PC to the target system.

   If you are connected to the Avaya IR or CONVERSANT system over a network, select **Network** for this tour.

5. To have IVR Designer notify you before overwriting any files with matching filenames, select the **Files exist** check box.

6. To have IVR Designer notify you when the transfer process has been completed, select the **Transfer is complete** check box.

7. To have IVR Designer transfer application source files along with the generated code, select the **Transfer Source** check box.

   You might want to transfer the source files if you want to archive them on your Avaya IR or CONVERSANT system.

8. Enter the transfer destination:

   If you are transferring over a network, enter information about the target system in the **Network** information fields:

   a) Enter the name or IP address of the Avaya IR or CONVERSANT system in the **Host:** field or select the host from the drop-down menu.

   b) Type the user login ID in the **Login:** field.

   c) Type the user password in the **Password:** field.

   If you are transferring on diskettes, select the drive you want from the **Floppy** drop-down menu. All available removable media drives appear in the drop-down menu. Do not select anything other than a diskette drive.

9. Click **OK**.

Transferring the Application

To transfer your application to the target voice response system:

1. In the **Application Transfer** tab of the Code Generation / Application Transfer window, select the files you want to transfer.

2. Click **Transfer** in the lower-right corner of the window.
The system displays the results of the application transfer, shown in the following figure.

If you are transferring to diskettes, you must transfer the files from the diskettes to the target system.

To continue, see Assigning VoiceXML applications to channels or numbers on page 207.

Assigning VoiceXML applications to channels or numbers

Once the code has been transferred to the target system, it must be assigned to channels or numbers before you can run the application.

⚠️ Caution

If you do not have access to a separate development system, you may want to read Assigning VoiceXML applications to channels or numbers without installing the application on your system. Installing training applications on your production system could affect performance.

You can install VoiceXML applications on the target system from within IVR Designer, using the Application Assign tab in the Code Generation / Application Transfer window.

In this topic, use the Application Assign tab to assign the application to one or more channels or numbers on the target system.
Assign the VoiceXML application to channels

To assign the VoiceXML application to channels on the target system using the Application Assign tab in the Code Generation / Application Transfer window:

1. Select the Application Assign tab, shown in the following figure, in the Code Generation / Application Transfer window.

2. Check to see if the channels to which you want to assign the application are available for assignment on the target system.
   a) Select Channel Services if it is not already selected.
   b) Enter the channel numbers.
   c) Click Display.
The **Output** field, shown in the following figure, displays information about the selected channel or channels.

3. Specify the channels to which to assign the application by entering the number or numbers of the channels in the Channel Number(s) fields.

4. Click **Assign**.
The system assigns the application to the designated channels and the **Output** field displays a message confirming the assignment, as shown in the following figure.

You can also assign the application to channels using the Web Administration tool on the Avaya IR system or cvis_menu on the CONVERSANT system.

**Assign the VoiceXML application to numbers**

To assign the VoiceXML application to ANI or DNIS numbers on the target system using the Application Assign tab in the Code Generation / Application Transfer window:

1. Select the **Application Assign** tab in the Code Generation / Application Transfer window.
2. Check to see if the numbers to which you want to assign the application are available for assignment on the target system.
   
   a) Select Number Services if it is not already selected.
b) Enter the called or calling numbers.

c) Click **Display**.

The **Output** field, shown in the following figure, displays information about the selected number or numbers.

![Code Generation / Application Transfer: VFIRST](image)

```
<table>
<thead>
<tr>
<th>CALLED NUMBERS</th>
<th>CALLING NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM 1000</td>
<td>T0 1999</td>
</tr>
<tr>
<td>2000</td>
<td>2999</td>
</tr>
<tr>
<td>any</td>
<td>any</td>
</tr>
</tbody>
</table>
```

To continue, see [Branching based on time, day, or date in a VoiceXML application](#) on page 212.
Branching based on time, day, or date in a VoiceXML application

A common type of voice application is one that identifies calls coming in on holidays, weekends, or after hours and transfers those calls to a message center.

The Time Branch node is used to branch to different actions based on time of day, day of week, date, or a variable value. In this tour, you create an application, named vincomng. You add a Time Branch node and specify branches for weekend, holiday, after hours, and business hour calls.

The vincomng application:

- Branches based on time, date, and day.
- Plays an appropriate message and takes action based on whether the call is received during business hours:
  - During business hours, plays a message and transfers calls to the operator.
  - Outside business hours, plays an appropriate message, transfers calls to the message center, hangs up and quits.

Note:
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Creating a VoiceXML phrase table on page 212.

Creating a VoiceXML phrase table

To create a phrase table to use in the application:

1. Create a new VoiceXML application, named vincomng.
2. Open the Globals Manager.
3. Right-click the Phrases Manager and select New Phrase Table...
The **Phrase Table Editor**, shown in the following figure, is displayed.

![Phrase Table Editor](image)

Name the Phrase Table **callin** and set **Starting Phrase Number** to **5000**. Select **Recorded Phrases** or **Text-to-Speech**, depending on your system, and click **OK**.

- If you select Recorded Phrases, you can specify a URI where the recorded phrases are located.

- If you select Text-to-Speech, you can select a language for speaking phrases. If no language is specified, the language for the application is used.
4. For each of the phrases listed in this step, right-click the Phrases Manager, select New, enter the text in the Phrase Editor, shown in the following figure, and click OK:

![Phrase Editor - "5000"](image)

Note:
Enter only the phrase text. The system assigns phrase numbers in sequence.

5000 Welcome to Big Bank.
5001 We are now closed.
5002 Your call is being forwarded to our twenty-four hour message center.
5003 Please wait while your call is being transferred.
5004 Thank you for calling.
5005 Goodbye.
5006 Please try again later.
5007 Happy New Year.
5008 You have reached the Big Bank information line.
5009 Your call cannot be transferred at this time.

To continue, see Creating a VoiceXML Time Branch application on page 214.

Creating a VoiceXML Time Branch application

To continue to create the vincomplex application with Time Branch nodes:
1. Drag a Time Branch node onto the design pad and name it **TimeBranch**, as shown in the following figure.

![Time Branch Node](image)

When you add a Time Branch node, IVR Designer creates one branch named *(Other)*. You insert additional branches to meet the requirements of your application. For each branch, you specify the conditions under which the branch is taken. The order of branches is important. The system takes the first branch with true conditions.

2. Insert a branch for the New Year holiday:
   
a) Right-click the *(Other)* branch.

   b) Select **Insert Branch**.

   The TimeBranch Events window, shown in the following figure, is displayed. It has tabs for branching based on time, day, date, and variable.

![Time Branch Events Window](image)
c) Select the **Date** tab, shown in the following figure.

The start and end dates require two digits each for month and day. The system automatically enters the slash.

d) Enter **01/01** in both the **Start Date** and **End Date** fields and click **OK**.

3. Use the **Day** tab, shown in the following figure to insert a branch for weekends. Remember to right-click the **(Other)** branch and select **Insert Branch**.

Use the drop-down menus to select the start and end days for the weekend.

4. Use the **Time** tab to insert a branch for the hours the business is closed (5 p.m. to 8 a.m.). Remember to use a twenty-four hour time format.
This completes the required branches, as shown in the following figure.

You can insert additional holidays if you want. In the rest of this tour, you will define what happens for each branch.

5. Create a greeting for the New Year holiday:
   a) Drag an Announcement node onto the 01/01 branch and name it **NewYearGreeting**.
   b) Double-click the Initial Prompt field and accept the prompt name.
   c) Click the phrase numbers, drag in the phrases, shown in the following figure, for the New Year greeting (5007, 5008, 5001, and 5002), and click OK.

After the announcement plays, the application should transfer the call to the message center. You must add a node to handle the transfer.

6. Create a node that transfers the call to the message center:
   a) Drag a Transfer Call node onto the open branch after the **NewYearGreeting** and name it **MessageCenter**.
   b) Keep the transfer type **Blind** and enter the telephone number for the message center.
Use your number or another number that you can use to test the application.

c) Double-click the **Message to Caller** field in the Node Inspector, shown in the following figure, to add a message that plays to callers before the transfer. Name the prompt **TransferMessage** and drag in phrase 5003.

![Node Inspector figure]

7. Complete the pass branch for the **MessageCenter** node:
   
a) Drag a Disconnect Call node to the pass branch of the **MessageCenter** node and name it **HangUp**.

b) Drag a Return node to the empty branch of **HangUp** and name it **Quit**.

8. Complete the fail branch of the **MessageCenter** node:
   
   If the transfer fails, the system should play a message and hang up.

   a) Drag an Announcement node to the fail branch of the **MessageCenter** node and name it **BadTransfer**.

b) Create an initial prompt for the node using phrases 5009 and 5006.
c) Use the Jump to Node function on the open branch under the **BadTransfer** node to jump to the **HangUp** node.

This completes the **NewYearGreeting** branch. The other branches are much simpler.

9. Complete the **Sat-Sun** branch:

a) Drag an Announcement node to the branch and name it **WeekendGreeting**.

b) Create an initial prompt using phrases 5008, 5001, and 5002.

c) Use the Jump to Node function on the open branch under the **WeekendGreeting** node to jump to the **MessageCenter** node, as shown in the following figure.

You have not completed this application. Save it now and you can complete it in Presenting menus in VoiceXML applications on page 224.

To continue, see **Copying nodes and VoiceXML applications** on page 219.

**Copying nodes and VoiceXML applications**

The topics in **Copying nodes and VoiceXML applications** provide examples of how to copy:

- Nodes within and between applications
VoiceXML application development tours

- Entire applications

**Note:**
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Copying VoiceXML nodes on page 220.

Copying VoiceXML nodes

When you want to reuse code with no changes, you can use the **Jump to Node** function to jump to the appropriate code section. With the **Copy** function, you copy nodes within an application or between applications. Once nodes are copied, you may modify them as needed. Copying nodes saves you time.

To continue, see Copying nodes within VoiceXML applications on page 220.

Copying nodes within VoiceXML applications

In this tour, you copy a node within an application. If you developed the **vincomng** application in Branching based on time, day, or date in a VoiceXML application on page 212, you complete that application. If you did not complete that tour, use the **svincomng** VoiceXML tour application provided with IVR Designer.

To copy nodes within an application:

1. Open the **vincomng** or **svincomng** VoiceXML tour application.
   Notice that you have an open branch for **17:00-8:00**. The action for this branch is the same as that for the **Sat-Sun** branch.

2. Select the **WeekendGreeting** node.

3. Click the **Copy** icon on the toolbar.
   The system displays a message about how it handles any Jump to Nodes.
   Notice that the node under **WeekendGreeting** is also highlighted.

4. Click the **17:00-08:00** branch.

5. Click the **Paste** icon on the toolbar.

6. Name the new node **AfterHoursGreeting** and click OK.
The node is pasted onto the new branch, as shown in the following figure.

You could change the prompt for the after hours greeting. In this case, the weekend prompt works, so you do not change it. The only thing left to do is to complete the Other branch. This is the branch that is reached only during business hours. It transfers the caller to the operator.

7. Drag a transfer call node to the (Other) branch, naming it TransferToOperator.

8. Select the [TransferMessage] prompt from the drop-down menu for the Message to Caller field.

   You may need to expand the Node Inspector window to locate the Message to Caller field.

   The application reuses the prompt that was used for the transfer to the message center, telling the caller to wait while the call is being transferred.

9. Enter a telephone number for the transfer.

10. Complete the pass branch of the TransferToOperator node by adding a jump to the HangUp node.

    Remember that Jump to Node is a right-click option.
11. Complete the fail branch of the **TransferToOperator** node with a jump to the **BadTransfer** node, as shown in the following figure.

![Diagram](image)

This completes the application.

To continue, see **Copying nodes between VoiceXML applications** on page 222.

**Copying nodes between VoiceXML applications**

IVR Designer makes it easy to reuse information by copying it within and between applications. You can also copy a single node or multiple nodes between applications.

There are a few things to consider when you copy nodes. When you copy multiple nodes, the nodes you copy must be contiguous—in other words, they must be next to each other in the application.

Of course, you can paste a copied node onto an empty node. You can also paste a node onto a node that is not empty. If the node is not empty, it must have at least as many subordinate nodes as the node that is copied. That’s because you need somewhere to place the subordinate nodes.

In **Copying nodes between VoiceXML applications**, you create a new VoiceXML application and copy nodes from an existing application.
To copy nodes between applications:

1. Create a new VoiceXML application and name it `vcpynode`.
2. Drag in an Announcement node and name it `AnnounceCopy`.
3. Save the application.
4. Open any existing application.
5. Click a node in the existing application and click the Copy icon on the Toolbar.

   Note that the node and all its subordinate nodes are highlighted. All highlighted nodes are copied. If you do not want all the nodes, you can delete some in the new application.
6. Use the File menu to return to the `vcpynode` application.

   You do not need to save the existing application because you did not change it.
7. Click the open branch after the `AnnounceCopy` node and click the Paste icon on the Toolbar.

   IVR Designer warns you that you cannot undo the paste.
8. Click Yes.

   The copied nodes are pasted into the `vcpynode` application. You can leave them as they are or make needed changes.

   You can practice copying more nodes between applications. To copy multiple nodes, click the first node to be copied and hold down the Shift key while you click the last node in the range to be copied. All nodes in the range are highlighted.

   You can also copy a node by:
   - Right-clicking the node and selecting Copy node from the menu
   - Highlighting the node and selecting Edit > Copy

To continue, see Copying VoiceXML applications on page 223.

### Copying VoiceXML applications

When you want to create an application that is similar to an existing application, you can copy the entire application and make the appropriate changes.

When you are developing a complex application, you can also save and test the application in stages. If you copy a working portion of the application, you can go back to it if the changes you make do not work.

To copy an entire application:
VoiceXML application development tours

1. Open any existing application.
2. Select Save As from the File menu.
3. Enter a new name for the application.

Only the application name changes. Because you copied the entire application, any associated resources are also copied.

To continue, see Presenting menus in VoiceXML applications on page 224.

Presenting menus in VoiceXML applications

Use Menu nodes to create code that presents a list of choices to callers. The topics in Presenting menus in VoiceXML applications show you how to create and use Menu nodes.

In this tour, you complete an application that gives callers access to loan information or branch locations. The vmenu application is based on the vincomng application that was created in the Branching based on time, day, or date in a VoiceXML application on page 212 tour. The menu application:

- Branches based on time and day.
- Takes action based on whether the call is received during business hours:
  - During business hours, presents a menu of choices, takes caller input, transfers calls to a selected extension, hangs up, and quits.
  - During non-business hours, plays a message telling callers that the office is closed, transfers calls to the message center, hangs up, and quits.

Note:
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Using menus in VXML applications on page 224.

Using menus in VXML applications

In this topic, you create a menu that callers use to choose the type of information they want to hear. You start with an existing application and add a menu. If you completed the vincomng application in Copying nodes between VoiceXML applications on page 222, you can use that application. If not, use the svmenu VoiceXML tour application provided with IVR Designer.
To create a menu:

1. Open the **incoming** application or the **svmenu** VoiceXML tour application, as shown in the following figure, and save it as **vmenu**.

   For information on saving an application under a different name, see **Copying VoiceXML applications** on page 223.

   ![Diagram of menu application](image)

   This application transfers callers to the operator during business hours. The new **menu** application gives callers the choice of hearing loan information or branch information. The first step is to remove the **TransferToOperator** node.

2. Right-click the **TransferToOperator** node and select **Delete Node**, confirming that you want to delete the node.

   The **(Other)** branch is now empty.
3. Drag a Menu node onto the empty branch and name it **MainMenu**, as shown in the following figure.

By default, Menu nodes include three numbered branches and a fail branch.

4. Create the initial prompt for the menu as follows, dragging in existing phrases and adding new phrases as needed.

   **Welcome to Big Bank.**
   **For loan information, press one.**
   **For branch locations, press two.**

5. Drag a Transfer Call node to the first branch under **MainMenu** and name it **LoanInformation**.

6. Add a telephone number for the transfer and select `[TransferMessage]` as the **Message to Caller**.

   Use a telephone number you can test.

7. Complete the pass and fail branches of the **LoanInformation** node, as shown in the following figure.
If the transfer completes correctly, the call should disconnect. That means that the pass branch should jump to the **HangUp** node. If the transfer fails, the call should be handled in the same way as the failed transfer to the message center.

8. Copy the **LoanInformation** node and paste it on branch [2]. Rename it **BranchInformation** and change the telephone number, as shown in the following figure.

For more information on copying nodes, see *Copying nodes within VoiceXML applications* on page 220.

This completes the branches to the loan information and branch information lines. The third branch is not needed.

9. Right-click the third branch and select **Delete Branch**.

The fail branch of the menu node is the only remaining empty node. The application just disconnects if the caller fails to respond to the menu successfully.
10. Jump to the **HangUp** node from this branch, as shown in the following figure.

![Diagram](image)

This completes the application.

11. Verify the application.

The verification results window, shown in the following figure, indicates items that have not been specified. The application can run without these items. Just close the window.

![Verification Results](image)

For more information on verifying applications, see *Verifying a VoiceXML application* on page 199.

12. Simulate the application, starting from the **TimeBranch** node.

For more information on simulating applications, see *Simulating a VoiceXML application* on page 200.
The simulation starts and takes the branch for the current day and time. It counts down and then continues, as shown in the following figure.

The simulation prompts you for how the call should be handled, as shown in the following figure.

13. Select either available option.

The simulation completes.

14. Simulate the application again, starting from the MainMenu node.
VoiceXML application development tours

In this case, you have to select an option. You can type it or click the number on the keypad.

15. If you have a development system, generate code and transfer, assign, and test the application. Make sure that you test all branches.

To continue, see VXML menu error prompts and response handling on page 230.

VXML menu error prompts and response handling

When you develop voice applications, you need to anticipate the needs and limitations of your callers. For example, callers may enter incorrect touchtones or fail to respond to prompts. With IVR Designer, you can develop applications that anticipate a variety of responses from callers.

On the Menu node, use the following editable attributes to specify prompts for different errors that callers may make.

- **A Bad Input Prompt** plays when the caller enters a choice that was not offered. For example, a caller may enter a touchtone for which there is no branch.

- **A Timeout Prompt** plays if the caller does not start to provide input within a specified time.

Also use the following editable attributes to specify how the Avaya IR or CONVERSANT system should deal with caller responses.

- **Input Parameters** identify how callers interact with the menu.
  - With **Allow Interrupt**, you can identify whether caller input can interrupt announcements.
  - With **Input ASR Mode**, you can select Touchtone or one of the VoiceXML grammars for caller input.

- **Timeouts** identify how long the system should wait for certain caller actions. For example, timeouts specify how long the system should wait for the first digit entered by a caller. Timeouts also specify how long to wait between digits entered by callers.

- **Tries** identifies whether to reprompt callers after they make errors.

To continue, see Specifying VXML menu error prompts on page 230.

Specifying VXML menu error prompts

Now you specify the error prompts and response handling for the menu you created in the vmenu application.

To specify error prompts:
1. Open the Node Inspector for the **MainMenu** node, as shown in the following figure.

2. Create a Bad Input prompt for this menu:
   a) Double-click the **Bad Input Prompt** field.
   b) Accept the prompt name.
   c) Enter the following prompt and accept the new phrase.
      
      I did not understand you.

3. Create a Timeout prompt for this menu:
   a) Double-click the **Timeout Prompt** field.
   b) Accept the prompt name.
   c) Enter the following prompt and accept the new phrase.
You must enter a response.

You do not need a Too Few Digits Prompt because the caller is entering a single digit.

4. Click the Response tab.

To continue, see Specifying VXML menu response handling on page 232.

Specifying VXML menu response handling

You have many options when setting response handling parameters, as shown in the following figure.

- You allow the caller to interrupt the prompt, so leave Allow Interrupt set to True.
- The application does not use automatic speech recognition, so leave the Input ASR Mode field at Touchtone.
- The default time the system waits before time out, First Digit Timeout, is five seconds. That works for this application. The three-second timeout between digits, the Inter-Digit Timeout, is also OK.
- After each try, the application replays the initial menu prompt, so leave Replay Initial Prompt set to True.
You did not need to change anything on the Response tab for this application. You know what each item means now, so you can decide if you need to change them for your applications.

To complete the application:
1. Save the application.
2. Verify and simulate the application.
3. If you have a development system, generate code and transfer and assign the application.

To continue, see Collecting information in VoiceXML applications on page 233.

Collecting information in VoiceXML applications

The Prompt and Collect node is similar to the Menu node. You use it to create code that requests longer input strings, such as account numbers, dollar amounts, or other entries. The topics in this section provide examples of how to create and use Prompt and Collect nodes.

In this tour, you complete an application that gives callers access to their account balance or to home or auto loan rates.

The vtelbank application:
- Welcomes the caller to the telephone banking line.
- Prompts the caller to select from menu options and takes action based on caller input:
  - If the caller selects the home rate option, the application announces the rate to the caller, and returns to the menu.
  - If the caller selects the auto rate option, the application announces the rate to the caller, and returns to the menu.
  - If the caller selects the account balance option, the application prompts the caller to enter an account number and transfers the caller to an account representative.
  - If the caller does not make a valid menu choice, the application plays a Goodbye message, and then hangs up and quits.

As you create this application, you will use existing phrases or create your own phrases by typing them into the Prompt Editor.

Note:
This tour makes use of a VoiceXML tour application from the IVR Designer CD-
VoiceXML application development tours

ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Using Prompt and Collect in VoiceXML applications on page 234.

Using Prompt and Collect in VoiceXML applications

In this topic, you open the partially developed svtelbnk VoiceXML application and add a Prompt and Collect node.

1. Open the svtelbnk application, as shown in the following figure.

![VoiceXML application diagram]

The application includes nodes for providing information on loan rates and for ending the call. You will add a Prompt and Collect node to the third branch. This node will collect the account number to provide the account balance.
2. Drag a Prompt and Collect node to the third branch of the menu and name it AskForAcctNo, as shown in the following figure.

3. Create an initial prompt that says the following:

   Please enter your account number followed by the pound sign.

4. Save the application.

To continue and complete the parameters of the Prompt and Collect node, see Prompts, Response, and Confirmation in VoiceXML on page 235.

Prompts, Response, and Confirmation in VoiceXML

On the Prompt and Collect node, you can specify error prompt and response handling parameters, similar to those for Menu nodes. You can also specify confirmation parameters.

On the Prompts tab, you specify the following prompts for different errors that callers may make.

- **A Bad Input Prompt** plays when the caller enters a choice that was not offered. For example, a caller may enter a touchtone for which there is no branch.
- **A Timeout Prompt** plays if the caller does not start input within a specified time.
- **A Too Few Digits Prompt** plays if the caller does not enter the minimum number of digits you specify.
- **A Confirm Input Prompt** repeats what the caller entered and asks the caller to confirm whether the entry is correct.

On the Response tab, you specify the following parameters that identify how the Avaya IR or CONVERSANT system should deal with caller responses.

- **Input Parameters**
VoiceXML application development tours

— Identify whether caller input can interrupt announcements with **Allow Interrupt**.

— Specify the **Maximum Length** and **Minimum Length** for the caller response.

— Specify the character to **Terminate Response With**. The pound sign (#) is most commonly used.

— Specify the **Speech Recognition Confidence Level** to set a threshold for the confidence level of the recognized result returned by the recognition engine.

- **Timeouts** identify how long the system should wait for certain caller actions. For example, timeouts specify how long the system should wait for the first digit entered by a caller. Timeouts also specify how long to wait between digits entered by callers.

- **Tries** identify the number of attempts a caller should be given. Tries also identifies how to reprompt callers after they make errors.

- **Caller’s Response** specifies how the caller will respond (touchtones or voice) and where to store the caller input. This must be a character variable. The system treats the entered digits as a string of characters, not a number. The default system variable for the caller’s response is `[CollectedDigits]`. The Caller's Response attributes also specify the language to use for speech recognition, the number of N-best values to be returned by the recognition engine, and where to store the N-best values.

On the **Confirmation** tab, you specify the following parameters that identify how caller response is to be confirmed.

- **Confirmation ASR Parameters** specify what, if anything, the system should confirm for automatic speech recognition.
  
  — **[YesDigit]** is a variable with a default value of 1. **[NoDigit]** is a variable with a default value of 2. If these are specified, the system considers the digit 1 to confirm and the digit 2 to not confirm.

  — **Confirmation ASR Input Mode** identifies whether a touchtone or a VoiceXML grammar is used to input the confirmation.

  — **Confirmation ASR Language** specifies the language to use for confirmation input from the caller.

  — **Confirmation Mode Chosen** specifies the variable that stores the value indicating how the caller responded to the Confirmation Prompt.

To continue, see **Specifying VoiceXML Prompt and Collect parameters** on page 236.

**Specifying VoiceXML Prompt and Collect parameters**

To specify the error prompts for the Prompt and Collect node in the vtelbnk application.
1. Click the **AskForAcctNo** node, as shown in the following figure, and open the Node Inspector.

![Node Inspector Diagram]

2. Create a **Bad Input Prompt** with the following message:
   
   *Sorry, the account number you entered is not valid.*

3. Use the **Timeout Prompt** from the **Menu** node.
   
   Select **MenuTimeoutPrompt** from the drop-down menu.

4. Create a **Too Few Digits** prompt with the following message:
   
   *You entered too few digits.*

5. Create a **Confirm Input** prompt:
   
   This prompt is used to confirm the caller input.
   
   a) Add the prompt and accept the default name.
   
   b) Enter the following as the first line of the prompt:

   ```
   You entered
   ```

   c) Drag the **[Collected Digits]** variable from the Variable Manager in the Globals Manager into the Prompt Editor.

   You can learn more about variables in **Using variables and speech in VoiceXML applications** on page 247.

   d) Complete the prompt with the following:

   ```
   If this is correct, press one.
   If this is incorrect, press two.
   ```

6. Click the **Response** tab of the Node Inspector.

   The response page for a Prompt and Collect node has more parameters to set than the **Menu node response page**.
7. For the input parameters:

a) Keep the **Allow Interrupt** parameter at **True**.

b) Set both the **Maximum Length** and **Minimum Length** to 6. (Account numbers are six-digit numbers.)

c) Change **Terminate Response With** to #. (Callers press the pound sign when they are finished with a response.)

d) Use the default values for **Timeouts** and **Tries**. The other parameters can also be left at the defaults, as shown in the following figure.
8. Click the **Confirmation** tab of the Node Inspector, shown in the following figure.

![Node Inspector](image)

9. Take a moment to review the default values.

   The default values for this application are acceptable. No action is required on this tab.

   This completes the Prompt and Collect node. You must now transfer the caller to an account representative.

10. Drag a Transfer Call node to the pass branch of the **AskForAcctNo** node and name it **TransferToAcctRep**.

11. Type a telephone number for the transfer and create a **Message to Caller** that says:

    **Please wait while your call is transferred to an account representative.**

12. Complete the pass branch of the **TransferToAcctRep** node with a jump to the **HangUp** node.

13. Drag an Announcement node to the fail branch of the **TransferToAcctRep** node, name it **TransferFail**, and create an initial prompt that says:

    **Your call could not be transferred at this time. Please try again later.**
14. Complete the open branch under the **TransferFail** node and the fail branch of the **AskForAcctNo** node with jumps to the **Menu** node, as shown in the following figure.

15. Save, verify, and simulate the application.

16. If you have a development system, generate code, transfer, assign, and test the application.

   See *Generating and assigning VoiceXML applications* on page 202 for more information.

To continue, see *Using calculations in VoiceXML applications* on page 240.

---

**Using calculations in VoiceXML applications**

Sometimes information that you want to give callers isn’t available directly. You may need to perform calculations to get the information they request.
You can use Set and Test nodes to perform the required calculations. The Set and Test node has many functions. The one you use in *Using calculations in VoiceXML applications* is setting the value of a variable based upon a calculation.

The application you develop:

- Welcomes callers.
- Prompts callers to enter a deposit amount, calculates annual interest on the amount entered, and announces that interest back to the caller.

**Note:**
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see *Performing calculations in VoiceXML* on page 241.

**Performing calculations in VoiceXML**

In this topic, you add a Set and Test node to a partially completed application. The added node computes annual interest.

To develop the **vearn** application:

1. Open the **svearn** VoiceXML tour application, shown in the following figure, and rename it **vearn**.

2. Review the application as it currently exists:

   a) Review the Variables Manager and notice the custom variables **[Deposit]** and **[Interest]**.

   Caller input is always considered to be characters, so caller input cannot be used directly in calculations. The **vearn** application saves the caller input into the **[CollectedDigits]** variable. You assign the value from **[CollectedDigits]** to a custom number variable called **[Deposit]** that is used in a calculation. The **[Interest]** variable stores the result of the interest calculation.
b) Review the Phrases Manager and Prompts Manager to see the prompts and phrases that have been included in the application.

c) Review the Node Inspector for the **Welcome** node.

   Allow Interrupt is set to True so callers can interrupt the prompt.

d) Review the Node Inspector for the **EnterDeposit** node.

   This node asks callers to input a deposit amount. For Responses, the Maximum Length is set to 10, with # to terminate responses. The caller response is saved in [CollectedDigits].

3. Drag a Set and Test node to the pass branch under the **EnterDeposit** node and name it **SetDeposit**.

   You can use the **Assign** expression in a Set and Test node to assign the value from one variable type to another variable type. The Assign expression enables you to specify Source and Destination variables. You can use multiple Assign expressions within the same Set and Test node, if needed. For instance, you can assign the value from one variable type to more than one other variable type. In this application, you assign the value from the [CollectedDigits] variable to the [Deposit] variable.

4. Open the Node Inspector, shown in the following figure.

   ![Node Inspector](image)

   On the node inspector for the **SetDeposit** node, you can specify expressions, branch conditions, or both. In this case, you specify one expression.
5. Click + at the right side of the expression field to add an element.

6. Double-click the open field or click ... to open the Expression Editor.

7. Complete the Expression Editor, as shown in the following figure:
   
a) Select Assign as the Expression Type.

b) Select [Deposit] as the Destination Variable.

c) Select [CollectedDigits] as the Source Variable.

8. Click OK to accept the assignment and close the window.
The Node Inspector, shown in the following figure, now shows the expression you created.

9. Drag another Set and Test node to the pass branch under the SetDeposit node and name it FigureInterest.

   You could also complete both calculations with a single Set and Test node.

10. Click + at the right side of the expression field to add an element.

11. Double-click the open field or click ... to open the Expression Editor.

12. Complete the Expression Editor, as shown in the following figure:

   The current interest rate is 5% or 1/20. You divide the deposit by 20 to determine the interest.

   a) Select Arithmetic as the Expression Type.

   b) Select [Interest] as the Destination Variable.

   c) Select [Deposit] as Operand 1.

   d) Select / as the Operator.
e) Type 20 as Operand 2.

![Image of Set Test Expression Editor]

13. Click **OK** to accept the value and close the window.

The Node Inspector, shown in the following figure, now shows the expression you created.

![Image of Node Inspector]

This completes the Set and Test node.

14. Drag an Announcement node to the pass branch of the **FigureInterest** node and name it **AnnounceInterest**.

This node speaks the result of the calculation back to the caller.

15. Complete the **AnnounceInterest** node:
a) Add an initial prompt that says:

```
The annual interest on a deposit of [Deposit] dollars is [Interest] dollars.
```

b) Set `Allow Interrupt` to `False`.

16. Complete the open branch under the `AnnounceInterest` node with a `Disconnect` node, named `HangUp`, and a `Return` node, named `Quit`, as shown in the following figure.

```
Welcome
⇒ EnterDeposit
⇒ SetDeposit
⇒ FigureInterest
        (default)
⇒ AnnounceInterest
⇒ HangUp
⇒ Quit
```

17. Drag an Announcement node to the fail branch of the `EnterDeposit` node and name it `TryAgain`.

18. Complete the `TryAgain` node:

a) Add an initial prompt that says:

```
There was a problem with your input.
Try again later.
```

b) Set `Allow Interrupt` to `False`. 
19. Jump to the **HangUp** node from the open branch under the **TryAgain** node, as shown in the following figure.

![VoiceXML diagram](image)

20. Save, verify, and simulate the application.

21. If you have a development system, generate code, and transfer, assign, and test the application.

To continue, see Using variables and speech in VoiceXML applications on page 247.

---

**Using variables and speech in VoiceXML applications**

For callers, speech is the most obvious part of voice applications. You can play:

- Prompts using Text-to-Speech or professional recordings
- Information just entered by callers
- Information from calculations

IVR Designer provides a selection of tools for creating, organizing, and using speech within applications. Variables are an example of one of these tools. When you create, organize, and use speech, you must follow certain rules. It is important that you learn these rules to use speech and variables effectively. This tour shows you how to use variables and speech in applications.

In some of the other tours, you may have worked with predefined system variables, such as `[CollectedDigits]`, and custom variables, such as `[Deposit]`. In this tour, you:

- Learn more about variables.
VoiceXML application development tours

- Create custom variables with different formats to see how they are spoken.
- Learn more about editing prompts and phrases.

You use two applications:

- **vspkvar** provides callers with the opportunity to enter a number and choose the format to hear it spoken back.
- **vtelbnk** provides callers with telephone banking menu choices. Modifying this application gives you practice in editing prompts and phrases.

**Note:**
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Variable formats in VoiceXML on page 248.

Variable formats in VoiceXML

IVR Designer permits six basic types of variable, each of which has inherent attributes and default values. To program successfully in IVR Designer, you should know these attributes and values, and the rules for using the different types of variables. For example, you may already know that **[CollectedDigits]** is a character-type variable, and that no arithmetic operations can be performed on character-type variables.

**VoiceXML variable types and formats**

Each type of variable has a default value, as well as default formats for how it is spoken. You may edit the Spoken As attribute of variables to have their values spoken as your application requires. The input that VoiceXML variables accepts may depend on the Text-to-Speech software that is used.

Here is a summary of the variable types that includes their attributes and their default values:

- **Character** accepts as input all standard ASCII characters, including special characters and punctuation. The extended ASCII character set is not supported.
  
  Spoken As: blank
  Default: blank

- **Currency** accepts numerals (0 through 9), a dollar sign, a leading minus sign, and a decimal points as input.
  
  Spoken As: currency
  Default: 0

- **Date** accepts numerals (0 through 9) as input.
Basic VoiceXML application development tours

- **Number** accepts numerals (0 through 9) and a leading minus sign as input.
  
  Spoken As: number  
  Default: 0

- **Real** accepts numerals (0 through 9), a leading minus sign, and a decimal point as input.

  Spoken As: number  
  Default: 0.0000

- **Time** accepts numerals (0 through 9) as input.

  Spoken As: time:hh24mmss  
  Default: 000000

The actual **Spoken As** formats can be changed.

**Using variables in VoiceXML**

You can change the attributes of variables to fit the needs of your applications. The most obvious attribute to change is **Spoken As**. When you specify that a number be spoken with a digit or an ordinal value, for instance, the change is immediately obvious when you call in to test the application.

You may also want to specify a default value for a variable, for example, if you have places in your application where an insurance rate can be calculated. Rather than enter the rate as a literal value in the **Operand** field of each Set and Test node that uses it, you create a variable called **[Rate]** and set its default value to the current rate. When the rate changes, you have to make the change in only one place, the Variables Manager.

You need to consider operation and variable types when setting up arithmetic operations and when converting variable values.

**Spoken As formats**

The value that appears in the **Spoken As** column of the Variables Manager depends on the type of variable. When you make certain changes to variable values, IVR Designer automatically makes corresponding changes, summarized in the table below.

<table>
<thead>
<tr>
<th>Change</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a new type for the variable.</td>
<td>The variable is assigned a default value in the <strong>Spoken As</strong> column.</td>
</tr>
</tbody>
</table>
You can set the **Spoken As** attribute of a variable in the Variables Manager. Click the **Spoken As** cell and select the format from the drop-down menu.

The **Spoken As** options are:

- **acronym** speaks the characters as individual characters.
  Example: The character string *IVR* is spoken as "I V R."
- **address** speaks the number as an address.
  Example: The number *9167* is spoken as "9 1 6 7."
- **currency** speaks the number as dollars and cents.
  Example: The number *105.32* is spoken as "one hundred five dollars and thirty-two cents."
- **date:ymd** speaks the number as a date, including the year, month, and day.
  Example: The number *20020101* is spoken as "January first, two thousand and two."
- **duration:hms** speaks the number as a duration, including hours, minutes, and seconds.
  Example: The number *153246* is spoken as "fifteen hours, thirty-two minutes, and forty-six seconds."
- **duration:hm** speaks the number as a duration, including hours and minutes.
  Example: The number *1532* is spoken as "fifteen hours and thirty-two minutes."
- **duration:ms** speaks the number as a duration, including minutes and seconds.
  Example: The number *1532* is spoken as "fifteen minutes and thirty-two seconds."
- **duration:h** speaks the number as a duration, including hours.
  Example: The number *15* is spoken as "fifteen hours."
- **duration:m** speaks the number as a duration, including minutes.
  Example: The number *32* is spoken as "thirty-two minutes."
- **duration:s** speaks the number as a duration, including seconds.
  Example: The number *12* is spoken as "twelve seconds."
- **measure** speaks the number as a measurement.
- **name** speaks the character string as the proper name of a person or other entity.
• **net:email** speaks the character string as an Internet email handle.
  Example: The character string *me@avaya.com* is spoken as "me at avaya dot com."

• **net:uri** speaks the character string as an Internet Uniform Resource Identifier (URI).
  Example: The character string *www.avaya.com* is spoken as "w w w dot avaya dot com."

• **number** speaks the number as a whole number.
  Example: The number 123 is spoken as "one hundred twenty-three."

• **number:digits** speaks the number as individual digits.
  Example: The number 123 is spoken as "one two three."

• **number:ordinal** speaks the number as an ordinal number.
  Example: The number 1 is spoken as "first."

• **telephone** speaks the number as a telephone number.

• **time:hms** speaks the number as a time, including hours, minutes, and seconds.
  Example: The number 010203 is spoken as "one oh two and three seconds A M."

To continue, see *Creating and speaking variables in VoiceXML* on page 251.

### Creating and speaking variables in VoiceXML

In this topic, you create an application, **vspkvar**, that uses different variable types and Spoken As formats. The application asks callers to enter a number. Then callers have a choice of how to have the number spoken back. This is not a realistic application, but it helps you understand the ways that variables are spoken.

The **vspkvar** application:

• Welcomes the caller.

• Prompts the caller to enter from one to seven digits.

• Prompts the caller to select from menu options to hear the entry spoken back in currency, number, or real format, or to end the call.

• For each option, converts the entry to the selected format, speaks it back to the caller, and returns to the menu.

• Takes action as needed to respond to caller errors:
  — If the caller does not make a valid menu choice, plays a Goodbye message, then hangs up and quits.
If the caller does not make a valid entry, plays a Goodbye message, then hangs up and quits.

To create the vspkvar application:

1. Open the svspkvar VoiceXML tour application, as shown in the following figure, and rename it vspkvar.

   ![VoiceXML tour application diagram]

   This application already includes many of the required nodes. You add and specify nodes for the menu branches.

2. Review the current application:
   
   a) Review the prompts in the Prompt Manager and the phrases in the Phrases Manager.
   
   b) Review the parameters for the GetInput node.

      Notice the **Save Response In** parameter on the **Response** tab. The application has a menu following this Prompt and Collect node. The caller’s response to the menu overwrites the value in the **[CollectedDigits]** variable before it can be assigned to another type of variable. To keep the value representing the callers response, the application saves the caller’s response in another character variable, called **[CharacterEntry]**.

3. Create the additional custom variables:
   
   a) Right-click the Variables Manager and select **New**.
   
   b) Name the first variable **MoneyEntry**.
   
   c) Select **currency** from the drop-down menu for variable **Type**.
d) Accept the default for **Spoken As**.

e) Create the remaining custom variables, with the indicated **Types** and **Spoken As** formats:

- **[NumberEntry]** is number, number.  **[RealEntry]** is real, number.

Notice the default values for these variables.

4. Insert a new menu branch for the option to end the call and complete the branch with a jump to the **Goodbye** node, as shown in the following figure.

   a) Right-click the fail branch of the **Menu** node.

   b) Select **Insert Branch**.

   c) Select 4 as the touchtone for the branch.

   d) Jump to the **Goodbye** node from the branch.

5. Drag a Set and Test node to the first **Menu** branch and name it **AssignCurrencyValue**, as shown in the following figure.

   This node assigns the value from the **[CharacterEntry]** variable to the currency variable, **[MoneyEntry]**.

6. Define the **Assign** expression, as shown in the following figure:

   a) Click + in the Node Inspector.
VoiceXML application development tours

b) Click the first element and click ....

c) Select Assign as the Expression Type.

d) Select [MoneyEntry] as the Destination Variable.

e) Select [CharacterEntry] as the Source Variable.

f) Click OK.

The Set and Test node now assigns the value in the [CharacterEntry] variable to a currency variable. Now the application needs to announce the value of the variable to the caller.

7. Drag an Announcement node to the open branch under the AssignCurrencyValue node and name it AnnounceCurrencyValue.

8. Drag in the phrases and variables to create the following prompt:

   In a currency format, that is [MoneyEntry].

9. Drag a Set and Test node to the second Menu branch and name it AssignNumberValue.

10. In the Node Inspector, add an element that assigns [CharacterEntry] to [NumberEntry].

11. Add an Announcement node to the open branch under AssignNumberValue, name it AnnouceNumberValue and create an announcement that says:

    In a number format, that is [NumberEntry].
12. Drag a Set and Test node to the third Menu branch and name it AssignRealValue.

13. In the Node Inspector, add an element that assigns [CharacterEntry] to [RealEntry].

14. Add an Announcement node to the open branch under AssignRealValue, name it AnnoucerefRealValue, and create an announcement that says:

   In a real format, that is [RealEntry] .

15. Complete the open branches under each of the Set and Test nodes with jumps to the Menu node, as shown in the following figure.

![Diagram](image)

This completes the vspkvar application.

16. Verify and simulate the application.

17. If you have a development system, generate code and transfer and assign the application.
18. Test the application with different menu selections.

There are many more variable **Types** and **Spoken As** formats that you can use in applications. This application should have given you an idea of some of the most commonly used options.

To continue, see Creating phrases in VoiceXML on page 256.

**Creating phrases in VoiceXML**

You can create phrases in three ways:

- Import a phrase table.
- Enter new phrases as you create prompts.
- Create or edit phrases directly in the **Phrases Manager** as you did if you completed Creating a VoiceXML phrase table on page 212 of Branching based on time, day, or date in a VoiceXML application on page 212. This can be helpful if your design specification includes a list or phrases or if you need to make changes to phrases.

In this topic, you work directly in the Phrases Manager to add and edit phrases. You add a new phrase for an additional menu option to transfer funds between accounts and change the option number in the existing phrase for the option to end the call.

To add and edit phrases:
1. Open the **svtlbnk1** VoiceXML tour application, as shown in the following figure, and save it as **vtelbnk1**.
2. Expand the Globals Manager and reposition the Phrases Manager and Prompts Manager as necessary to view all the existing phrases, as shown in the following figure.

<table>
<thead>
<tr>
<th>Phrases</th>
<th>Recorded</th>
<th>Outdated</th>
<th>Phrase Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td></td>
<td></td>
<td>Welcome to the Big Bank telephone banking line.</td>
</tr>
<tr>
<td>5001</td>
<td></td>
<td></td>
<td>For today's best home loan rate, press one.</td>
</tr>
<tr>
<td>5002</td>
<td></td>
<td></td>
<td>For today's best auto loan rate, press two.</td>
</tr>
<tr>
<td>5003</td>
<td></td>
<td></td>
<td>For your account balance, press three.</td>
</tr>
<tr>
<td>5004</td>
<td></td>
<td></td>
<td>To end the call, press four.</td>
</tr>
<tr>
<td>5005</td>
<td></td>
<td></td>
<td>Please enter your six digit account number followed by the pound sign.</td>
</tr>
<tr>
<td>5006</td>
<td></td>
<td></td>
<td>The lowest current three year auto loan rate is six percent.</td>
</tr>
<tr>
<td>5007</td>
<td></td>
<td></td>
<td>The lowest current thirty year home loan rate is seven percent.</td>
</tr>
<tr>
<td>5008</td>
<td></td>
<td></td>
<td>You must enter a response.</td>
</tr>
<tr>
<td>5009</td>
<td></td>
<td></td>
<td>Goodbye.</td>
</tr>
<tr>
<td>5010</td>
<td></td>
<td></td>
<td>Sorry, the account number you entered is not valid.</td>
</tr>
<tr>
<td>5011</td>
<td></td>
<td></td>
<td>You entered too few digits.</td>
</tr>
<tr>
<td>5012</td>
<td></td>
<td></td>
<td>I did not understand you.</td>
</tr>
<tr>
<td>5013</td>
<td></td>
<td></td>
<td>You entered</td>
</tr>
<tr>
<td>5014</td>
<td></td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>5015</td>
<td></td>
<td></td>
<td>If this is correct, press one.</td>
</tr>
<tr>
<td>5016</td>
<td></td>
<td></td>
<td>If this is incorrect, press any other key.</td>
</tr>
<tr>
<td>5017</td>
<td></td>
<td></td>
<td>Please wait while your call is transferred to the account balance line.</td>
</tr>
<tr>
<td>5018</td>
<td></td>
<td></td>
<td>Your call could not be transferred at this time.</td>
</tr>
<tr>
<td>5019</td>
<td></td>
<td></td>
<td>Please try again later.</td>
</tr>
</tbody>
</table>

3. Right-click anywhere in the Phrases Manager and select New.

4. Type the new phrase below and click OK.

   For bank hours and locations, press four.

   The new phrase is added to the Phrases Manager. Now there are two phrases that tell callers to press four. The next step is to change the option number for ending the call to five.

5. Select the phrase that says:

   To end the call, press four.

   You can select a phrase by clicking the phrase and clicking ..., by double-clicking the phrase, or by right-clicking the phrase and selecting Edit Phrase...

6. Change the phrase to read:

   To end the call, press five.

7. Click OK.

   The phrase is now changed in the Phrases Manager.
To continue, see Editing prompts in VoiceXML on page 259.

Editing prompts in VoiceXML

In other tours, you created prompts as you built applications. You did this by typing or dragging in phrases or by dragging in variables. Editing prompts directly in the Prompts Manager saves you from having to go back to a node to locate and edit its prompt. In the Prompts Manager, you can:

- Edit the text of prompts.
- Insert phrases, variables, controls, and breaks.

In this topic, you continue developing the vtelbank1 application that you created in Editing phrases on page 256 in VoiceXML. You edit the Menu prompt to include the new option and edit another prompt.

To continue developing the application:

1. Reposition the Prompts Manager so you can view the prompts, as shown in the following figure.

<table>
<thead>
<tr>
<th>Prompts</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AskForAcctNoBAd]</td>
<td>Sorry, the account number you entered is not valid.</td>
</tr>
<tr>
<td>[AskForAcctNoCnf]</td>
<td>You entered [CollectedDigits]. If this is correct, press one. If this is inco</td>
</tr>
<tr>
<td>[AskForAcctNoInit]</td>
<td>Please enter your six digit account number followed by the pound sign.</td>
</tr>
<tr>
<td>[AskForAcctNoTooF]</td>
<td>You entered too few digits.</td>
</tr>
<tr>
<td>[AutoLoanRateInit]</td>
<td>The lowest current three year auto loan rate is six percent.</td>
</tr>
<tr>
<td>[HomeLoanRateInit]</td>
<td>The lowest current thirty year home loan rate is seven percent.</td>
</tr>
<tr>
<td>[MenuBInputDialog]</td>
<td>I did not understand you.</td>
</tr>
<tr>
<td>[MenuInitialPrompt]</td>
<td>For today's best home loan rate, press one. For today's best auto loa</td>
</tr>
<tr>
<td>[MenuTimeoutPrompt]</td>
<td>You must enter a response.</td>
</tr>
<tr>
<td>[TransferFailInitProm]</td>
<td>Your call could not be transferred at this time. Please try again la</td>
</tr>
<tr>
<td>[TransferToAcctLine]</td>
<td>Please wait while your call is transferred to the account balance lin</td>
</tr>
<tr>
<td>[WelcomeInitialProm]</td>
<td>Welcome to the Big Bank telephone banking line.</td>
</tr>
</tbody>
</table>

2. Select the [MenuInitialPrompt].

You can select the prompt by clicking the prompt and clicking ..., by double-clicking the prompt, or by right-clicking the prompt and selecting Edit Prompt...

The Prompt Editor for the selected prompt is displayed. Notice that the phrase to end the call has been replaced by the edited phrase.

3. Right-click at the beginning of the phrase for ending the call.
Notice that you can insert variables, phrases, phrase breaks, and controls, as shown in the following figure.

4. Select **Insert Phrase**...

The available phrases are listed, as shown in the following figure.

5. Select the phrase for branch hours and locations and click **OK**.

The phrase is added to the **Prompt Editor** and the prompt is complete.

6. Click **OK** to save the edited prompt.
7. Open the Prompt Editor for **AskForAcctNoBadInputPrompt**, shown in the following figure.

![Prompt Editor](image)

You edit this prompt to insert a variable and a pause.

8. Insert the variable **[CollectedDigits]** after the word *entered* in this prompt.
   a) Right-click between *entered* and *is*.
   b) Select **Insert Variable**.
   c) Select the **[CollectedDigits]** variable.
   d) Click **OK**.

9. Insert a pause before the variable.
   a) Right-click before the variable.
   b) Select **Insert Control**.
c) Select break_small and click OK.

The variable and break are now included in the prompt, as shown in the following figure.

10. Accept the edited prompt and the new phrases.

You will not complete the changes to this application here. If you were to do so, you would:

- Rename the [4] branch to [5].
- Add a [4] branch with the nodes to play a message with bank hours and locations.

To continue, see Using embedded call flows in VoiceXML applications on page 263.
Using embedded call flows in VoiceXML applications

You use the Jump to Node function to reuse code in an IVR Designer application. If you want to reuse a piece of code and then continue with the program, the Jump to Node function does not meet your needs. When you jump to a node, there is no way to return to the program flow.

IVR Designer has another function, called an embedded call flow, that you can use to reuse code and return to the program flow after the reused code completes. Besides using embedded call flows to reuse code, you can use them to break long applications into sections. Applications broken up into sections are easier to read.

You can create embedded call flows in two ways:

- Drag in a call flow node and then edit the call flow to develop the code.
- Take existing code within an application and encapsulate it, turning it into an embedded call flow.

**Note:**
This tour makes use of a VoiceXML tour application from the IVR Designer CD-ROM. If you did not install the sample applications with IVR Designer, install them before you continue with this tour.

To continue, see Creating an embedded call flow in a VoiceXML application on page 263.

Creating an embedded call flow in a VoiceXML application

To create an application, vmenuflw, that uses a Call Flow node:

1. Create a new VoiceXML application and name it vmenuflw.

2. Drag in a Call Flow node and name it Menu, as shown in the following figure.

   ![Menu node](image)

   The number 1 in parentheses (1) after the node name indicates that this is the first instance of the node in the application. If you use the [Menu] call flow again later in the application, it will include a two in parentheses (2).

   Any open branch in an embedded call flow automatically returns to the (Default) branch in the calling flow, in this case the [Main] call flow. The program flow continues from that
point. Any Return node in an embedded call flow adds a branch to the call flow node in the [Main] call flow.

The [Main] tab is the only one that appears at the bottom of the design pad at this point, even though you added a Call Flow node to the [Main] call flow. The tab for the [Menu] call flow will be displayed when you edit it.


There is now a blank design pad for the [Menu] call flow. Notice the new [Menu] tab, as shown in the following figure.

4. Build the [Menu] call flow:
   a) Drag in a Menu node and name it MainMenu.
   b) Create an initial prompt that says:

      Press one for account information.
      Press two for loan information.
      Press three for branch locations and hours.

   c) Drag a Transfer Call node to the [1] branch. Name it Account, and enter a telephone number for the transfer.
   d) Drag a Disconnect Call node to the pass branch of the Account node and name it HangUp.

      After the HangUp node, you would normally add a Return node. Because this is an embedded call flow, a Return node is not needed. Just leave the branch open, and it will act as a Return node to the [Main] call flow.
   e) Drag an Announcement node to the fail branch of the Account node and name it BadTransfer.
   f) Create an initial prompt that says:

      Your call cannot be transferred at this time.
      Please try again later.
   g) Complete the [Menu] call flow with similar branches to loan information and to branch information.

      Remember that you can use the Jump to Node function where it is appropriate.
The completed call flow should look like the following figure:

Notice that all branches end by jumping to the **HangUp** node. The **HangUp** node returns to the **[Main]** call flow through the open branch. If some of the branches did not return to the **[Main]** call flow through the **HangUp** node, you could add Return nodes here that would go back to different return branches in the **[Main]** call flow. These return branches could then accommodate different actions.

5. Click the **[Main]** tab to return to the **[Main]** call flow.

6. Drag a Return node to the open (Default) branch and name it **Quit**.
   
   This completes the **vmenuflw** application.

7. Save the application.

To continue, see **Encapsulating a VoiceXML call flow** on page 265.

**Encapsulating a VoiceXML call flow**

The **vmenuflw** application is a simple application. You might decide that it does not need a separate call flow for the menu.

To expand the Call Flow node and bring the nodes within it into the **[Main]** call flow:
1. Right-click the [Menu] node in the [Main] call flow and select **Expand Node**.

Notice how the Call Flow node is replaced by the **MainMenu** node from the call flow. The **Quit** Return node appears on what was the open branch, as shown in the following figure.

Even though you expanded the call flow node, the [Menu] tab is displayed at the bottom of the design pad. The call flow remains until you delete it.
2. Click the Call Flow Inventory icon to see the call flows in this application, as shown in the following figure.

![Call Flows](image)

3. Right-click the [Menu] call flow and delete it.

The [Menu] call flow has been deleted and the tab is gone.

If you want the menu to be a separate call flow again, you can encapsulate the node into a call flow.

4. Right-click the MainMenu node and select Encapsulate. Name the new call flow Menu, as shown in the following figure.

![MainMenu](image)

The MainMenu node and its branches are now a separate call flow again. The tab does not appear again until you edit the [Menu] node.

When you expanded the original menu call flow, the Quit Return node was placed onto the open branch. When you encapsulated the new menu node, the branch was no longer open. The system created a return branch (Return1) in the [Main] call flow corresponding to the Return node in the menu call flow. You would add a jump to the Quit node to the (Default) branch.

5. Edit the [Menu] call flow, as shown in the following figure.
VoiceXML application development tours

Remember that **Edit Flow** is a right-click option.

Notice the Return node on the branch under **HangUp**. This is what caused the (Return1) branch to be added to the **[Main]** call flow.

To continue, see Reusing and Copying VoiceXML call flows on page 268.

**Reusing and Copying VoiceXML call flows**

You can copy nodes or use a Jump to Node function to repeat actions. Other options are to reuse or copy call flows.

To complete an application by reusing and copying call flows:
1. Open the `svtimflw` VoiceXML tour application, as shown in the following figure, and rename it `timeflw`.

2. Review the existing application.

   Notice that this application checks for day of the week. Monday through Friday, it checks for business hours. During work hours, it presents a menu. Outside of work hours, the application transfers the call and ends.

3. Click the `[TransferAndEnd]` tab to view the call flow, shown in the following figure.

   This call flow transfers the call to the message center and hangs up.

4. Return to the `[Main]` call flow.

   Now you will specify the `(Other)` branch under the `CheckDay` node. This branch should also transfer the call and end. You will reuse the `[TransferAndEnd]` call flow.

5. Right-click the `(Other)` branch and select *Reuse Call Flow*.

6. Select the `[TransferAndEnd]` call flow.
The new [TransferAndEnd] call flow node uses the same call flow. Because it is the second instance, it has a 2 at the end, as shown in the following figure.

7. Complete the open (Default) branch with a jump to the Quit node.
   
   Next you will copy the [TransferAndEnd] call flow to the branches under MainMenu.

8. Right-click the [1] branch and select Copy Call Flow.

9. Select the [TransferAndEnd] call flow and rename it TransferToAccount.

10. Edit the [TransferToAccount] call flow.

11. Rename the TransferAndEnd node to TransferToAccount and change the telephone number, as shown in the following figure.
This call flow now transfers callers to the account information line.

12. Return to the [Main] call flow.

13. Jump to the Quit node from the (Default) branch.

You could complete the [2] and [3] branches by copying the TransferAndEnd call flow to them and completing the default branches with a jump to the Quit node.

The fail branch hangs up the call and quits. The HangUp node exists in other call flows, but not in the [Main] call flow. You could create the required nodes here, or you could copy the nodes from another call flow.

To continue, see Advanced VoiceXML application development tours on page 272.
Advanced VoiceXML application development tours

Advanced VoiceXML application development tours provide you with instructions for creating VoiceXML applications that give you practice with some of the more advanced features of Avaya IVR Designer. You should have a good understanding of basic IVR Designer features before you complete these tours.

You can view and print the entire Advanced VoiceXML application development tours section in PDF format.

To continue, see Handling strings in VoiceXML applications on page 272.

This section includes the following topics:

Handling strings in VoiceXML applications............................... 272
Using asynchronous events in VoiceXML applications ............... 288
Running TAS applications from VoiceXML.............................. 292
Passing data between VoiceXML applications......................... 300
Using voice input in VoiceXML applications............................ 304
Using database operations in VoiceXML applications.............. 313
Using VoiceXML 2.1 tags in VoiceXML applications.................. 321
Using stored procedures in VoiceXML applications............... 322
Testing and Troubleshooting VoiceXML applications............... 327

Handling strings in VoiceXML applications

There are many types of character strings commonly used in voice applications, including telephone numbers, IDs, social security numbers, product and inventory item numbers, and account balance amounts. Such strings may be entered by a caller or passed to the application by another application.

The Set and Test node of IVR Designer includes several expressions that help you work with strings. The topics in this section provide examples of how to manipulate strings in applications.
To continue, see Using the VoiceXML Length expression on page 273.

Using the VoiceXML Length expression

Using calculations in VoiceXML applications on page 240 showed you how to use the Expression Editor feature of the Set and Test node to specify variables and literal values to be used when the application runs. One type of expression, the Length expression, counts the number of characters in a string and returns the value of that number to a variable.

This topic introduces the Length expression of Set and Test and shows you how to use it in an application. It also reviews the Set and Test Branch Condition Editor.

The application you develop, named length:

- Uses the Length expression to determine the length of an account number entered by the caller.
- Branches, based on the length of the account number.

This type of application is useful, for example, when you want different treatment for accounts of types that use account numbers of different lengths.

The application length:

- Prompts the caller to enter a 5-digit, 6-digit, or 7-digit account number followed by the pound sign (#).
- Stores the entered account number in a custom character-type variable [CIAcctNum].
- Counts the number of characters in [CIAcctNum] and stores the number in a custom number-type variable [LengthOfCIAcctNum].
- Takes action based on the length of the string:
  - If the string is 5 characters long, branches to a Text-to-Speech announcement, hangs up, and quits.
  - If the string is 6 characters long, blind transfers the caller to a live agent. Whether the transfer is successful or fails, it then hangs up and quits.
  - If the string is 7 characters long, branches to a Text-to-Speech announcement, hangs up, and quits.
  - If the string is any other length, branches to a Text-to-Speech announcement, and repeats the prompt for an account number.
  - If the caller fails to respond appropriately in the default number of tries, announces a Text-to-Speech problem message, and then hangs up, and quits.
VoiceXML application development tours

To create the \texttt{vlength} application:

1. Create a new VoiceXML application named \texttt{vlength}.
2. Create the following custom variables:
   
   — \texttt{[CIAcctNum]}. Character variable with the \textit{Spoken As} field set to \texttt{number:digits}
   
   — \texttt{[LengthOfCIAcctNum]}. Number variable with the \textit{Spoken As} field set to \texttt{number}

3. Create the call flow of the application to look like the sample call flow shown in the following figure. Name the nodes as shown.

4. Edit the \texttt{EnterAcctNum} node so that:
   
   — The caller hears:
     
     \texttt{Please enter your 5, 6, or 7 digit account number followed by the pound sign.}
   
   — The minimum number of collected digits is 1 and the maximum is 7.
   
   — The caller response is terminated with a \#. 
   
   — The caller response is saved in the variable \texttt{[CIAcctNum]}.

274 Avaya IVR Designer Help for VoiceXML Applications
5. Edit the CheckLengthOfAcctNum node as follows:

- **Expressions** tab:
  
  Length: \([\text{LengthOfCIAcctNum}] = [\text{CIAcctNum}]\)

- **Branch Conditions** tab:
  
  If \([\text{LengthOfCIAcctNum}] = 5\) Goto Branch (CIAcctNumHas5Digits)

  If \([\text{LengthOfCIAcctNum}] = 6\) Goto Branch (CIAcctNumHas6Digits)

  If \([\text{LengthOfCIAcctNum}] = 7\) Goto Branch (CIAcctNumHas7Digits)

**Note:**

For this application, you specify a single branch condition for each branch. You can specify multiple conditions by clicking `+` in the lower right corner of the window. You can specify each of the condition combinations as **AND** or **OR**, where **AND** requires that both conditions must be met, and **OR** requires that at least one of the conditions is met. You can also create logical groupings of conditions with the **Group** check box.

6. Edit the Has5Digits node so that the caller hears the Text-to-Speech message:

   **The customer account number has 5 digits.**

7. Edit the 6DigitTransfer node so that:

   - The transfer type is **Blind**.

   - The transfer extension is one that you can test.

   - The caller hears the message:

   **Please wait while you are transferred.**

8. Edit the Has7Digits node so that the caller hears the Text-to-Speech message:

   **The customer account number has 7 digits.**

9. Edit the WrongNumberOfDigits node associated with the (default) branch of the CheckLengthOfAcctNum node, so that:

   - The caller hears the Text-to-Speech message:

   **The account number entered is invalid, please try again.**

   - The application jumps to the EnterAcctNum node so the caller can try again.
10. Edit the **SystemProblems** node associated with the fail branch of the **EnterAcctNum** node, so that the caller hears the Text-to-Speech message:

   The system is experiencing problems. Please call back later. Goodbye.

11. Remember to edit the phrase table to select **Text-to-Speech** for code generation.

12. Verify and simulate the application. Simulate for five conditions:
   - 5-digit response
   - 6-digit response
   - 7-digit response
   - An invalid number of digits
   - No response from the caller

13. Save the application.

14. Generate the code for the application. If you have a development system, transfer, assign, and test the application.

   When you test the **length** application:
   - Enter several account numbers using 5 digits, 6 digits, 7 digits, and other lengths.
   - Enter no account number.

To continue, see Using the VoiceXML ItemCount and ItemOf expressions on page 276.

### Using the VoiceXML ItemCount and ItemOf expressions

In this topic, you develop an application that uses the Set and Test **ItemCount** and **ItemOf** expressions to handle strings.

The application:

- Plays a welcome announcement that explains the purpose of the application.
- Using a Set and Test node, assigns a comma-delimited string with 4 items to a variable.
- Using a Set and Test node, counts the number of items in the list contained in **[LANString]** and returns that value to a variable named **[NumberOfItems]**.
- Announces the number of items counted in **[LANString]**.
- Using a Set and Test node, returns the value of each item to a different variable, designated **[CustID]**, **[CustName]**, **[CustBal]** and **[CustPhone]**, respectively.
• Places the contents of [CustBal] into a new currency variable [Balance_Currency].
• Announces the customer name and balance.
• Hangs up and quits.

To create the vitem application:

1. Create a new VoiceXML application named vitem.
2. Add and name the nodes as shown in the following figure.

3. Create the following variables for use in the Set and Test nodes (See Creating and speaking variables in VoiceXML on page 251 for more information on creating custom variables.):
   - [LANString]. Character variable
   - [NumberOfItems]. Number variable
   - [CustID]. Character variable
   - [CustName]. Character variable
   - [CustBal]. Character variable
   - [CustPhone]. Character variable
   - [Balance_Currency]. Currency variable

4. Specify the nodes, using the default values, except as indicated below:
   a) Edit the CreateLANString node to assign the following string value to the [LANString] variable:

      123456789, Your Name, 77.77, 3035551212
b) Edit the ItemCount node to count the number of items in the [LANString] variable, using the ItemCount expression.

c) Edit the ItemOf node as shown in the following figure:

![Node Inspector Window]

5. Create the following prompts and phrases (use Text-to-Speech for the phrases if you have it on your system, or record the phrases):

   — **ExplainPurpose** initial prompt:

   The Set and Test item count expression counts the quantity of comma separated items. The Set and Test Item of expression places a particular item in the string into a variable.

   — **AnnounceNumberOfItems** initial prompt:

   The quantity of items is [NumberOfItems].

   — **ItemOf** initial prompt:

   The customer [CustName] has a balance of [Balance_Currency].

   Good bye.

   This completes the application.

6. Verify and simulate the application.
7. When you are satisfied with the application, generate the code. If you have a
development system, transfer the application to the Avaya IR or CONVERSANT system,
assign it to a channel, and test it.

To continue, see Using the VoiceXML Concat expression on page 279.

Using the VoiceXML Concat expression

The Set and Test Concat expression combines two strings. In this topic, you create an
application that combines strings to create a complete account number. The strings that are
combined are:

- An account number entered by the customer
- An extension that indicates whether the account is a checking (ck) or savings (sv)
  account.

From the customer perspective, there is one account number for all accounts. For this
reason, customers cannot provide the correct extensions. The extensions are added after the
customer selects whether to hear information about checking or savings. For example, if the
account number is 12345, and the customer selects savings, the concatenated (complete)
account number is 12345sv. If the customer selects checking, the complete account
number is 12345ck. Once the concatenated number is available, the application transfers the
caller to an operator.

The vconcat application does the following:

- Welcomes the caller.
- Prompts the caller to enter an account number followed by the pound sign. The
  information is saved in a user-defined variable [CIAccountNumber].
  
  **Note:**
  Valid account numbers range from five to ten digits.
- Asks the caller to press 1 for checking or 2 for savings.
- Takes action based on caller response:
  - If 1 is pressed, concatenates the caller-input account number with the checking
    extension ck, transfers the caller to an operator, hangs up, and quits.
  - If 2 is pressed, concatenates the caller-input account number with the savings
    extension sv, transfers the caller to an operator, hangs up, and quits.
  - If the Prompt and Collect or the account-type menu query fails, plays a problem
    message, hangs up, and quits.

To develop the vconcat application:
VoiceXML application development tours

1. Create a new VoiceXML application named \texttt{vconct}.

2. Create the custom variables required by the application requirements:
   
   - \texttt{[CIAccountNumber]}. Character variable
   
   - \texttt{[CompleteAccountNumber]}. Character variable
   
   - \texttt{[CheckingExtension]}. Character variable with a default value of \texttt{ck}
   
   - \texttt{[SavingsExtension]}. Character variable with a default value of \texttt{sv}

3. Create the call flow to look like the sample call flow below. Name the nodes as in the sample, shown in the following figure.
4. Edit the Set and Test nodes to concatenate the account number entered by the caller from [CIAccountNumber] with the appropriate extension from [CheckingExtension] or [SavingsExtension].

5. Use the Node Inspector to specify the remaining nodes as required. Use the application requirements at the beginning of this topic to guide you in setting attributes. Use default values unless otherwise indicated in the application requirements.

6. Create the prompts and phrases as specified below. (Use Text-to-Speech for the phrases or record the phrases.) You may want to create your own BadInput and Timeout prompts.

   — **Welcome** initial prompt:
     
     Welcome to the concat bank application.

   — **EnterAcctNum** initial prompt:
     
     Please enter your account number.

   — **ChooseAccountType** initial prompt:
     
     For checking account information, press 1. For savings account information, press 2.

   — **ProblemMsg** initial prompt:
     
     I’m sorry, we are experiencing difficulties, please try again later.

7. Verify and simulate the application, using the following scenarios:

   — When prompted for the account number, enter an account number. When prompted for an account type, enter: 1.
VoiceXML application development tours

— When prompted for the account number, enter an account number. When prompted for an account type, enter: 2.

— When prompted for an account number, do not respond.

— When prompted for an account number, enter an account number. When prompted for an account type, do not respond.

8. Save the application.

9. When you are satisfied with the application, generate the code. If you have a development system, transfer the application, and assign it to a channel.

10. Test and trace the application.

To continue, see VoiceXML Substring and Parse comparison on page 282.

VoiceXML Substring and Parse comparison

With the Set and Test node Substring and Parse expressions, you can use a portion of a larger character string in an application, but there are important differences between these two expressions.

• The Substring expression selects a portion of an existing string and places its value in a destination variable. You do this by designating the Starting at Position and the Maximum Length values. Note that the Substring expression leaves the source string intact.

• The Parse expression breaks apart the source string into smaller substrings. Note that the Parse expression alters the source string.

The Starting at Position value is determined by counting the length of the source string from either direction:

• **Left to right.** Start with 0 and count up: 0, 1, 2, and so on, for each character in the string, starting with the leftmost character.

• **Right to left.** Start with -1 and count in negative numbers: -1, -2, -3, and so on, for each character in the string, starting with the rightmost character.

Example:

<table>
<thead>
<tr>
<th>Counting from left to right</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
</table>
The Maximum Length is the number of characters to select, starting at the specified position. Either way you count to reach the Starting at Position, the system always selects characters from left to right.

A Maximum Length of 0 (zero) in the Substring expression means that all the characters starting at the specified position are selected and copied into a destination variable (up to the maximum size of the destination variable).

The Parse expression, unlike Substring, *breaks apart* the source string. With the Parse expression, you split a character string into two smaller strings. To use Parse, you need to specify one or more Parsing Separators (for example, space, period, or comma) and a Source String. Parse takes the portion of the source string before the specified parsing separator and strips it off the source string, placing its value in your designated Destination Variable. Note that this action modifies the source string. The remaining part of the string becomes the new version of the original source string. If you need to keep the original value, save it to another variable before calling the Parse expression.

To continue, see Using the VoiceXML Substring expression on page 283.

**Using the VoiceXML Substring expression**

In this topic, you develop an application that uses the Set and Test Substring expression. The application collects a 9-digit identification (ID) number as a single string, then speaks it back as three separate variables. The application gives the caller the opportunity to correct the ID number.

The application:

- Makes an announcement explaining the purpose of the application.
- Prompts the caller to enter a 9-digit ID number.
- Reprompts if the caller does not enter a 9-digit number within the default time-out period.
- Makes an announcement, hangs up, and quits if the Prompt and Collect fails.
- Uses the Substring expression to place the first three characters, middle two characters, and final four characters of the identification number string in different destination variables.
VoiceXML application development tours

- Includes a Menu node initial prompt that speaks back the ID number the caller entered in three parts: the first three digits, the second two, and the final four. The Menu node tells the caller to press 1 if the ID number is correct as entered, and otherwise to press 2.
  - If the caller selects 1, it thanks the caller and then hangs up and quits.
  - If the caller selects 2, it offers the caller another opportunity to enter the ID number.
  - If the Menu node fails, it thanks the caller and then hangs up and quits.

To create the vsubstrg application:

1. Create a new VoiceXML application named vsubstrg.
2. Create the custom variables as specified below (To select the values for the variables holding the digits of the ID number, the Substring expression counts from left to right.):
   - [CI_ID]. Character variable
   - [FirstThree]. Character variable for the first three digits of the ID number, with the Spoken As field set to number:digits
   - [MiddleTwo]. Character variable for the second two digits of the ID number, with the Spoken As field set to number:digits
   - [LastFour]. Character variable for the final four digits of the ID number, with the Spoken As field set to number:digits
3. Create the call flow to look like the sample call flow in the following figure. Name the nodes as in the sample. Use the Node Inspector to specify the nodes as required. Use default values unless otherwise indicated.
4. Create the following prompts and phrases as specified in the application requirements. (Use Text-to-Speech for the phrases or record the phrases.)

   - **Welcome** initial prompt:
     
     Welcome to substring.
     This program collects a nine digit identification number as a single string, then speaks it back as three separate variables.

   - **EnterID** initial prompt:
     
     Please enter a nine digit identification number now.

   - **EnterID** timeout prompt:
     
     Please enter a nine digit ID number now.

   - **ConfirmNumber** initial prompt:
     
     You entered
     [FirstThree]
     ,
     [MiddleTwo]
     ,
     [LastFour]
     .
     If this is correct, press 1. Otherwise, press 2.

   - **ThanksForCalling** initial prompt:
     
     Thank you for calling.
     Goodbye.

5. Verify and simulate the application.

6. If you are using Text-to-Speech, verify that it is used for **PhraseTable** for code generation.

7. When you are satisfied with the application, generate the code. If you have a development system, transfer the application, install it, and assign it to a voice channel.

8. Test the application on the Avaya IR or CONVERSANT system by calling and responding with a range of responses.

   a) When prompted, enter: **123456789**
     
     Verify that you hear the correct response.

   b) When asked for confirmation, enter: **3**
VoiceXML application development tours

Verify that the application plays the confirmation prompt again.

c) When asked again for confirmation, enter: 2

Verify that the system prompts you again for a nine-digit ID number.

d) When prompted, enter: 987654321

e) When asked for confirmation, enter: 1

Verify that the system says "Goodbye" and quits.

f) When prompted, enter: 12345

You should be prompted three times, and then the application should say "Goodbye" and quit.

To continue, see Using the VoiceXML Parse expression on page 286.

Using the VoiceXML Parse expression

In this topic, you create an application, vparse, that shows how the Parse expression alters the source string. The application contains a custom variable named [PhoneNumber] that has the default value of 303-555-1212. The Parse expression is used to strip off the area code.

The vparse application does the following:

- Uses the Parse expression of the Set and Test node to strip off the area code from [PhoneNumber] and place it in a custom variable [AreaCode].
- Announces the area code.
- Announces the remaining seven-digit telephone number.
- Hangs up and quits.

To create the vparse application:

1. Create an application, naming it vparse.

2. Create the following required custom variables.

   - [PhoneNumber]. Character variable, with the Spoken As field set to number:digits, and a default value of 303-555-1212
   
   - [AreaCode]. Character variable, with the Spoken As field set to number:digits
3. Create the call flow to look like the sample call flow in the following figure. Name the nodes as in the sample. Specify the nodes as required. Use default values unless other values are indicated by the application requirements.

For the Set and Test node Parse expression, use a hyphen (-) as the Parsing Separator.

4. Create prompts and phrases for the nodes:

   — **Announce10digitPhoneNumber** initial prompt:

     The ten digit phone number is [PhoneNumber].

   — **AnnounceAreaCode** initial prompt:

     The area code is [AreaCode].

   — **Announce7DigitPhoneNumber** initial prompt:

     The remaining seven digit phone number is [PhoneNumber].

5. Verify and simulate the application.

   Notice the value of [PhoneNumber] after running a simulation.

6. When you are satisfied with the application, generate the code. If you have a development system, transfer the application, install it, and assign it to a voice channel.

7. Test the application.

To continue, see Using asynchronous events in VoiceXML applications on page 288.
Using asynchronous events in VoiceXML applications

The Asynchronous Event node puts the system in a state where, upon detecting a hang up or error event, it drops the call, executes a specified call flow, and terminates processing of the call. In this tour, you use the Asynchronous Event node to capture information about calls that end in caller hang ups.

To continue, see Defining asynchronous events in VoiceXML on page 288.

Defining asynchronous events in VoiceXML

In this topic, you create a VoiceXML application that prompts the caller for a nine-digit identification number, stores the number in a custom variable, and makes an announcement. When the caller hangs up, the application clears the variable.

The application you create, called \texttt{vasync}:

- Sets the system to look for a hang up.
- Makes an announcement that describes the Asynchronous Event node.
- Prompts the caller for a nine-digit identification (ID) number.
- Takes action based on caller response
  - If the ID number is successfully collected, makes an announcement.
  - If the ID number is not successfully collected, disconnects and quits.
  - If a caller hangs up, executes a call flow that clears the variable.

To create the \texttt{vasync} application:

1. Create a new VoiceXML application, naming it \texttt{vasync}.
2. Drag an Asynchronous Event node to the design pad and name it \texttt{WaitForHangUp}.
3. Add an Announcement node and name it \texttt{DescribeAsyncEvent}. Edit it as follows:
   a) Create the Initial Prompt to say:

   \begin{quote}
   Welcome to the Asynchronous Event node application. The Asynchronous Event node looks for the caller to hang up. When the caller hangs up, it executes a specified call flow.
   \end{quote}

   b) Set the Allow Interrupt field to False.
4. Create a new character variable named CIID with Spoken As set to number:digits.
5. Add a Prompt and Collect node and name it GetIDNumber. Edit it as follows:
   - Create an Initial Prompt that says:
     
     Please enter your nine digit social security number.

     - On the Response tab, set the following fields to the specified values:
       
       Maximum Length: 9
       
       Minimum Length: 9
       
       Save Response In: [CIID]

6. Add an Announcement node and name it AcceptedIDNumber. Edit it as follows:
   - Create an Initial Prompt that says:
     
     Your nine digit ID number has been accepted. If you hang up now, it will be added to a database table, along with the date of your call.

7. Add an Announcement node and name it Goodbye. Have it say something appropriate for ending the call.

8. Add a Disconnect Call node and name it HangUp.

9. Add a Return node and name it Quit.

10. Jump to the Goodbye node from the fail branch of GetIDNumber.

    When done, the call flow should look like the following figure:

    ![Call Flow Diagram]

    With the exception of the Asynchronous Event node named WaitForHangUp, the [Main] call flow is complete.
11. Drag a Call Flow node to the bottom of the [Main] call flow and name it ClearIDNumber, as shown in the following figure. Note that it does not connect to the existing call flow.

12. Edit the [ClearIDNumber] call flow.

13. Drag a Set and Test node to the design pad and name it ClearIDNumber.

14. Create an expression that assigns the value of 0 to the [CIID] variable.

   When you are finished, your [ClearIDNumber] call flow should look like the following figure:

   You do not need to add anything to the default branch of the ClearIDNumber node.

15. Return to the [Main] call flow.

16. Edit the Asynchronous Event node.
The Node Inspector, shown in the following figure, allows you to specify handling of Caller Hangup and Error events.

17. Select the [ClearIDNumber] call flow for **Caller Hangup**.

18. Save, verify, and simulate the application.

19. If you have a development system, generate code, transfer and assign the application.

20. Test the application by doing the following:
   
   — When prompted, enter the digits **123456789**, and then hang up before the announcement is finished.
   
   — When prompted, enter the digits **12345678** for each of the three times you are prompted for an ID number. The third time, hang up after entering the digits.
   
   — Do not respond for each of the three times you are prompted for an ID number. Try hanging up at three different points in the script without entering any digits.
   
   — Do not respond to any prompts, and do not hang up until you hear the phrase, "Goodbye."

To continue, see **Running TAS applications from VoiceXML** on page 292.
Running TAS applications from VoiceXML

TAS has capabilities that are not available in VoiceXML. To make use of these capabilities in VoiceXML applications you can call and execute TAS applications from VoiceXML applications.

The CallScript node calls and executes a TAS application on the target Avaya IR or CONVERSANT system. You can use the CallScript node to designate arguments to be passed to the TAS application to be executed. These arguments must be character-type variables. The called TAS application uses a Getarg external function to collect any variable data that is passed to it, and the Ret2vxml external function to return variable data and control to the calling VoiceXML application. Using the Getarg and Ret2vxml nodes on page 294 explains how to use the two TAS external functions in conjunction with the CallScript node.

The CallScript node cannot be used to call a VoiceXML application. IVR Designer includes another means of calling and executing VoiceXML applications. You can use the Save and Exec node and the Load Variables node to call another IVR Designer VoiceXML application and pass arguments to it. Passing data between VoiceXML applications on page 300 explains how to use these nodes.

To continue, see Using the Call Script node on page 292.

Using the Call Script node

In this topic, you create an application, vcalltas, that uses the CallScript node to pass control to one of two existing applications: tasapp1 or tasapp2.

The vcalltas application does the following:

- Prompts the caller to press 1 to run TAS application number 1 or to press 2 to execute TAS application number 2.
- Takes action based on caller response:
  - If the caller presses 1, the application passes control to the tasapp1 application, which starts executing.
  - If the caller presses 2, the application passes control to the tasapp2 application, which starts executing.

Both of the secondary applications, tasapp1 and tasapp2, do the following:

- Announce that vcalltas passed control to the new application (identifying which application has been called).
• Disconnect and quit.

To create the **vcalltas** application:

1. Create a new VoiceXML application named **vcalltas**.
2. Drag a Menu node onto the design pad, and name it **MenuOfPrograms**.
3. Add a Return node to the fail branch, and name it **Quit**.
5. Create an initial prompt for the **MenuOfPrograms** Menu node that says:
   
   **To transfer to tas application one, press one now.**
   **To transfer to tas application two, press two now.**
6. Add a CallScript node to branch [1], name it **CallApplication1**, and set the Script Name to **tasapp1** in the Node Inspector.
7. Add a CallScript node to branch [2], name it **CallApplication2**, and set the Script Name to **tasapp2** in the Node Inspector.
8. Make the empty branches of the **CallApplication1** and **CallApplication2** nodes jump to the **Quit** node.

The finished application call flow should look like the following figure:

```
MenuOfPrograms
    1 CallApplication1
        ⇒ Quit
        ✗ Quit
    2 CallApplication2
        ⇒ Quit
        ✗ Quit
    ✗ Quit
```

9. Save and verify the application.
10. If you have a development system, generate code and transfer and install the application.

**Create the tasapp1 and tasapp2 applications**

To create each of the secondary (**tasapp1** and **tasapp2**) applications:

1. Create a new TAS application named **tasapp1** or **tasapp2**.
2. Drag an Announcement node to the design pad and name it `AnnounceBranch1Success` or `AnnounceBranch2Success`.

3. Create the initial prompt for the Announcement node to say the following:

   The v call tas application has passed control to tas application one (or two). Goodbye.

4. Add a Disconnect Call node and name it `Disconnect`.

5. Add a Return node and name it `Quit`, as shown in the following figure.

6. Save and verify the application.

7. If you have a development system, generate code and transfer and install the application.

   Do *not* assign either of these applications to a voice channel.

8. Test the `vcalltas` application and verify that it calls the correct TAS application, based on your menu choice.

   To continue, see *Using the Getarg and Ret2vxml nodes* on page 294.

### Using the Getarg and Ret2vxml nodes

The `Getarg` and `Ret2vxml` external functions are used in TAS applications called from VoiceXML applications using the CallScript node. These external functions get arguments that are passed to the TAS application and return arguments and control to the VoiceXML application. The passed arguments must be character-type variables.

In this topic, you use the CallScript node and the `Getarg` and `Ret2vxml` external functions to pass strings (character values contained in variables) between applications. You use three IVR Designer applications: `vclltas2`, `tasapp1`, and `tasappid`.

The primary VoiceXML application, `vclltas2`, uses the CallScript node to pass control to one of two other applications. It does the following:

- Prompts the caller to press 1 or to press 2.
- Takes action based on caller response:
  - If the caller presses 1, the application passes control to `tasapp1`, which starts executing, announces that `tasapp1` has control, disconnects, and quits.
— If the caller presses 2, the application prompts the caller for a 9-digit identification (ID) number, and passes control and the ID number to `tasappid`, which executes `Getarg` external functions that accept the process name and the ID number, announces the ID number, and returns control to `vclltas2`, which announces the value of the returned ID number, disconnects, and quits.

To create the `vclltas2` application:

1. Open the `vcalltas` application from the previous topic and save it as `vclltas2`.
2. Create two custom character variables:
   - `[ProcessName]`
   - `[ReturnedID]` spoken as `number`
3. Delete the `CallApplication2` node.
   
   Note that the `tasapp1` application and branch already exist and do what you want, so there is no need to modify them.

4. Add a Prompt and Collect node to branch [2] of the `MenuOfPrograms` node, name it `EnterID`, and edit the node as follows:

   — The prompt text for the initial prompt should say:

     **Please enter your nine digit ID number.**

   — On the **Response** tab, set the minimum and maximum lengths to 9.
5. Add a CallScript node to the pass branch of the EnterID node. Name it Call_tasappid and edit it as shown in the following figure:

6. Add an Announcement node to the pass branch under Call_tasappid, name it AnnounceReturn, and set the initial prompt to:

   The ID number returned to this application is, [ReturnedID].

7. Add a Disconnect Call node to the pass branch under AnnounceReturn and name it Disconnect.

8. Complete the vcltas2 application by making the fail branch of the Call_tasappid node, the pass branch of the Disconnect node, and the fail branch of the EnterID node jump to the Quit node.
When finished, the `vcltas2` call flow should look like the following figure:

9. Save and verify the application.

10. If you have a development system, generate code and transfer, and install the application.

**Create the tasappid application**

For this topic, you reuse the `tasapp1` application that you created for branch [1] of Using the Call Script on page 292pt node. You do, however, create a new TAS application for branch [2].

To create the new application:

1. Create a new TAS application, named `tasappid`.

2. Import private copies of the `Getarg` and `Ret2vxml` external functions.

   These external functions should be displayed on the External Functions Palette and in the External Functions Manager.

   For more information on importing external functions, see Make_Call.

3. Create custom character variables:

   — `[ReceivedID]` spoken as **AC** with a length of 9

   — `[ProcessName]` spoken as **A** with a length of 24
4. Drag a Getarg node to the design pad. Name it **GetProcessName**, and edit it as follows:

   — Arg. Number: 1 (Enter as a literal value.)

   The CallScript node uses Passing Variables 1 to pass the process name to this external function. Using the drop-down menu for this field, notice that all available variables are number variables.

   — Address: [ProcessName]

   The [ProcessName] variable stores the value of the first argument that is passed. Notice that all the variables available for use in this field are character variables.

   — Length: 24

5. Add a Getarg node, name it **GetID**, and edit it as follows:

   — Arg. Number: 2 (Enter as a literal value.)

   Passing Variables 2 in the vclltas2 application passes the value of the [CollectedDigits] variable.

   — Address: [ReceivedID]

   — Length: 9

6. Add an Announcement node after the **GetID** node, name it **AnnounceID**, and set the initial prompt to:

   **The ID number passed to this application is, [ReceivedID]** .
7. Add a Ret2vxml external function node, name it **ReturnTo_vclltas2**, and edit it as shown in the following figure:

The process name and ID number are returned to the calling VoiceXML application.

8. Add a Disconnect Call node to the call flow, and name it **Disconnect**.

9. Add a Return node, and name it **Quit**.
When finished, the call flow should look like the following figure:

![Call Flow Diagram](image)

10. Save and verify the application.
11. Simulate the `vclltas2` application.
   Note how control is passed between the applications.
12. If you have a development system, generate code and transfer and install the application.
   Do not assign the `tasappid` application to a voice channel.
13. Test the `vclltas2` application.
   To continue, see Passing data between VoiceXML applications on page 300.

### Passing data between VoiceXML applications

The Save and Exec node calls another IVR Designer application on the voice system and the Load Variables node loads variable data passed from another application. For instance, account information gathered from callers can be passed to another IVR Designer application.

To continue, see Save and Exec node and Load Variables node in VoiceXML on page 300.

### Using the Save and Exec node and Load Variables node in VoiceXML

In this topic, you develop an application that uses the Save and Exec node and the Load Variables node. The application uses the Save and Exec node to pass control and existing variable data to a second IVR Designer application. The second IVR Designer application accepts (loads) the passed variable data and announces the value of the data. The second application does not return control to the first application.
You create two IVR Designer applications, vsvexec and vldvar.

The primary application, vsvexec, does the following:

- Sets the value of a variable to be passed to the vldvar application.
- Announces the purpose of the application.
- Uses a Save and Exec node to pass control (without return) and the values of three variables to the vldvar application.

The secondary application, vldvar, does the following:

- Uses a Load Variables node to accept the variable values from the vsvexec application and places them in the new variables.
- Announces the values of the variables and quits.

**Create the vsvexec application**

You create the vsvexec application to pass control and the values of the three variables to the vldvar application using a Save and Exec node. When you define the Save and Exec node in an application, you select variables within that application and place them as numbered elements to pass to the called application. The data passed by the Save and Exec node must be of the same type in both applications, or the values may be corrupted.

To create the vsvexec application:

1. Create a new VoiceXML application, naming it vsvexec.
2. Create the custom variables for this application:
   - [Variable1]. Character variable with a default value of Vasco da Gama
   - [Variable2]. Character variable with no default value (the value is set to 303-555-1212 using a Set and Test node)
   - [Variable3]. Number variable with a default value of 1497
3. Drag a Set and Test node onto the design pad and name it SetVariablesToBePassed. Use this node to assign a value of 303-555-1212 to the [Variable2] variable.
4. Add an Announcement node and name it AnnouncePurpose. Create an initial prompt to describe what this application is and what it does.
5. Add a Save and Exec node, name it ExecuteVLDVAR, and edit it as follows:
   - Script Name: vldvar
   - Return Control: False
VoiceXML application development tours

— Three variables are passed. Highlight the **Name** field and click + to add three elements.

Element (1): [Variable1]

Element (2): [Variable2]

Element (3): [Variable3]

When finished, the call flow should look like the following figure:

![Call Flow Diagram](image)

6. Verify, simulate, and save the application.

7. If you have a development system, generate code and transfer and assign the application.

Create the **vldvar** Application

You create an application named **vldvar** to accept (load) the values of the variables sent from the **vsvexec** application. The **vldvar** application does not return control to **vsvexec**. You use a Load Variables node to accept the values and load the numbered elements. It does not use the names of the original variables, so you may designate new variable names into which to load the elements. The elements are received in the same order as in the Save and Exec node that passed them. Keep that in mind when naming the nodes in the new application.

To create the **vldvar** application:

1. Create the custom variables for this application.
   
   — [**Name**]. Character variable
   
   — [**PhoneNumber**]. Character variable, Spoken As **number:digits**
   
   — [**NumberValue**]. Number variable

2. Drag a Load Variables node onto the design pad and name it **LoadVariablesFromVsvexec**. Edit it as follows:
Advanced VoiceXML application development tours

— Highlight the **Name** field and double-click the + icon to add three elements. Assign these elements to the following variables:

   Element (1): [Name]
   Element (2): [PhoneNumber]
   Element (3): [NumberValue]

3. Add an Announcement node and name it **SayValues**. The initial prompt text should say the following:

   You are now in the second program.
   The value of the Name variable is, [Name] .
   The value of the Phone Number variable is, [PhoneNumber] .
   The value of the Number Value variable is, [NumberValue] .

4. Add a Disconnect node and a Return node.

   When finished, the call flow for **vldvar** should look like the following figure:

5. Verify and simulate the application.

6. If you have a development system, generate code and transfer the application.
   
   Do not assign the application to a voice channel. (You want **svexec** to be assigned to the channel.)

**Test and trace the vsvexec application**

To test and trace the **vsvexec** application:

1. Connect to the Avaya IR or CONVERSANT system and log in.
2. Start a trace with the command (where ## is the assigned channel):
trace ## area SE | tee /tmp/vsvexec-##.1out

3. Start a trace with the command (where ## is the assigned channel):

trace ## | tee /tmp/vsvexec-##.2out

To continue, see Using voice input in VoiceXML applications on page 304.

Using voice input in VoiceXML applications

VoiceXML is a Web-based language that uses a voice browser to interact with users. For this reason, voice input is a frequent component of VoiceXML applications. VoiceXML grammars specify what to listen for after a caller is prompted to enter information.

In this tour, you create custom VoiceXML grammars and use them in an application that accepts voice input from users. This type of application requires a Speech Recognition engine installed with the Avaya IR system.

To continue, see Creating custom VoiceXML grammars on page 304.

Creating custom VoiceXML grammars

IVR Designer includes standard VoiceXML grammars for accepting voice input. These grammars support both DTMF, which listens for touchtone input and voice entry. The standard VoiceXML grammars, which are adapted to the current locale, include the following:

- **boolean.** Affirmative and negative phrases
- **date.** Dates, including month, day, and year.
- **digits.** Digits, 0 through 9
- **currency.** Phrases that specify a currency amount
- **number.** Strings of digits that may include a decimal point
- **phone.** Phrases that specify a telephone number
- **time.** Phrases that specify a time, including hours and minutes

For more information on VoiceXML grammars, see Grammars for VoiceXML applications.

With IVR Designer, you can also create custom VoiceXML grammars as needed to support the input expected by an application.
In this topic, you create VoiceXML grammars for use in the voicein application. The voicein application:

- Welcomes the caller.
- Prompts the caller for a telephone number.
- Prompts the caller for the size of the pizza to be ordered, with an option to cancel the order.
- Prompts the caller for toppings, with options to list toppings or indicate that they are done with toppings.
- When the caller is done specifying toppings, announces the complete pizza order.
- When the caller hangs up, passes the telephone number, and pizza order to another application.

To create the custom VoiceXML grammars:

1. Create a new VoiceXML application and name it voicein.
2. Right-click the VXML Grammars manager and select New.
3. Name the grammar PizzaSizeGrammar.
4. Type the words for the grammar as shown below, pressing Enter after each word.

5. Click OK to accept this grammar.
6. Create a new grammar naming it **ToppingGrammar** and adding the following words:
   - pepperoni
   - sausage
   - green peppers
   - onions
   - olives
   - extra cheese
   - mushrooms
   - list
   - done

7. Save the application.

These grammars can now be used in the application along with the standard grammars.

To continue, see Using VoiceXML grammars for voice input on page 306.

### Using VoiceXML grammars for voice input

In this topic, you complete the **voicein** application, using the grammars you created in Creating custom VoiceXML grammars on page 304.

1. Create the custom variables required for the **voicein** application:
   - **[DeliveryTime]**. Character variable, Spoken As duration:m, with a default value of 30
   - **[PhoneNumber]**. Character variable, Spoken As telephone
   - **[PizzaSize]**. Character variable
   - **[ToppingList]**. Character variable

2. Create the following prompts:
   - **[AddToppingBadInputPrompt]**:
     We do not carry the topping you requested. Please try again.
   - **[AddToppingInitialPrompt]**:
     Please say a topping you want to add to your pizza. Say list to
get a list of the available toppings. Say done when you are finished.

- [AddToppingTimeoutPrompt]:
  Please say the name of a topping.

- [AnnounceAddedToppingInitialPrompt]:
  We will add [CollectedDigits] to your pizza.

- [AnnounceErrorInitialPrompt]:
  We cannot complete your pizza order at this time. Please call our non-automated service number to place your order. Goodbye.

- [AnnouncePizzaCompleteInitialPrompt]:
  Thank you. Your pizza is complete. We will make a [PizzaSize] pizza with the following toppings: [ToppingList]. Your order will arrive in [DeliveryTime] or sooner.

- [AnnounceToppingListInitialPrompt]:
  The following toppings are available: pepperoni, sausage, green pepper, onions, olives, mushrooms, and extra cheese.

- [GetPhoneNumberBadInputPrompt]:
  That was not a valid phone number.

- [GetPhoneNumberConfirmInputPrompt]:
  The phone number you provided is [PhoneNumber]. If this is correct, say yes or press one. If this is incorrect, say no or press two.

- [GetPhoneNumberInitialPrompt]:
  Please say or key in your ten digit phone number.

- [GetPhoneNumberTimeoutPrompt]:
  I did not get your phone number.

- [GetPhoneNumberTooFewDigitsPrompt]:
  I need a full ten digit phone number with area code.

- [GetPizzaSizeBadInputPrompt]:
  Please say small, medium, or large.

- [GetPizzaSizeInitialPrompt]:
  Would you like a small, medium, or large pizza? Or say cancel to end your order.

- [WelcomeInitialPrompt]:
  Welcome to the automated pizza line.
3. Create the [Main] call flow to look like the sample call flow shown in the following figure, using the prompts you created:

---

[Diagram of a VoiceXML call flow]

**Note:**

Before you can insert the named branches (list, done, and cancel), you must specify the VoiceXML grammars to use for the Prompt and Collect Nodes. To insert the named branches, click the icon on the Touchtones window and select the appropriate name from the Grammar Word List.
4. Edit the [EndCall] call flow to look like the sample call flow shown in the following figure:

![Call Flow Diagram]

5. Edit the EndCallAndPassData node to specify [EndCall] as the call flow for Inbound Call Terminated.

6. Edit the GetPhoneNumber node:
   - Change the Response parameters as shown in the following figure:

   ![Node Inspector]

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Interrupt</td>
<td>True</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>10</td>
</tr>
<tr>
<td>Minimum Length</td>
<td>10</td>
</tr>
<tr>
<td>Terminate Response With</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeouts</td>
<td></td>
</tr>
<tr>
<td>First Digit Timeout</td>
<td>15</td>
</tr>
<tr>
<td>Inter-Digit Timeout</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Tries</td>
<td></td>
</tr>
<tr>
<td>Tries Available</td>
<td>[Tries]</td>
</tr>
<tr>
<td>Tries Taken</td>
<td>[Attempts]</td>
</tr>
<tr>
<td>Replay Initial Prompt</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Caller's Response</td>
<td></td>
</tr>
<tr>
<td>Save Response In</td>
<td>[PhoneNumber]</td>
</tr>
<tr>
<td>Save Raw Response In</td>
<td></td>
</tr>
<tr>
<td>Recognizer Confidence</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Input ASR Mode</td>
<td>Phone</td>
</tr>
<tr>
<td>Mode Chosen</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Prompts</td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td></td>
</tr>
<tr>
<td>Confirmation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Notes</td>
<td></td>
</tr>
</tbody>
</table>
Select **Phone** from the drop-down menu in the **Input ASR Mode** field to identify Phone as the standard Phone VoiceXML grammar to use for this node.

— Continue to edit the **GetPhoneNumber** node, changing the Confirmation parameters as shown in the following figure:

```
Select **Boolean** for **Confirmation ASR Input** to identify a yes or no confirmation input, and set **Yes Digit** to **true** and **No Digit** to **false**.

7. Edit the **GetPizzaSize** node:

— Use the same prompt for Bad Input and Timeout.
— Complete the Response tab as shown in the following figure:

![Node Inspector]

- Input Parameters:
  - Allow Interrupt: True
  - Maximum Length: 5
  - Minimum Length: 5
  - Terminate Response With: 

- Timeouts:
  - First Digit Timeout: 5
  - Inter-Digit Timeout: 3

- Tries:
  - Tries Available: [Tries]
  - Tries Taken: [Attempts]
  - Replay Initial Prompt: True

- Caller's Response:
  - Save Response In: [PizzaSize]
  - Save Raw Response In:
  - Recognizer Confidence:
  - Input ASR Mode: VoiceXML Grammar PizzaSizeGrammar
  - Mode Chosen:

- Prompts / Response / Confirmation:

- Node Notes:

In this case, you use the custom **PizzaSizeGrammar** for input.

8. Edit the **AddTopping** node:

   — Use the same prompt for Bad Input and Timeout.
— Complete the Response tab as shown in the following figure:

In this case, you use the custom **ToppingGrammar** for input.

9. Edit the **AddToppingToList** node with two expressions:

   — concat: [CollectedDigits] = [CollectedDigits] concatenated with a blank space : maximum length of 0

   — concat: [ToppingList] = [ToppingList] concatenated with [CollectedDigits] : maximum length of 0
10. Edit the **SendCallData** node in the **[EndCall]** call flow as shown in the following figure:

```
<table>
<thead>
<tr>
<th>SendCallData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script Name</td>
</tr>
<tr>
<td>Return Control</td>
</tr>
<tr>
<td>Put Return Value</td>
</tr>
<tr>
<td>Request Method</td>
</tr>
<tr>
<td>Variables</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>(1)</td>
</tr>
<tr>
<td>(2)</td>
</tr>
<tr>
<td>(3)</td>
</tr>
</tbody>
</table>
```

The **ordata** script is another application you would create to receive the values of the variables so the pizza order can be filled. You will not create this now.

11. Verify, simulate, and save the application.

To continue, see Using database operations in VoiceXML applications on page 313.

---

**Using database operations in VoiceXML applications**

In VoiceXML applications, you can use Inline Code nodes to perform database operations on internal or external databases.

To continue, see Using Inline Code nodes to perform database operations on page 314.
Using Inline Code nodes to perform database operations

In this topic, you specify arguments for Inline Code nodes in a payroll application. The `vpay` application allows payroll clerks to administer a payroll history database, named PAY, which includes the following fields for each employee record:

- EMPNUM - employee number
- PAY - hourly pay rate
- EFFDATE - effective date for this pay rate

New records are created when new pay rates are assigned, so an employee may have multiple records in the database. Records with previous pay rates are not deleted when new pay rates are assigned. Instead, the record with the latest effective date is used in payroll calculations.

The `vpay` application does the following:

- Prompts the caller for an employee number.
- Retrieves a record for that employee number and reads the pay rate and effective date in that record.
- Prompts the caller to select from menu options to:
  - Update the record
  - Delete the record
  - Access the next record for this employee
  - Access a record for another employee
  - Quit
- Uses Inline Code nodes to perform these operations.

Review the call flow and globals

Begin by opening the `svpay` application and reviewing it. The `svpay` application includes all required nodes in the call flow. In this tour, you specify the arguments for some of the nodes.

1. Open the `svpay` application and save it as `vpay`.
2. Review the current application:
   - Review the custom variables in the application.
These variables are used to create the parameter string for the stored procedure and to retrieve and speak the values from the database. Notice the Spoken As values of the variables.

— Review the prompts and phrases in the application.

Notice the variables included in the prompts.

— Open the node inspector and review the parameters for the `EnterEmployeeNumber` node.

Notice that the input must be exactly five digits. It is stored in both the `[CollectedDigits]` and `[EmployeeNumber]` variables.

**Define the SELECT operation**

The SELECT operation chooses records that meet the specified criteria and retrieves data from the first record.

1. Locate the `SetSelectParameters` node and specify its parameters:

   — Assign `PAY` to the `[TNAME]` variable.

     The `[TNAME]` variable identifies the name of the table to be searched.

   — Concatenate `EMPNUM=` with the `[EmployeeNumber]` variable and place the result in the `[CRITERIA]` variable. Set the maximum length to 0.

     The database will be searched for records with an employee number equal to that in the `[EmployeeNumber]` variable. When the maximum length is set to 0, the length of the destination variable determines the maximum length.

   — Assign `EMPNUM,PAY,EFFDATE` to the `[GETCOLUMNS]` variable.

     This variable is used to identify the fields to be retrieved in the selected record.

   — Assign `SELECT` to the `[CMD]` variable.

     The `[CMD]` variable specifies the type of operation to perform. The SELECT operation locates a record that meets the specified criteria.

2. Locate the `GetEmployeeData` node and view the associated code.
This Inline Code node executes the code listed below to select records that meet the specified criteria and retrieve data from the first record.

```xml
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get" maxage='0' namelist="CHANNEL CMD TNAME CRITERIA GETCOLUMNS DIPNAME">
  <filled>
    <assign name="document.RetEmpNum" expr="MyServlet.EMPNUM"/>
    <assign name="document.RetPay" expr="MyServlet.PAY"/>
    <assign name="document.RetEffDate" expr="MyServlet.EFFDATE"/>
  </filled>
</subdialog>
<catch event="error.badfetch">
  <prompt> Could not locate specified URL </prompt>
  <goto expr="#d3f8ult3z1tF0rm"/>
</catch>
```

The inline code passes a standard set of parameters to the database. If a record that meets the specified criteria is found, the code places the data into the defined variables. If database access is a problem, the `RetCode` variable is assigned a negative value. If the specified URL cannot be found, the code plays an announcement and then branches to the end of the application. You can modify the code to specify different return values or different error handling.

**Note:**
Development of code like the preceding sample is beyond the scope of this tour.

3. Locate the `FormatData` node and review its parameters.

   This node places the returned values for pay and effective data into variables with appropriate Spoken As formats.

4. Locate the `SelectAction` node and review its parameters.

   Notice the options available to the payroll clerk.

**Define the UPDATE operation**

The UPDATE operation makes changes to a specified field in a record that matches defined criteria. For purposes of this example, assume a payroll clerk would update a record if data had been entered incorrectly. The payroll clerk can choose to update the pay field or the effective date field. In either case, the application prompts for the correct information.

1. Locate the `SetPayUpdate` node and specify its parameters:

   — Assign PAY to the `[TNAME]` variable.
The [TNAME] variable identifies the name of the table to be searched.

— Concatenate EMPNUM= with the value of the [EmployeeNumber] variable and place the result in the [CRITERIA] variable. Set the maximum length to 0.

— Concatenate the value of [CRITERIA] with ,PAY= and place that value back in [CRITERIA]. Set the maximum length to 0.

— Concatenate the value of [CRITERIA] with the value of [RetPay] and place that value back in [CRITERIA]. Set the maximum length to 0.

These criteria select the record with the given employee number and pay values.

— Concatenate PAY, with the value of the [CollectedDigits] variable and place the result in the [SETCOLUMNS] variable. Set the maximum length to 0.

The PAY field will be updated with the value in the [CollectedDigits] variable.

— Assign UPDATE to the [CMD] variable.

The [CMD] variable specifies the UPDATE operation, which makes a change to the specified field of a record.

2. Locate the **UpdatePay** node and view the associated code.

This Inline Code node executes the code listed below to update the PAY field.

```xml
<var name="CHANNEL" exp="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get" maxage='0s' namelist="CHANNEL CMD TNAME CRITERIA SETCOLUMNS DIPNAME">
    <filled>
        <assign name="document.RetCode" exp="MyServlet.retcode"/>
    </filled>
</subdialog>
<catch event="error.badfetch">  
    <prompt>Could not locate specified URL</prompt>
    <goto exp="#d3f8ult3l1fF0rm1"/>
</catch>
```

The only output of this operation is the return code, and its value whether the update operation was successful. A negative return code implies an error condition. If the update fails, the application plays an announcement and quits. If the update succeeds, the caller is returned to the menu to select another update or to select a different action.

3. Locate the **SetDateUpdate** node and specify its parameters:

— Assign PAY to the [TNAME] variable.
— Concatenate EMPNUM= with the value of the [EmployeeNumber] variable and place the result in the [CRITERIA] variable. Set the maximum length to 0.

— Concatenate the value of [CRITERIA] with ,EFFDATE= and place that value back in [CRITERIA]. Set the maximum length to 0.

— Concatenate the value of [CRITERIA] with the value of [RetEffDate] and place that value back in [CRITERIA]. Set the maximum length to 0.

— Concatenate EFFDATE, with the value of the [CollectedDigits] variable and place the result in the [SETCOLUMNS] variable. Set the maximum length to 0.

The EFFDATE field will be updated with the value in the [CollectedDigits] variable.

— Assign UPDATE to the [CMD] variable.

4. Locate the **UpdateDate** node and view the associated code.

This Inline Code node executes the same code to update the EFFDATE field. Note that both the PAY and the EFFDATE fields could have been updated in a single operation.

```xml
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get" maxage='0' namelist="CHANNEL CMD TNAME CRITERIA SETCOLUMNS DIPNAME">
    <filled>
        <assign name="document.RetCode" expr="MyServlet.retcode"/>
    </filled>
</subdialog>
<catch event="error.badfetch">
    <prompt>Could not locate specified URL</prompt>
    <goto expr="#d3f8ult3z1tF0rm"/>
</catch>
```

**Define the DELETE operation**

The DELETE operation deletes a record that matches specified criteria. For purposes of this example, assume that records are deleted if a new pay record is added when the rate has not actually changed.

1. Locate the **SetDeleteParameters** node and specify its parameters:
   — Assign PAY to the [TNAME] variable.
   — Concatenate EMPNUM= with the value of the [EmployeeNumber] variable and place the result in the [CRITERIA] variable. Set the maximum length to 0.
— Concatenate the value of [CRITERIA] with ,EFFDATE= and place that value back in [CRITERIA]. Set the maximum length to 0.

— Concatenate the value of [CRITERIA] with the value of [RetEffDate] and place that value back in [CRITERIA]. Set the maximum length to 0.

— Assign DELETE to the [CMD] variable.

The [CMD] variable specifies the DELETE operation, which deletes the specified record.

2. Locate the **DeleteRecord** node and view the associated code.

This Inline Code node executes the code listed below to delete the specified record.

```xml
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get" maxage="0s" nameList="CHANNEL CMD TNAME CRITERIA DIPNAME">
  <filled>
    <assign name="document.RetCode" expr="MyServlet.recode"/>
  </filled>
</subdialog>
<catch event="error.badfetch">
  <prompt>Could not locate specified URL</prompt>
  <goto expr="#d3flult3xlF0rm"/>
</catch>
```

If the delete operation fails, the application plays an announcement and quits. If the operation succeeds, the application returns to the action menu.

**Define the NEXT operation**

The NEXT operation repeats the previous SELECT operation.

1. Locate the **SetNextParameters** node and specify its parameters:

   This is the first node encountered when the payroll clerk chooses to go onto the next record.

   — Assign NEXT to the [CMD] variable.

   The [CMD] variable specifies the NEXT operation, which repeats the SELECT operation and finds the next record for the employee number.

2. Locate the **GetNextEmployeeRecord** node and view the associated code.
This Inline Code node executes the code listed below to select the next record.

```xml
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get"
maxage='0' namelist="CHANNEL CMD DIPNAME">
  <filled>
    <if cond="MyServlet.recode &lt;= '0'">
      <assign name="document.RetEmpNum" expr="MyServlet.EMPNUM"/>
      <assign name="document.RetPay" expr="MyServlet.PAY"/>
      <assign name="document.RetEffDate" expr="MyServlet.EFFDATE"/>
    </if>
  </filled>
</subdialog>
<catch event="error.badfetch">
  <prompt>Could not locate specified URL</prompt>
  <goto expr="#d3f8ult3z1tF0rm"/>
</catch>
```

The inline code passes fewer parameters to the database than the original SELECT operation does. The other parameters values used by the database are the same values as those for the previous SELECT command. The inline code retrieves the same outputs.

If there are no more records for the given employee number, the application gives the caller the option to create a new record for this employee, to access data for a different employee, or to quit.

If the caller chooses to create a new record, the application prompts for the new pay rate and the new effective date. Notice these nodes in the call flow.

**Define the INSERT operation**

The INSERT operation adds a new record with specified field values.

1. Locate the **SetInsertParameters** node and specify its parameters:
   - Assign `PAY` to the `[TNAME]` variable.
   - Concatenate `EMPNUM`, with the value of the `[EmployeeNumber]` variable and place the result in the `[SETCOLUMNS]` variable. Set the maximum length to 0.
   - Concatenate the value of `[SETCOLUMNS]` with `,PAY`, and place that value back in `[SETCOLUMNS]`. Set the maximum length to 0.
   - Concatenate the value of `[SETCOLUMNS]` with the value of `[NewPay]` and place that value back in `[SETCOLUMNS]`. Set the maximum length to 0.
   - Concatenate the value of `[SETCOLUMNS]` with `,EFFDATE`, and place that value back in `[SETCOLUMNS]`. Set the maximum length to 0.
— Concatenate the value of [SETCOLUMNS] with the value of [CollectedDigits] and place that value back in [SETCOLUMNS]. Set the maximum length to 0.

— Assign INSERT to the [CMD] variable.

The [CMD] variable specifies the INSERT operation, which inserts a record with the specified field values.

2. Locate the **InsertRecord** node and view the associated code.

This Inline Code node executes the code listed below to insert a record.

```xml
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method='get' maxage='0s' namelist="CHANNEL CMD TNAME SETCOLUMNS DIPNAME">
  <filled>
    <assign name="document.RetCode" expr="MyServlet.retcde"/>
  </filled>
</subdialog>
<catch event="error.badfetch">
  <prompt>Could not locate specified URL</prompt>
  <goto expr="#d3f8ult3x1tF0rm"/>
</catch>
```

If the insert operation fails, the application plays an announcement and quits. If the operation succeeds, the application returns to the action menu.

The remaining nodes are complete. This completes the **vpay** application.

3. Simulate, verify, and save the application.

Before this application will work, the database must exist on the target system.

To continue, see Using stored procedures in VoiceXML applications on page 322.

---

**Using VoiceXML 2.1 tags in VoiceXML applications**

In VoiceXML applications, you can use VoiceXML 2.1 tags.

For more information, see Using Inline Node to perform operations using VoiceXML 2.1 tags on page 322.
Using Inline Node to perform operations using VoiceXML 2.1 tags

This describes the usage of <for each> and <Data> tag in the VXML applications. For more information, see http://www.w3.org/TR/voicexml21/#sec-data and http://www.w3.org/TR/voicexml21/#sec-foreach.

Using stored procedures in VoiceXML applications

In VoiceXML applications, you can use Inline Code nodes that call stored procedures to:

- Execute procedures stored in an internal or external database
- Pass values to and retrieve values from those procedures

For example, you can retrieve account balances, locations, or shipping dates.

To continue, see Using Inline Code nodes to execute stored procedures on page 322.

Using Inline Code nodes to execute stored procedures

In this topic, you specify arguments for stored procedure nodes in an application that allows callers to find store locations near a specified zip code. The vlocate application executes the LOCATE stored procedure, which exists on a database.

The vlocate application does the following:

- Prompts the caller for a zip code.
- Creates a parameter string that includes the zip code.
- Uses an Inline Code node to execute the LOCATE stored procedure on a remote database to retrieve the street, city, and state of a store close to the zip code.
- Announces the street, city, and phone number to the caller.
- Uses an Inline Code node to retrieve the next record returned by the previous stored procedure call.
- Announces the street, city, and phone number to the caller and continues to check for additional records.

To complete the vlocate application:
1. Open the svlocate application and save it as vlocate.

The svlocate application includes all nodes in the call flow. In this tour, you specify the arguments for some of the nodes.

2. Review the current application.

   — Review the custom variables in the application.

      These variables are used to create the parameter string for the stored procedure and to retrieve and speak the values from the database. Notice the default values of [PARAM1] and [PARAM2]. Also notice the Spoken As values of the variables.

   — Review the prompts and phrases in the application.

      Notice the variables included in the prompts.

   — Open the node inspector and review the parameters for the GetZipCode node.

      Notice that the input must be exactly five digits.

3. Locate the SetParameters node and specify its parameters:

   — Assign STOREDPROC to the [CMD] variable.

      The [CMD] variable specifies the type of operation to perform.

   — Assign DBDIP2 to the [DIPNAME] variable.

      [DIPNAME] specifies the database DIP on the target platform that is configured to access the database with the stored procedure. In this example, DBDIP2 has been used.

   — Assign LOCATE to the [SPNAME] variable.

      The stored procedure that is executed by this application is named LOCATE.

   — Assign the value of [PARAM1] to the [PARAMS] variable.

   — Concatenate the value of [PARAMS] with the value of [CollectedDigits] and place that value back in [PARAMS]. Set the maximum length to 0.

      When the maximum length is set to 0, the length of the destination variable determines the maximum length.

   — Concatenate the value of [PARAMS] with the value of [PARAM2] and place that value back in [PARAMS]. Set the maximum length to 0.
These three elements create the parameter string for the stored procedure into the
[PARAMS] variable. For example, if the zip code in [CollectedDigits] is 12345, the
parameter string is

STRING,12345,IN,STRING,STREET,OUT,STRING,CITY,OUT,STRING,PHONE,OUT

The input to the stored procedure is the zip code. The outputs from the stored
procedure are the street, city, and phone number for a store near the zip code. The
STRING keyword in the above example specifies the data type of the individual
parameter passed to or returned from the stored procedure.

— Assign NO to the [SPRETCODE] variable.

The StoredProc call generates a return code to identify the success of its execution of
the stored procedure. In some cases, the stored procedure may also return a numeric
return code. When the stored procedure returns its own return code, SPRETCODE
should be set to YES. In this example, the stored procedure does not return its own
return code, so SPRETCODE is set to NO.

— Assign YES to the [SPRESULTSET] variable.

If the stored procedure returns a result set that may include multiple records,
SPRESULTSET is set to YES. In this example, there may be multiple records for the
same zip code.

4. Locate the GetFirstStore node and view the associated code.
This Inline Code node executes the code listed below to execute the stored procedure and retrieve the store location.

```xml
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method='get' maxage='0s' nameList="CHANNEL CMD SPNAME PARAMS SRESULT SET DIPNAME">
  <filled>
    <if cond="MyServlet.recode &gt; '0'">
      <assign name="document.RetCode" expr="MyServlet.recode"/>
      <assign name="document.STREET" expr="MyServlet.STREET"/>
      <assign name="document.CITY" expr="MyServlet.CITY"/>
      <assign name="document.PHONE" expr="MyServlet.PHONE"/>
    </if>
  </filled>
</subdialog>
<catch event="error badfetch">
  <prompt> Could not locate specified URL </prompt>
  <go to expr="#d3f3ult3x1tF0rm"/>
</catch>
```

The inline code passes a standard set of parameters to the database. If data from the database is accessible, the code places that data into the defined variables. If there is a problem with database access, the RetCode variable is assigned a negative value. If the specified URL cannot be found, the code plays an announcement indicating that and then branches to the end of the application. You can modify this code to specify different return values or different error handling.

5. Locate the SetNextParameters node and specify its parameters:
   - Assign NEXT to the [CMD] variable.
   - Assign DBDIP2 to the [DIPNAME] variable.

6. Locate the GetNextStore node and view the associated code.
This Inline Code node executes the code listed below to execute the stored procedure and retrieve the next store location for the zip code.

```
<var name="CHANNEL" expr="Channel"/>
<subdialog name="MyServlet" src="http://localhost/servlet/JdbcServlet" method="get" maxage='0s' namelist="CHANNEL CMD DIPNAME">
  <filled>
    <if cond="MyServlet.retcode&gt;0">
      <assign name="document.RetCode" expr="MyServlet.retcode"/>
      <assign name="document.STREET" expr="MyServlet.STREET"/>
      <assign name="document.CITY" expr="MyServlet.CITY"/>
      <assign name="document.PHONE" expr="MyServlet.PHONE"/>
    </if>
  </filled>
</subdialog>
<catch event="error.badfetch">
  <prompt>Could not locate specified URL</prompt>
  <goto expr="#3f3ult3x1tF0rm"/>
</catch>
```

This inline code passes fewer parameters to the database. It uses the parameter values from the previous STOREDPROC command. The inline code retrieves the same outputs as the GetFirstStore node.

7. Locate the MoreStores node and specify its parameters:

- Branch to the (NoMoreStores) branch if the value of the [ReturnCode] variable is equal to 0.

A return code of 0 indicates that there were no additional records found for the zip code. In this case, the application announces that there are no more stores and then hangs up and quits. If it receives a result, the application jumps to the Announce Location node and continues.

The remaining nodes are complete. This completes the vlocate application.

8. Simulate, verify, and save the application.

Before this application will work, the stored procedure must be created on a database on the target system.

To continue, see Testing and Troubleshooting VoiceXML applications on page 327.
Testing and Troubleshooting VoiceXML applications

The first steps in testing and troubleshooting a VoiceXML application are to verify and simulate the application. These steps identify potential problems before you generate code and assign the application to a channel.

After you assign an application on the Avaya IR or CONVERSANT system, you should test the application thoroughly before you place it in service. When you test an application, you can trace the flow of the application.

The topic in this section provides examples of how to trace applications.

To continue, see Tracing a VoiceXML application on page 327.

Tracing a VoiceXML application

The Avaya IR or CONVERSANT system contains a trace facility that you can use to follow the steps of an IVR Designer application.

To trace an application:

1. Assign the application to a channel on the Avaya IR or CONVERSANT system.
2. Connect to the Avaya IR or CONVERSANT system.
3. Enter the login, password, and terminal type for your system.
4. Check that the application you want to trace is listed using the following command:
   
   display services

5. Enter the following command (where ## is the channel number to which the application is assigned):

   trace ## | tee /tmp/application name-##.out

   The tee command writes output to standard output (generally to your monitor screen) and to a specified file.

   Unless otherwise specified, write all trace outputs to the /tmp directory so the trace output files are automatically deleted each time the system is rebooted.

6. Dial the number for the voice channel. When prompted, respond appropriately.

   To stop a trace, press the Delete key on the keyboard.
Trace script execution

When tracing an application script, you can view the script execution STEPs only, for trace output that is easier to read, while using up less disk space. The previous example showed a default trace of an executing script on a specific channel. A specific area may be traced. In the next example, area SE (Script Execution) is traced.

1. Enter the following command (where ## is the channel number to which the application is assigned):

   `trace ## area SE | tee /tmp/application name-##SE.out`

2. Dial the number for the voice channel. When prompted, respond appropriately.

   Note that the STEPs of the script execution are the only lines displayed in the trace.

   To stop the trace, press the **Delete** key on the keyboard.

This completes the VoiceXML application development tours.
Reference

Reference contains reference material including descriptions of menus, menu options, icons, palettes, nodes, and system variables.

This section includes the following topics:

Menus and menu options........................................................................................................ 330
Palettes .................................................................................................................................. 336
Avaya IVR Designer icons ...................................................................................................... 342
Node descriptions .................................................................................................................. 349
System variable descriptions ................................................................................................. 380
Menus and menu options

The IVR Designer user interface has the following menus:

- File
- Edit
- View
- Tools
- Help

This section includes the following topics:

File menu options ..........................................................................330
Edit menu options ..........................................................................331
View menu options ........................................................................333
Tools menu options ........................................................................334
Help menu options .........................................................................335

File menu options

The following table describes the options on the File menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Creates a new application.</td>
</tr>
<tr>
<td>Open...</td>
<td>Opens an existing application.</td>
</tr>
</tbody>
</table>

Note:
To open or save an application on another (networked) computer, you must have the appropriate read and write permissions for the application.
### Open Script Builder Application

Opens an existing Script Builder application on a CONVERSANT system, and converts the Script Builder application to an IVR Designer application. For more information about converting Script Builder applications, see Converting Script Builder applications.

**Note:**
To open a Script Builder application, you must have a LAN connection between the IVR Designer PC and the CONVERSANT system with the Script Builder application.

### Delete

Deletes the current application.

### Close

Closes the current application.

If you made any changes to the application since the last Save action, IVR Designer asks whether you want to save the application before closing it.

### Save

Saves the current application.

**Note:**
If you are attempting to open or save an application on another (networked) computer, you must first make sure the appropriate read/write permissions are set for the application you want to use.

### Save As...

Saves the application with a different name.

### Print...

Opens the **Print Reports** window, which lets you select which reports you want to print.

### (Recent Applications)

Lists recently opened applications and can be used as shortcuts to reopen them.

### Exit

Closes any open applications and quits IVR Designer.

If you made any changes to the application since the last Save action, IVR Designer asks whether or not you want to save the application before closing it.

## Edit menu options

The following table describes the options on the **Edit** menu.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Undo</strong></td>
<td>Undoes the last action or series of actions. Selecting <strong>Undo</strong> from the <strong>Edit</strong> menu or pressing the <strong>Undo</strong> icon on the <strong>Toolbar</strong> lets you undo (reverse) the last action or series of actions. The number of undo levels is limited only by the number of undoable actions performed since the last Save action. Actions are undone in reverse of the order in which they were done. <strong>Note:</strong> Some actions cannot be undone. If you place and leave the cursor over the Undo icon, a popup message appears telling you what action will be undone. The same message also appears in the status bar.</td>
</tr>
<tr>
<td><strong>Redo</strong></td>
<td>Redoes any actions that were undone using the <strong>Undo</strong> command. Selecting <strong>Redo</strong> from the <strong>Edit</strong> menu or pressing the <strong>Redo</strong> icon on the <strong>Toolbar</strong> lets you redo the last action or series of actions that were undone using the Undo feature. Redo is limited only by the number of Undo actions that have been performed. Actions are redone in reverse of the order in which they were undone (or, in other words, in the same order they were originally done). Once you perform another action after performing an Undo or a Redo action, Redo is no longer active or available. If you place and leave the cursor over the Redo icon, a popup message appears telling you what action will be redone. The same message also appears in the status bar.</td>
</tr>
<tr>
<td><strong>Cut</strong></td>
<td>Copies all node attributes and application resources (including encapsulated call flows) necessary to make the selected node or nodes fully functional when it is pasted into a new location. It removes only the node itself, and not the associated resources, from the original application. When you cut a single node, that node and all subordinate nodes attached to it are cut. To cut a single node without cutting its subordinate nodes, you must first disconnect the subordinate nodes, then cut the desired node.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Copies all node attributes and application resources (including encapsulated call flows) necessary to make the selected node or nodes fully functional when pasted into a new location. When you copy a single node, that node and all subordinate nodes attached to it are copied. To copy a single node without copying its subordinate nodes, you must first disconnect the subordinate nodes, then copy the desired node.</td>
</tr>
<tr>
<td><strong>Paste</strong></td>
<td>Copies all nodes currently on the clipboard to a selected open branch, or set of open branches in the case of multiple-node paste operations.</td>
</tr>
<tr>
<td><strong>Configure Toolbar...</strong></td>
<td>Opens the <strong>Configure Toolbar</strong> window which lets you customize the Toolbar.</td>
</tr>
<tr>
<td><strong>Configure Node Palette...</strong></td>
<td>Opens the <strong>Configure Node Palette</strong> window which lets you customize the Node Palette.</td>
</tr>
</tbody>
</table>
Menus and menu options

Find...

Searches across all call flows in the application to find:

- Text strings used in component names
- Locations where specified items are used
- Unused items
- Text strings

---

View menu options

Palette Options

If a palette is hidden (unchecked) using the View menu options and then displayed, the palette displays at the bottom of whatever palettes are currently displayed.

The following table describes the palette options on the View menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbar</td>
<td>Toggles between displaying and hiding the Toolbar. When checked, the Toolbar is displayed.</td>
</tr>
<tr>
<td>Node Palette</td>
<td>Toggles between displaying and hiding the Node Palette. When checked, the Node Palette is displayed.</td>
</tr>
</tbody>
</table>

Globals Manager options

Clicking on the name of a global resource manager displays that resource manager wherever it was last placed.

If any of the global resource managers are docked in the Globals Manager, clicking on their names in the View menu (or on their icons on the Toolbar) displays the entire Globals Manager. To display only one resource manager (Prompts, for example), you must first undock the resource manager from the Globals Manager.

The following table describes the Globals Manager options on the View menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Globals</td>
<td>Opens the Globals Manager.</td>
</tr>
<tr>
<td>Variables</td>
<td>Opens the Variables Manager.</td>
</tr>
<tr>
<td>Phrases</td>
<td>Opens the Phrases Manager.</td>
</tr>
<tr>
<td>Prompts</td>
<td>Opens the Prompts Manager.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VXML Grammars</td>
<td>Opens the VoiceXML Grammars Manager.</td>
</tr>
<tr>
<td>Tools menu options</td>
<td>The following table describes the options on the <strong>Tools</strong> menu.</td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Call Flow Inventory</td>
<td>The Call Flow Inventory tool lets you:</td>
</tr>
<tr>
<td></td>
<td>• See what call flows exist for your application</td>
</tr>
<tr>
<td></td>
<td>• See where each call flow is used within the application, and what other call flows (if any) are used within each call flow</td>
</tr>
<tr>
<td></td>
<td>• Locate a specific call flow within an application</td>
</tr>
<tr>
<td></td>
<td>• Provide an alternate means of dragging a call flow to the design pad</td>
</tr>
<tr>
<td>Code Generation/Application Transfer</td>
<td>The Code Generation tool generates the files necessary to transfer and install the application onto the target system.</td>
</tr>
<tr>
<td></td>
<td>The Application Transfer tool transfers the application files to the target system.</td>
</tr>
<tr>
<td>Node Inspector</td>
<td>The Node Inspector tools lets you:</td>
</tr>
<tr>
<td></td>
<td>• Edit node attributes easily and quickly.</td>
</tr>
<tr>
<td></td>
<td>• Display different information and editing options for each node, depending on the node's predetermined attributes.</td>
</tr>
<tr>
<td>Preference Editor</td>
<td>The Preference Editor lets you:</td>
</tr>
<tr>
<td></td>
<td>• Set application and toolwide preferences for using IVR Designer.</td>
</tr>
<tr>
<td></td>
<td>• Control the working environment for IVR Designer</td>
</tr>
<tr>
<td>Simulation</td>
<td>The Simulation tool lets you:</td>
</tr>
<tr>
<td></td>
<td>• Run your application in a simulated environment.</td>
</tr>
<tr>
<td></td>
<td>• Find problems within the application, thus reducing the amount of time required to test your application on the target system.</td>
</tr>
<tr>
<td>Verify Design</td>
<td>The Verify Design tool lets you:</td>
</tr>
<tr>
<td></td>
<td>• Check your application design for possible errors or omissions.</td>
</tr>
<tr>
<td></td>
<td>• View each call flow's results in a separate display.</td>
</tr>
<tr>
<td></td>
<td>• Easily locate nodes with possible errors or omissions.</td>
</tr>
</tbody>
</table>
Help menu options

The following table describes the options on the Help menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents...</td>
<td>Opens the IVR Designer Help Topics. From there, you can navigate to topics of interest.</td>
</tr>
<tr>
<td>Search for Help on...</td>
<td>Opens the online Help index. To find a help topic, type the first few letters of the topic you are looking for, then click <strong>Display</strong>.</td>
</tr>
<tr>
<td>About...</td>
<td>Displays the About Avaya IVR Designer information box, which contains a copyright notice and the Avaya IVR Designer version number.</td>
</tr>
</tbody>
</table>
Palettes

- IVR Designer has two palettes, each with its own function and set of icons. The palettes are located below the menu bar. The palettes are as follows:
  - The **Toolbar** – contains icons for accessing the IVR Designer menus
  - The **Node Palette** – contains node icons that you drag and drop onto the Design Pad to create your voice response applications

The methods for configuring these palettes are basically the same. Each palette can be configured independently as follows:

- Individual icons can be rearranged and displayed or hidden.
- Groups of icons can be rearranged.
- Palettes can be undocked from or docked to the menu bar.

This section includes the following topics:

- Opening a Palette Configuration window ................................................337
- Rearranging icons on a palette..............................................................337
- Using the Allow Scrolling option ..........................................................338
- Using the Tabs option............................................................................339
- Docking/undocking palettes.................................................................339
- Right-click menu options for palettes..................................................340
Opening a Palette Configuration window

To open a window for configuring a palette, select the appropriate **Configure [Palette]...** option from the **Edit** menu. IVR Designer displays the selected palette configuration window. The following illustration shows the **Configure Toolbar** window.

Rearranging icons on a palette

To customize the placement of icons on any of the palettes, you can:

- Move whole groups (categories) of icons at once.
- Move individual icons within a group.
- Choose whether or not each icon should be displayed.

**Moving categories or groups of icons**

To move a category or group of icons:

1. Open the appropriate palette configuration window.
2. Highlight in the **Category** pane the category (group) of icons you want to move.
3. To move the category of icons to the left on the palette, click the **Up** arrow.
4. To move the category of icons to the right on the palette, click the **Down** arrow.
5. Click **OK**.
Moving a single icon

To move a single icon:

1. Open the appropriate palette configuration window.
2. Highlight in the Items pane the item (individual icon) you want to move.
3. To move the item to the left on the palette, click the Up arrow.
4. To move the item to the right on the palette, click the Down arrow.
5. Click OK.

Selecting icons for display

To select or remove an icon for display:

1. Open the appropriate palette configuration window.
2. Highlight the Category of the icon you want to select or remove.
3. To select (display) an icon on the palette, check its box.
4. To remove (hide) an icon from the palette, clear its checkbox.

Using the Allow Scrolling option

The Allow Scrolling option causes the icons of a palette not to wrap if they do not fit in the width of the window. However, they can still be accessed using the scrolling arrows.

The following figure shows an example of a palette with the Allow Scrolling option turned on.

![Palette with Allow Scrolling option turned on]

To use the Allow Scrolling option, choose one of the following methods:

**Method 1**
1. Right-click anywhere on the appropriate palette.
2. Select Allow Scrolling from the menu.

**Method 2**
1. Open the appropriate palette configuration window.
2. Click the Allow Scrolling checkbox in the lower left corner of the window.
3. Click OK.
Using the Tabs option

The **Use Tabs** option displays only one category (group) of icons at a time on the palette. To view another category of icons, click the appropriate tab.

The following figure shows an example of a palette with the **Use Tabs** option turned on.

To use the Use Tabs option, choose one of the following methods:

**Method 1**
1. Right-click anywhere on the appropriate palette.
2. Select **Use Tabs** from the menu.

**Method 2**
1. Open the appropriate palette configuration window.
2. Click the **Use Tabs** checkbox in the lower left corner of the window.
3. Click **OK**.

Docking/undocking palettes

You can undock (remove) a palette from the menu bar at the top of the design pad and place it wherever you want. You can also dock (attach) it again either to the top or the bottom of the design pad window.

**Undocking a palette**

Use the following procedure to undock a palette:

1. Click-and-drag anywhere in the palette area. Be careful *not* to click an icon.
   
   When you first click and hold the primary mouse button down in the palette area, the icons disappear and a dashed outline appears around the entire palette area. As you drag the palette away from the menu bar, the dashed outline changes to a solid black outline. At this point you can release the mouse button, and the palette remains where you release the button.

2. Place the palette where you want it.
When you first release an undocked palette, it has the same configuration and general appearance as when it was docked.

Once it is undocked, you can resize the new palette window and place it wherever you want. Limitations on the minimum width and height of the undocked palette window are determined by the number of icons within the groups (categories) and the number of categories.

To resize the undocked palette, click and drag an edge or corner, the same as you would any other secondary window.

To move the undocked palette, click-and-drag the window by the title bar to the new location.

**Docking a palette**

Use the following procedure to dock a palette:

1. Click-and-drag the palette to the top or to the bottom of the design pad. Be careful to click the palette in a blank area between icons.

   When attempting to dock a palette, make sure you do not click-and-drag using the palette's title bar. If you do, you will not be able to dock the palette.

   When you first click and hold the primary mouse button down in the palette window, the icons disappear and a solid black outline appears around the entire window area. As you drag the window toward the menu bar or the bottom of the design pad, the solid black outline changes to a dashed outline.

2. Release the mouse button to attach the palette.

   **Note:**
   A newly docked palette always attaches itself at the top of the displayed palettes. Therefore, if you want a palette to be placed at the top of the palettes, you must dock it last.

---

**Right-click menu options for palettes**

The palette right-click menu is available by right-clicking anywhere on any of the palettes. It is not available within any of the Configure [Palette] windows.

The following table describes the options available on the palette right-click menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure...</td>
<td>This option opens the palette configuration window for the selected palette.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Hide</td>
<td>This option hides the selected palette from view.</td>
</tr>
<tr>
<td></td>
<td>To view the palette again, you must select the appropriate palette from the View menu.</td>
</tr>
<tr>
<td>Show Hints</td>
<td>This option toggles on and off. When checked (toggled on), it displays &quot;balloon&quot;-type help for the palette elements.</td>
</tr>
<tr>
<td>Allow Scrolling</td>
<td>This option toggles on and off. When checked (toggled on), it turns on the Allow Scrolling option.</td>
</tr>
<tr>
<td>Use Tabs</td>
<td>This option toggles on and off. When checked (toggled on), it turns on the Use Tabs option.</td>
</tr>
</tbody>
</table>
Avaya IVR Designer icons

Avaya IVR Designer icons provides a visual reference guide for icons used in IVR Designer, and to gives a brief description of each icon's functions.

As IVR Designer is a graphically oriented tool, many of its commands, tools, and features are available using icons. In IVR Designer, these icons are arranged on the following palettes, each with its own function and set of icons.

- The **Toolbar** contains icons for menus and commands found on the menu bar.
- The **Node Palette** contains the icons for all node objects.

In addition, several of the development tools and editors have their own sets of icons.

This section includes the following topics:

- Toolbar icons ................................................................. 342
- Node Palette icons .......................................................... 345
- Code Generation/Application Transfer tool icons ............... 347
- Simulation tool icons ....................................................... 347

### Toolbar icons

The toolbar icons are all related to the menus on the menu bar of the main screen. The icons are grouped according to menu.

### File Menu icons

The following table describes the **File** menu icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Related menu option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>✉️</td>
<td>New</td>
<td>Creates a new application.</td>
</tr>
<tr>
<td>📋</td>
<td>Open</td>
<td>Opens an existing application.</td>
</tr>
</tbody>
</table>
Avaya IVR Designer icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Related menu option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Closes the current application.</td>
<td></td>
</tr>
<tr>
<td>Save</td>
<td>Saves the current application to disk.</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the current application.</td>
<td></td>
</tr>
<tr>
<td>Print</td>
<td>Prints the application.</td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td>Closes any open application and exits Avaya IVR Designer.</td>
<td></td>
</tr>
</tbody>
</table>

Edit Menu icons

The following table describes the **Edit** menu icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Related menu option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Reverses (undoes) the last action or series of actions since the last Save action.</td>
<td></td>
</tr>
<tr>
<td>Redo</td>
<td>Reverses (redoes) any actions performed by the Undo option, one at a time.</td>
<td></td>
</tr>
<tr>
<td>Cut</td>
<td>Removes selected node(s) from the call flow and places them on the clipboard.</td>
<td></td>
</tr>
<tr>
<td>Copy</td>
<td>Places a copy of selected node(s) on the clipboard.</td>
<td></td>
</tr>
<tr>
<td>Paste</td>
<td>Pastes the nodes from the clipboard onto any open branch.</td>
<td></td>
</tr>
<tr>
<td>Find...</td>
<td>Searches across all call flows in the application to find:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Text strings used in component names</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Locations where specified items are used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Unused items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Text strings</td>
<td></td>
</tr>
</tbody>
</table>

View Menu icons

The following table describes the **View** menu icons.
### Global Menu Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Related menu option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Globals" /></td>
<td>Globals</td>
<td>Opens the Globals Manager and all associated resource manager.</td>
</tr>
<tr>
<td><img src="image" alt="Variables" /></td>
<td>Variables</td>
<td>Opens the Variables Manager and the Globals Manager, if it is docked there.</td>
</tr>
<tr>
<td><img src="image" alt="Phrases" /></td>
<td>Phrases</td>
<td>Opens the Phrases Manager and the Globals Manager, if it is docked there.</td>
</tr>
<tr>
<td><img src="image" alt="Prompts" /></td>
<td>Prompts</td>
<td>Opens the Prompts Manager and the Globals Manager, if it is docked there.</td>
</tr>
<tr>
<td><img src="image" alt="VXML Grammars" /></td>
<td>VXML Grammars</td>
<td>Opens the VoiceXML Grammars Manager and the Globals Manager, if it is docked there.</td>
</tr>
</tbody>
</table>

### Tools Menu Icons

The following table describes the **Tools** menu icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Related menu option</th>
<th>Action</th>
</tr>
</thead>
</table>
| ![Call Flow Inventory Tool](image) | Call Flow Inventory Tool | Opens the Call Flows window, which lets you:  
- See what call flows exist for your application  
- See where each call flow is used within the application, and what other call flows (if any) are used within each call flow  
- Locate a specific call flow within an application  
- Provide an alternate means of dragging a call flow to the Design Pad. |
| ![Code Generation/Application Transfer Tool](image) | Code Generation/Application Transfer Tool | Opens the Code Generation/Application Transfer window, which lets you:  
- Generate the code necessary to run the application on the target system  
- Transfer the generated code to the target system. |
| ![Node Inspector Tool](image) | Node Inspector Tool | Opens the Node Inspector window, which lets you:  
- Edit node attributes easily and quickly  
- Display different information and editing options for each node, depending on the node's predetermined attributes. |
| ![Preference Editor](image) | Preference Editor | Opens the Preference Editor, which lets you set global preferences for using IVR Designer. |
**Verify Design Tool**

Runs the Verify Design tool, which automatically searches through each call flow of your application to check your application design for possible errors or omissions, and displays the results in the Verification Results window.

---

**Help Menu icon**

The following table describes the **Help** menu icon.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Related menu option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Help Icon" /></td>
<td>Help Contents...</td>
<td>Opens the online Help.</td>
</tr>
</tbody>
</table>

---

**Node Palette icons**

Each Node Palette icon represents a different type of node object you can use in your application. These icons are grouped according to general node functionality.

**Control node icons**

The following table describes the **Control Node** icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Flow Icon" /></td>
<td>Call Flow node</td>
<td>Creates a new call flow or encapsulates a set of nodes.</td>
</tr>
<tr>
<td><img src="image" alt="Return Icon" /></td>
<td>Return node</td>
<td>Stops the execution of the call flow that contains it.</td>
</tr>
<tr>
<td><img src="image" alt="Set Test Icon" /></td>
<td>Set and Test node</td>
<td>Manipulates and evaluates a wide range of variable expressions and branch conditions.</td>
</tr>
<tr>
<td><img src="image" alt="Time Branch Icon" /></td>
<td>Time Branch node</td>
<td>Directs the call based on time of day, day of the week, or specific date.</td>
</tr>
<tr>
<td><img src="image" alt="Error Icon" /></td>
<td>Asynchronous Event node</td>
<td>Defines the subroutine to be executed if an asynchronous event, such as a caller hanging up or an error, is detected.</td>
</tr>
</tbody>
</table>
Telephony node icons

The following table describes the **Telephony Node** icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td>Transfer Call node</td>
<td>Transfers a call to another number.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Icon" /></td>
<td>Disconnect Call node</td>
<td>Disconnects the incoming call and executes the next node in the call flow (if one exists).</td>
</tr>
</tbody>
</table>

Miscellaneous node icons

The following table describes the **Miscellaneous Node** icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td>Inline Code node</td>
<td>Lets you to write VoiceXML code directly in your IVR Designer application.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Icon" /></td>
<td>Save &amp; Exec node</td>
<td>Calls another application on the target system or on a remote web server. The called application must be a VoiceXML application. This node is typically used in conjunction with the Load Variables node.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Icon" /></td>
<td>Load Variables node</td>
<td>Loads variable data passed to it from another IVR Designer application into the current application; typically used in conjunction with the Save &amp; Exec node and is usually the first node in the current application.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Icon" /></td>
<td>Log/Trace</td>
<td>Lets you print out the values of selected variables on the target voice response system.</td>
</tr>
<tr>
<td><img src="image5.png" alt="Icon" /></td>
<td>CallScript</td>
<td>Calls a TAS application on the target system and executes the TAS application.</td>
</tr>
</tbody>
</table>

IVR (Interactive Voice-Response) node icons

The following table describes the **IVR Node** icons.
### Code Generation/Application Transfer tool icons

The following table describes the **Code Generation/Application Transfer Tool** icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Icon/function name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>📁</td>
<td>Display Files on Remote System or Floppy</td>
<td>(Available only on the Application Transfer tab) Displays the files in the destination area before transferring your application.</td>
</tr>
<tr>
<td>🌐</td>
<td>Telnet</td>
<td>(Available only on the Application Transfer tab) Establishes a telnet connection without exiting IVR Designer.</td>
</tr>
</tbody>
</table>

### Simulation tool icons

The following table describes the **Simulation Tool** icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Icon/function name</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>⏪</td>
<td>Stop Simulation</td>
<td>Stops a simulation.</td>
</tr>
<tr>
<td>⚪</td>
<td>Pause Simulation</td>
<td>Pauses a simulation.</td>
</tr>
<tr>
<td>⏯</td>
<td>Run Simulation</td>
<td>Runs a simulation from the currently active node or from where it was stopped or paused.</td>
</tr>
<tr>
<td>☐</td>
<td>Reset Variables</td>
<td>Resets the variables to their defaults; for example, to run the simulation again.</td>
</tr>
<tr>
<td>✊</td>
<td>Step-Through Simulation</td>
<td>Steps through a simulation one node at a time.</td>
</tr>
<tr>
<td>Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Respond with Speech</td>
<td>Simulates spoken responses for the speech recognition features.</td>
<td></td>
</tr>
<tr>
<td>Generate Asynchronous Event</td>
<td>Simulates an asynchronous event, such as a customer hanging up in the middle of an application.</td>
<td></td>
</tr>
<tr>
<td>Force Timeout</td>
<td>Skips the countdown, or delay, that is programmed into many nodes, and go to the next node.</td>
<td></td>
</tr>
<tr>
<td>Dial Pulse Recognition</td>
<td>Simulates a Dial Pulse Recognition (DPR) response from a caller.</td>
<td></td>
</tr>
</tbody>
</table>
Node descriptions

Nodes are the basic building blocks used to create an application. IVR Designer offers many different types of nodes, each with its own attributes and behaviors. You can use the Node Inspector to view and edit the attributes and elements of a node.

VoiceXML nodes have the same names as TAS nodes that have similar capabilities. The most significant difference between a TAS node and a similar VoiceXML node is the node attributes and attribute values that are available from the Node Inspector.

This section includes the following topics:

Announcement node ................................................................. 350
Asynchronous Event node ......................................................... 351
Call Flow node ........................................................................ 352
CallScript node ........................................................................ 353
Disconnect Call node ............................................................... 355
Inline Code node ..................................................................... 356
Load Variables node ................................................................. 357
Log/Trace node ........................................................................ 358
Menu node ................................................................................. 359
Prompt and Collect node ......................................................... 362
Return node ............................................................................. 367
Save and Exec node ................................................................. 368
Set and Test node ..................................................................... 370
Time Branch node ................................................................. 377
Transfer Call node ................................................................. 378
Announcement node

The Announcement node plays an announcement to the caller. The following figure shows the Announcement node.

Example

ABC Automotive has a customer hot line. The caller who reaches the hot line is greeted with the following announcement:

"Thank you for calling the ABC Automotive Customer Service hot line."

Behavior

- When executed, a simple announcement is played to the caller.
- No data is collected from the caller.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Prompt</td>
<td>From the drop-down menu, select the prompt you want to use as the announcement, or double-click to create a new prompt.</td>
</tr>
<tr>
<td>Allow Interrupt</td>
<td>To allow the caller to interrupt the Initial Prompt, select True from the drop-down menu. To play the Initial Prompt without interruption, select False from the drop-down menu.</td>
</tr>
<tr>
<td>Maximum Age</td>
<td>This attribute should be used only for recorded announcements, and it indicates that the VoiceXML document is willing to use content whose age is no greater than the specified time in seconds. The document is not willing to use stale content, unless the Maxstale attribute is also specified.</td>
</tr>
<tr>
<td>Maximum Stale Time</td>
<td>This attribute should be used only for recorded announcements, and it indicates that the VoiceXML document is willing to use content that has exceeded its expiration time. If a value is specified for this attribute, then the document is willing to accept content that has exceeded its expiration time by the specified number of seconds.</td>
</tr>
<tr>
<td>Fetch Timeout</td>
<td>This attribute should be used only for recorded announcements. Type or from the drop-down menu select a variable to specify the number of seconds to wait for the content to be returned.</td>
</tr>
</tbody>
</table>
Fetch Hint  |  This attribute should be used only for recorded announcements, and it defines when content should be retrieved from the server.

To specify that a file may be downloaded when the page is loaded, select Prefetch from the drop-down menu. To specify that a file should only be downloaded when actually needed, select Safe from the drop-down menu.

---

Asynchronous Event node

The **Asynchronous Event** node defines the call flow to be executed if an asynchronous event, such as a caller hanging up or an error, is detected. The following figure shows the Asynchronous Event node.

![Asynchronous Event](image)

For more information about and practice using the Asynchronous Event node, see Using asynchronous events in VoiceXML applications on page 288.

**Example**

A caller has called the ABC Automotive Inquiry Line to get some information but hangs up before completing the call. Once the caller hangs up, the Asynchronous Event node is used to terminate the application so it is ready for a new caller.

**Behavior**

- This node specifies the call flow that is executed when an asynchronous event is detected.

- When the system detects an asynchronous event, the following sequence occurs. The system:
  1. Drops the call
  2. Jumps to the specified call flow
  3. Executes the call flow
  4. Terminates application processing of the call

**Attributes**

Use the Node Inspector to assign and edit the following node attributes.
<table>
<thead>
<tr>
<th>Group</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Name</td>
<td>From the drop-down menu, select the variable in which to store the event name.</td>
<td></td>
</tr>
<tr>
<td>Event Message</td>
<td>From the drop-down menu, select the variable in which to store the event message.</td>
<td></td>
</tr>
<tr>
<td>Event Types</td>
<td>Caller Hangup</td>
<td>From the drop-down menu, select the call flow to execute when a caller hangs up.</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>From the drop-down menu, select the call flow to execute for an error event.</td>
</tr>
</tbody>
</table>

**Call Flow node**

The **Call Flow** node is used to create a new call flow or to encapsulate a set of nodes.

To create a new call flow, drag the Call Flow node icon to any open branch of a call flow. You can also create a disconnected call flow by dragging the Call Flow node to a call flow window without dropping it onto an open branch.

You can also use the Call Flow node to encapsulate a portion of a call flow and make a new, separate call flow out of it. The new call flow can then be viewed in its own call flow window on the design pad.

You can reuse or copy the encapsulated node anywhere in the application, as if it were a single node. If you want to perform the same operation in more than one place in an application, you can use the Call Flow node like a subroutine.

You can also encapsulate a large call flow within an application. By encapsulating a large call flow, several nodes can be collapsed into one node until you want to expand it. Encapsulating nodes can also improve the readability of large applications.

To locate and edit the call flow in an encapsulated node, click the node to highlight it, then select Edit Flow from the menu. IVR Designer displays the encapsulated call flow on the design pad.

The following figure shows the Call Flow node.

![Call Flow node](image)

**Note:**
Deleting a Call Flow node does not delete the associated call flow. For
information about deleting a call flow, see Right-Click Menu options for the Call Flow Inventory tool on page 78.

For more information about and practice using the Call Flow node, see Using embedded call flows in VoiceXML applications on page 263.

**Behavior**

- When the Call Flow node is reached, the encapsulated call flow is executed.
- All branches from nodes contained in the encapsulated node return to the Default branch unless additional branches are inserted. If a branch is added to the encapsulated node, it appears as a Return node in the call flow.
- Any branch of the call flow that does not return to a specified branch within its parent call flow returns to the Default branch.

**Attributes**

When you insert a Call Flow node in a call flow, it creates one branch named Default. This branch cannot be renamed or deleted. You can also insert additional branches on this node.

The number in parentheses after the Call Flow name represents each instance of the call flow used in the application. For example, the first time you use the node in an application, (1) follows the call flow node name. The second instance is followed by a (2), and so on.

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide Execution</td>
<td>When set to True, this attribute hides the execution of the encapsulated node during Simulation.</td>
</tr>
</tbody>
</table>

**CallScript node**

The CallScript node calls a TAS application on the target system and executes the TAS application. The called TAS application should use the Getarg external function to collect the variable data that is passed to it, and the Ret2vxml external function to return variable data and control to the calling application. For practice using the CallScript node, see Running TAS applications from VoiceXML on page 292.

**Note:**

The CallScript node can be used only for VoiceXML applications that will run on a Avaya Interactive Response system.
The CallScript node cannot be used to call a VoiceXML application. For information about calling a VoiceXML application from another VoiceXML application, see Save and Exec node.

The following illustration shows the CallScript node.

### Attributes

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pass Variables</strong></td>
<td>Click the plus icon to add a new variable element to the called application. All variables defined for the current application display on this drop-down menu. Type a literal value, or from the corresponding drop-down menu, select the variable whose value you want to pass to the called application. The Name column field displays the number assigned to each variable selected. The Name column field displays how many variables have been passed to the called application. The # elements field displays the number of arguments that will be returned from the called application.</td>
</tr>
<tr>
<td><strong>Script Name</strong></td>
<td>Type a literal value, type the name of a variable that contains the name of the application, or from the drop-down menu select the variable that contains the name of the application or script to execute.</td>
</tr>
<tr>
<td><strong>Put Return Value In</strong></td>
<td>From the drop-down menu, select the variable that you want to store the return value from the called application.</td>
</tr>
<tr>
<td><strong>Put Return arguments number in</strong></td>
<td>From the drop-down menu, select the variable that you want to store the number of arguments that will be returned from the called application.</td>
</tr>
</tbody>
</table>

### Behavior

- This node calls a TAS application on the target system and executes the TAS application.
- If the target system cannot execute the TAS application, for example, if the application does not exist on the target system, the Fail branch is taken.
- If the target system cannot execute the TAS application, for example, if the application does not exist on the target system, the Fail branch is taken.
| Return Variables | Click the plus icon to add a new variable element to be returned from the application you are calling.

The # elements field displays how many variables have been selected. The Name column field displays the number assigned to each variable selected.

From the corresponding drop-down menu, select the variable whose value you want returned from the called application. All variables defined for the current application display on this drop-down menu. |

---

**Disconnect Call node**

The **Disconnect Call** node disconnects the incoming call and executes the next node in the call flow. The following figure shows the Disconnect Call node.

![DisconnectCall](image)

**Example**

A customer calling ABC Automotive gets the desired information and hangs up. The system disconnects and resets all the variables to be ready for the next caller.

**Behavior**

- Disconnects incoming calls and executes the next node
- Allows the call flow to be terminated at specific user-defined points
- Is not required to successfully terminate the dialog with the caller

**Attributes**

There are no user-configurable attributes for the Disconnect Call node.
Inline Code node

The Inline Code node allows you to write VoiceXML code directly in your IVR Designer application. When the code is generated and the application is transferred, the VoiceXML code is inserted at the point where the Inline Code node is placed in the application. The following figure shows the Inline Code node.

For more information about the Inline Code node, see Using Inline Code nodes to perform database operations on page 314.

Behavior

- The Inline Code node takes VoiceXML code as you have written it and inserts it into the generated application code.

- Since the VoiceXML code can do whatever the developer wants it to do, the behavior of this node depends on what has been written into it.

- This node allows you to paste from the clipboard into the Inline Code Editor.

- You can use single-byte or double-byte text in the Inline Code Editor.

  Note:
  Code is inserted as is into your application. No syntax checks are performed. Any errors are not detected until you attempt to run the application on the target system.

Attributes

Use the Node Inspector to assign and edit the following node attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>The Text field displays the first part of any VoiceXML code you have written. To write VoiceXML code, click the button. In the Inline Code window, enter your VoiceXML code in the Text of Code field. When you are finished, click OK. The VoiceXML code that you enter in the Inline Code window will be placed inside a VoiceXML form. Therefore, the VoiceXML code should use Form Items such as &lt;field&gt;, &lt;record&gt;, &lt;transfer&gt;, &lt;subdialog&gt;, &lt;block&gt;, and &lt;initial&gt;.</td>
</tr>
</tbody>
</table>


Node descriptions

Load Variables node

The **Load Variables** node loads variable data passed to it from another application into the current application.

The Load Variables node is typically used in conjunction with the **Save and Exec** node and is usually the first node in the current application. The Return Control attribute of the Save and Exec node should be set to True.

The following figure shows the Load Variables node.

![LoadVariables1](image)

**Example**

ABC Automotive has two developers working on the application for its Inquiry Line, which is a very large application. To make it easier for the two developers to do their work, they split the application into two parts, with each developer working on part of it as a separate application. They design it in such a way that control passes from one part to the other at a certain point in the script, and certain variable values are passed at the same time. They use the Load Variables node to receive the variable data, such as account numbers, being passed from the first application into the second.

**Behavior**

This node loads variable data from another application passed by the Save and Exec node and assigns the data to variables defined for the current application.

**Attributes**

Use the Node Inspector to assign and edit the following node attribute.
Attribute | Description
---|---
Variables | Click the plus icon to add a new variable element to receive variable data from another application.
The # elements field displays how many variables have been selected. The Name column field displays the number assigned to each variable selected.
Type the name or from the corresponding drop-down menu select the variable you want to assign the received variable data to. All variables defined for the current application are displayed on this drop-down menu.
**Note:**
If you type a variable name, it must be the name of a variable already defined for the application.

---

**Log/Trace node**

The **Log/Trace** node lets you print out the values of selected variables on the target voice response system. This is useful primarily as a debugging tool after installing your application on the target system.

**Note:**
For the Log/Trace node to be useful, logging must be turned on for the VoiceXML interpreter on the target system.

The following figure shows the Log/Trace node.

![LogTrace node](image)

**Example**

You have installed your application on the target system, and you want to make sure the application is updating a particular variable properly. You would place a Log/Trace node in the call flow both before and after the variable is supposed to be updated. The log file on the CONVERSANT or Avaya IR system displays the value of the variable both before and after it is updated.

**Behavior**

- The values of the selected variables are displayed in the log file on the target voice response system.
- You can use single-byte or double-byte text in the Text field.
Attributes

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Group</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Text</td>
<td>Use the Text attribute to enter text that you want to appear in the log for this Log/Trace node. To enter text, click the button. In the Log Node window, enter the text that you want to appear in the log for this node. When you are finished, click OK.</td>
</tr>
<tr>
<td>Variables</td>
<td>Elements</td>
<td>Click the button to add variables to trace at this point in the call flow, and then select the variable you want from the drop-down menu. The # elements field displays how many variables have been selected. The Name column field displays the number assigned to each variable selected. From the corresponding drop-down menu, select the variable whose values you want to trace. All variables defined for the current application are displayed on this drop-down menu.</td>
</tr>
</tbody>
</table>

Menu node

The Menu node presents a list of options from which the caller can choose, then moves to different parts of the application based on that choice. The following figure shows the Menu node.

![Menu node diagram]

When you insert a Menu node in a call flow, IVR Designer creates four branches and assigns each branch a default identifying number. You can rename or delete branches, or you can insert additional branches.

For more information about and practice using the Menu node, see Presenting menus in VoiceXML applications on page 224.
Example

The ABC Automotive Inquiry Line gives the caller a menu of options to choose from:

"For the Service Department, press 1.
For the Parts Department, press 2.
To find out the status of an order, press 3."

A valid response from the caller would be a touchtone of 1, 2, or 3.

Behavior

- This node plays the Initial Prompt as specified in the Node Inspector (see the Prompts tab).
- The Initial Prompt presents a predetermined menu of choices to the caller.
- The Menu node directs the call based upon the caller’s response to the menu choices.
- If the caller enters an invalid response, the Bad Input Prompt is played. If the Replay Initial Prompt field is set to True, the Initial Prompt also plays after the Bad Input Prompt.
- If the caller does not respond within the time set in the First Digit Timeout field, the Timeout Prompt is played. If the Replay Initial Prompt field is set to True, the Initial Prompt also plays.
- If the caller does not respond on the last try, the Timeout branch is taken if one has been defined. Otherwise, the Fail branch is taken by default.
- Valid responses from the caller are saved in the appropriate variable.

Inserting branches

To insert a new branch on a Menu node:

1. Right-click where you want the new branch to be inserted.

   Note:
   When the new branch is inserted, it moves the selected branch down.

   IVR Designer displays the Touchtones keypad window.

2. Do one of the following:
   - Click the digit (or digits) on the Touchtones keypad that you want to assign to label the branch.
   - To define the branch as a timeout branch, click the icon.
Node descriptions

— To define the branch using a VoiceXML grammar, click the icon.

Clicking this icon opens a Grammar Word List window. This window displays the words defined for the grammar specified in the Input ASR Mode attribute for this node.

3. Click OK.

Renaming branches

To rename (assign a different identifier to) a branch on a Menu node:

1. Right-click the branch you want to rename, then select Rename Branch from the menu. IVR Designer displays the Touchtones keypad window.

2. Do one of the following:

— Click the digit or digits you want to assign to label the branch.

— To rename the branch as a timeout branch, click the icon.

— To rename the branch using a VoiceXML grammar, click the icon.

Clicking this icon opens a Grammar Word List window. This window displays the words defined for the grammar specified in the Input ASR Mode attribute for this node.

3. Click OK.

Deleting branches

To delete a branch from a Menu node:

1. Right-click the branch you want to delete.

2. Select Delete from the menu.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Group</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompts</td>
<td></td>
<td>Initial Prompt</td>
<td>From the drop-down menu, select the prompt you want to play when the Menu node begins to execute, or double-click to create a new prompt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bad Input Prompt</td>
<td>From the drop-down menu, select the prompt to play when the caller enters invalid input, such as an invalid account number.</td>
</tr>
</tbody>
</table>
### Prompt and Collect node

The **Prompt and Collect** node is used to prompt a caller for input such as account numbers or credit card numbers. The following figure shows the Prompt and Collect node.

![PromptAndCollect.png](image)

When you insert a Prompt and Collect node in a call flow, IVR Designer creates two branches, which cannot be renamed or deleted. You can, however, insert additional branches, which can be renamed.

For more information about and practice using the Prompt and Collect node, see Collecting information in VoiceXML applications on page 233.

#### Example

A customer calls the ABC Automotive Inquiry Line to find out about the status of an order. The caller is prompted for the order number:

### Reference

<table>
<thead>
<tr>
<th><strong>Response</strong></th>
<th><strong>Input Parameters</strong></th>
<th><strong>Timeout Prompt</strong></th>
<th>From the drop-down menu, select the prompt to play when the caller waits too long to respond to the Initial Prompt.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response</strong></td>
<td><strong>Input Parameters</strong></td>
<td><strong>Allow Interrupt</strong></td>
<td>To allow the caller to interrupt while prompts are playing, select <strong>True</strong> from the drop-down menu. To play the prompts without interruption, select <strong>False</strong> from the drop-down menu.</td>
</tr>
<tr>
<td><strong>Input ASR Mode</strong></td>
<td><strong>Timeout</strong></td>
<td><strong>First Digit Timeout</strong></td>
<td>Type or from the drop-down menu select a variable to specify the number of seconds to allow a caller to respond to the Initial Prompt.</td>
</tr>
<tr>
<td><strong>Input ASR Mode</strong></td>
<td><strong>Timeout</strong></td>
<td><strong>Inter-Digit Timeout</strong></td>
<td>Type or from the drop-down menu select a variable to specify the number of seconds to allow a caller for entering each subsequent digit in a multiple-digit response.</td>
</tr>
<tr>
<td><strong>Tries</strong></td>
<td><strong>Replay Initial Prompt</strong></td>
<td>If you want the Initial Prompt to replay each time the caller enters an invalid response (after any error messages when appropriate), select True.</td>
<td></td>
</tr>
</tbody>
</table>
"The order number is located in the upper left corner of your order form, just underneath your name and address. Please enter your eight-digit order number, followed by the pound sign."

After the caller enters the requested order number and pound sign, the following prompt plays:

"You entered your order number as: 31598433. Is this correct? If so, say Yes at the tone."

**Behavior**

- The Initial Prompt plays, requesting the caller to enter input.
- The caller's response is collected for future use.
- The caller may be asked to confirm the response entered.
- The caller's input can be validated against a variable.

**Inserting branches**

To insert a new branch in a Prompt and Collect node:

1. Right-click where you want the new branch to be inserted.
2. Select **Insert Branch** from the menu.
   
   IVR Designer displays the Touchtones dialog box.
3. Do one of the following:
   
   - Click the digit (or digits) on the Touchtones keypad that you want to assign to label the branch.
   
   - To define the branch as a **timeout** branch, click the [ ] icon.
   
   - To define the branch using a VoiceXML grammar, click the [ ] icon.

   Clicking this icon opens a **Grammar Word List** window. This window displays the words defined for the grammar specified in the Input ASR Mode attribute for this node.
4. Click **OK**.

**Renaming branches**

To rename (assign a different identifier to) a branch:

1. Right-click the branch you want to rename.
   
   IVR Designer displays the Touchtones keypad window.
2. Select **Rename Branch** from the menu.

3. Do one of the following:
   - Enter the digit (or digits) you want to assign to the branch.
   - To rename the branch as a *timeout* branch, click the icon.
   - To rename the branch using a VoiceXML grammar, click the icon.

   Clicking this icon opens a **Grammar Word List** window. This window displays the words defined for the grammar specified in the Input ASR Mode attribute for this node.

4. Click **OK**.

**Prompt and Collect node attributes**

**Attributes**

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Group</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompts</td>
<td>Initial Prompt</td>
<td>From the drop-down menu, select the prompt you want to play when the Prompt and Collect node begins to execute, or double-click to create a new prompt.</td>
<td></td>
</tr>
<tr>
<td>Error Prompts</td>
<td>Bad Input Prompt</td>
<td>From the drop-down menu, select the prompt to play when the caller enters invalid input, such as an invalid account number.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Timeout Prompt</td>
<td>From the drop-down menu, select the prompt to play when the caller waits too long to respond to the Initial Prompt.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Too Few Digits Prompt</td>
<td>From the drop-down menu, select the prompt to play when the caller does not enter enough digits.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Confirm Input Prompt</td>
<td>From the drop-down menu, select the prompt to play after the caller enters a response. This prompt is used to get feedback as to whether or not the caller's initial response was correct.</td>
<td></td>
</tr>
<tr>
<td>Response</td>
<td>Input Parameters</td>
<td>Allow Interrupt</td>
<td>To allow the caller to interrupt while prompts are playing, select <strong>True</strong> from the drop-down menu. To play the prompts without interruption, select <strong>False</strong> from the drop-down menu.</td>
</tr>
<tr>
<td>Node descriptions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Length</td>
<td>Type or from the drop-down menu select a variable to specify the maximum number of digits to accept from the caller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum Length</td>
<td>Type or from the drop-down menu select a variable to specify the minimum number of digits to accept from the caller.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminate Response With</td>
<td>Type or from the drop-down menu select a variable to specify the touchtone character to be entered by the caller to indicate the end of a response.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speech Recognition Confidence Level</td>
<td>Type or from the drop-down menu select a variable to specify the Speech Recognition Confidence Level. The value you specify sets a threshold for the confidence level of the recognized result returned by the recognition engine. If the confidence level of the recognized result is below the specified value, the result is rejected (a no match event is thrown). The range is 0 through 100 percent. A value of 0 means that minimum confidence is needed to accept a recognition result. If this attribute is set to none or left blank, the default value for the target system (set in the default.xml file) is used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeouts</td>
<td>First Digit Timeout</td>
<td>Type or from the drop-down menu select a variable to specify the number of seconds to allow a caller to respond to the Initial Prompt.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inter-Digit Timeout</td>
<td>Type or from the drop-down menu select a variable to specify the number of seconds to allow a caller for entering each subsequent digit in a multiple-digit response.</td>
<td></td>
</tr>
<tr>
<td>Tries</td>
<td>Tries Available</td>
<td>Type or from the drop-down menu select a variable to specify the number of tries to allow a caller to respond to the Initial Prompt. The default value is [Tries].</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tries Taken</td>
<td>From the drop-down menu, select a variable to store the number of tries it took a caller to enter a correct response. The default value is [Attempts].</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Replay Initial Prompt</td>
<td>If you want the Initial Prompt to replay each time the caller enters an invalid response (after any error messages when appropriate), select True.</td>
<td></td>
</tr>
<tr>
<td>Caller's Response</td>
<td>Input ASR Mode</td>
<td>From the drop-down menu, select whether you want to use Touchtones (default), or a VoiceXML grammar to collect input from the caller.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASR Language</td>
<td>(Not available if Touchtone is selected as the Input ASR Mode) From the drop-down menu, select the language to use for speech recognition.</td>
<td></td>
</tr>
<tr>
<td>Maximum N-Best Values</td>
<td>Type or from the drop-down menu select a variable to specify the maximum number of N-best values to be returned by the recognition engine. This must be a number variable. The default value is 1. <strong>Notes:</strong> The value you specify for this attribute determines how many results the recognizer will return. If the Input ASR Mode attribute is set to Touchtone or if the caller responds to an ASR grammar with touchtones, only one result will be returned, no matter what value the Maximum N-Best Values attribute is set to.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of N-Best Values returned</td>
<td>From the drop-down menu select a variable to store the number of N-best values returned by the recognition engine. This must be a number variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-Best Interpretations</td>
<td>Select the plus icon to add one or more variables to store the interpretation(s) returned by the recognition engine. This must be a character variable. The default value for the first element is [CollectedDigits].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-Best Utterances</td>
<td>Select the plus icon to add one or more variables to store the response(s) (utterances) from the caller. This must be a character variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-Best Confidences</td>
<td>Select the plus icon to add one or more variables to store the confidence level(s) in the recognized result(s) returned by the recognition engine. This must be a character variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-Best Input Modes</td>
<td>Select the plus icon to add one or more variables to store the values (dtmf or voice) that indicate how the caller responded to the initial prompt. This must be a character variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confirmation</td>
<td>Confirmation ASR Parameters</td>
<td>Yes Digit</td>
<td>Enter the digit, using a variable name or a literal value, to be used to confirm the caller's response. The default value is [YesDigit]. <strong>Notes:</strong> If a prompt is specified for the Confirm Input Prompt attribute, a value must be specified for the Yes Digit attribute. If the Input ASR Mode attribute is set to Boolean, set the Yes Digit attribute to <strong>true</strong>.</td>
</tr>
</tbody>
</table>
### Return node

The **Return** node stops the execution of the call flow that contains it. The following figure shows the Return node.

![Return node](image)

**Behavior**

- In the **[Main]** call flow, this node stops the execution of the application or returns control to the calling application.

- In any other call flow, this node results in the call flow returning and taking the branch with the same name as the Return node.

**Attributes**

Use the Node Inspector to assign and edit the following node attribute:

<table>
<thead>
<tr>
<th>Node Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Digit</td>
<td>Enter the digit, using a variable name or a literal value, to be used to negate the caller's response. The default value is [NoDigit]. <strong>Note:</strong> If the Input ASR Mode attribute is set to Boolean, set the No Digit attribute to <strong>false</strong>.</td>
</tr>
<tr>
<td>Confirmation ASR Input Mode</td>
<td>(Not available if None is selected for the Confirm Input Prompt attribute) From the drop-down menu, select whether you want to use Touchtones, or a VoiceXML grammar as confirmation input from the caller. The default value is Touchtone</td>
</tr>
<tr>
<td>Confirmation ASR Language</td>
<td>(Not available if Touchtone is selected as the Confirmation ASR Input Mode) From the drop-down menu, select the language to use for confirmation input from the caller.</td>
</tr>
<tr>
<td>Confirmation Mode Chosen</td>
<td>(Not available if Touchtone is selected as the Confirmation ASR Input Mode) From the drop-down menu, select the variable that stores the value indicating how the caller responded to the Confirmation Prompt.</td>
</tr>
</tbody>
</table>
Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Value</td>
<td>Type or from the drop-down menu select the variable that holds the return value. This attribute allows the called application to return a number to the calling application. A return value can only be specified within the main call flow, which enables the main call flow to be called by another application using a Save and Exec node (see the Save and Exec node description). If a return value is not specified, the &lt;exit&gt; tag is used, which causes the call to terminate.</td>
</tr>
</tbody>
</table>

Save and Exec node

The **Save and Exec** node calls another application on the target system or on a remote web server.

To return control to the calling application, set the **Return Control** attribute to **true**. When the Return Control attribute is set to **true**, the Save and Exec node can be used to pass variable data to a Load Variables node. For information about the Load Variables node, see Load Variables node on page 357.

To submit variables to a CGI script on a remote web server, set the Return Control attribute to **false**.

The called application can be any of the following:

- A VoiceXML application created using IVR Designer that resides on the target system or on a remote web server.
- A VoiceXML document that resides on the target system or on a remote web server.
- A CGI script that resides on a remote web server.

The called application cannot be a TAS application. For information about how to call a TAS application from a VoiceXML application, see CallScript node on page 351.

The following figure shows the Save and Exec node.

![SaveAndExec node](image)

**Example**

The Save and Exec node can be used similar to a function call. You can create a set of functions that can be shared among documents in a single application, or that can be shared
among several applications. For example, functions could be created that request information from the caller such as a credit card number or an account number.

**Behavior**

- This node calls another application and executes it, passing control to the new application.
- Control can return to the current application if the Return Control attribute is set to **true**.
- If the target system cannot execute the new application, for example if the application does not exist on the target system, the Fail branch is taken.

**Attributes**

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Group</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Script Name</td>
<td>Type a literal value, type the name of a variable containing a URI, or from the drop-down menu select the variable containing the URI of the application to execute. If you type a literal value and the value has a period or a forward slash ( / ), it is assumed that it is a URI. Otherwise, it is assumed that the specified value is a script name, and it will be converted by the Code Generation tool into a URI pointing to a location on the target system. The URI generated by the Code Generation tool points to an application with the specified script name in the directory above the calling application's directory. If you type or select a variable, the variable must contain a valid URI for the application that you want to be executed.</td>
</tr>
<tr>
<td></td>
<td>Return Control</td>
<td>When the Return Control attribute is set to <strong>True</strong>, another dialog – which is a subdialog of the current one – is invoked, expecting the subdialog to return control to the calling application. Set the Return Control attribute to <strong>True</strong> to pass variable data to a Load Variables node. Setting the Return Control attribute to <strong>True</strong> enables the Put Return Value In attribute. When the Return Control attribute is set to <strong>False</strong>, the variables specified in the Variables attribute are submitted to the application specified in the Script Name attribute. The value specified in the Script Name attribute should point to a CGI script. <strong>Note:</strong> If you want to pass variable data to a Load Variables node, do not set the Return Control attribute to <strong>False</strong>.</td>
</tr>
</tbody>
</table>
### Reference

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put Return Value In</td>
<td>This attribute can be set only if the Return Control attribute is set to <code>True</code>. From the drop-down menu, select the variable that you want to store the return value.</td>
</tr>
<tr>
<td>Request Method</td>
<td>From the drop-down menu, select Get (default) or Post.</td>
</tr>
<tr>
<td>Media Encoding Type</td>
<td>Type the MIME type of the submitted document.</td>
</tr>
<tr>
<td>Resource Fetching</td>
<td></td>
</tr>
<tr>
<td>Maximum Age</td>
<td>This attribute indicates that the VoiceXML document is willing to use content whose age is no greater than the specified time in seconds. The document is not willing to use stale content, unless the Maxstale attribute is also specified.</td>
</tr>
<tr>
<td>Maximum Stale Time</td>
<td>This attribute indicates that the VoiceXML document is willing to use content that has exceeded its expiration time. If a value is specified for this attribute, then the document is willing to accept content that has exceeded its expiration time by the specified number of seconds.</td>
</tr>
<tr>
<td>Fetch Timeout</td>
<td>Type or from the drop-down menu select a variable to specify the number of seconds to wait for the content to be returned.</td>
</tr>
<tr>
<td>Fetch Hint</td>
<td>This attribute defines when content should be retrieved from the server.</td>
</tr>
<tr>
<td></td>
<td>To specify that a file may be downloaded when the page is loaded, select Prefetch from the drop-down menu. To specify that a file should only be downloaded when actually needed, select Safe from the drop-down menu.</td>
</tr>
<tr>
<td>Variables</td>
<td>Click the plus icon to add a new variable element to be passed to the application you are calling.</td>
</tr>
<tr>
<td></td>
<td>The # elements field displays how many variables have been selected. The Name column field displays the number assigned to each variable selected.</td>
</tr>
<tr>
<td></td>
<td>From the corresponding drop-down menu, select the variable whose values you want to pass to the new application. All variables defined for the current application display on this drop-down menu.</td>
</tr>
</tbody>
</table>

### Set and Test node

The **Set and Test** node allows you to manipulate and evaluate a wide range of variable expressions and branch conditions. This node has two basic uses: It allows you to assign values to variables and to test whether a variable is, for example, greater than another variable or value. The kinds of variable expressions you can set and test include the following:
Node descriptions

- Arithmetic
- Assign
- Concat
- ItemCount
- ItemOf
- Length
- Parse
- Substring

**Note:** Before you can set test conditions for any of these expressions, you must first add a branch to the node.

The following figure shows the Set and Test node.

![SetAndTest](image_url)

**Example**

ABC Automotive wants to perform a telephone customer survey. The survey consists of six questions. The company wants to track how many participants were involved in the survey. Callers must respond to at least four questions to be considered a participant.

The Set and Test node is used to set the value of a variable to the number 4. Each time the caller responds to a question, the variable is decremented. At the end of the survey, if the variable’s value is 0 or less, the caller is added to the number of participants in the survey.

**Behavior**

- The behavior of the Set and Test node depends largely on what expressions and conditions are set for the variable.
- If no logical expression turns out to be true, the Default branch is taken.
- Expressions are always executed first, in the order in which they appear on the Expressions tab. The Branch Conditions, if any, are then executed.

**Attributes**

You can insert additional branches for a Set and Test node. Use the Node Inspector to assign and edit the following attributes.
Reference

<table>
<thead>
<tr>
<th>Tab</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressions</td>
<td>Expressions</td>
<td>To add expressions, click the plus icon. To set an expression’s attributes, select an element, and then click the **button. The <strong>Set Test Expression Editor</strong> displays.</td>
</tr>
<tr>
<td>Branch Conditions (Optional)</td>
<td>Conditions</td>
<td>To add a Branch Conditions statement, click the plus icon. <strong>Note:</strong> You must add at least one branch to the Set and Test node before you can set Branch Conditions elements. If you attempt to set branch conditions for an element and there is no branch other than the default branch on the node, IVR Designer displays the following error message: Cannot create branch condition. This Node does not have any branches. To set branch conditions, click the **button for any element other than the default branch condition. (You cannot change the attributes of the default branch condition.) The <strong>Set Test Branch Condition Editor</strong> displays.</td>
</tr>
</tbody>
</table>

**Set Test Expression Editor**

The Set Test Expression Editor opens whenever you click the **button for an expression already defined in the selected Set and Test node.

From the Expression Type drop-down menu, select the type of expression you want to create and set.

**Note:**
You can create new variables in the Set Test Expression Editor in any field that uses variable values. To create a new variable, type the name for the variable enclosed in square brackets [ ] in the appropriate field. When you move to another field, IVR Designer automatically creates the new variable, places it in the Variables Manager, and assigns the variable the appropriate type and default attributes.

Many of the fields in the Expression Editor allow double-byte text. See Using double-byte data in VoiceXML applications on page 126 for a complete listing.

**Arithmetic expression**

The Arithmetic expression performs arithmetic operations on two selected operands and returns the resulting value. The following table describes the fields for the Arithmetic expression type.
### Assign expression

The Assign expression assigns the value of a variable or a literal value to another variable. The following table describes the fields for the Assign expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Variable</td>
<td>Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source Value</td>
<td>Type the literal value, or from the drop-down menu select the variable containing the value, to be assigned to the variable.</td>
</tr>
</tbody>
</table>

### Concat expression

The Concat expression concatenates two character strings into a single string. The following table describes the fields for the Concat expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Variable</td>
<td>Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source String 1</td>
<td>Type the character string, or from the drop-down menu select the variable containing the string to be assigned as the first string.</td>
</tr>
<tr>
<td>Source String 2</td>
<td>Type the character string, or from the drop-down menu select the variable containing the string to be assigned as the second string.</td>
</tr>
<tr>
<td>Maximum Length</td>
<td>Type the number of characters, or from the drop-down menu select the variable containing the number of characters, to control the maximum number of characters permitted in the concatenated string. If this field is set to 0 (zero), no length check is performed. The length of the destination variable determines the maximum length.</td>
</tr>
</tbody>
</table>

### ItemCount expression

The ItemCount expression counts the number of items in a list (a character string with list items separated by commas) and returns the total count as a number. Note that even though
double-byte characters (such as Chinese) can be included in the list, the items must be separated by a single-byte comma (English comma).

The following table describes the fields for the ItemCount expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>Type or from the drop-down menu select the name of the variable to which</td>
</tr>
<tr>
<td>Variable</td>
<td>you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source List</td>
<td>Type or from the drop-down menu select the variable containing a list of</td>
</tr>
<tr>
<td></td>
<td>items to be counted.</td>
</tr>
</tbody>
</table>

**ItemOf expression**

The ItemOf expression selects a particular item from a list (a character string with list items separated by commas) and returns its value. Note that even though double-byte characters (such as Chinese) can be included in the list, the items must be separated by a single-byte comma (English comma).

The following table describes the fields for the ItemOf expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>Type or from the drop-down menu select the name of the variable to which</td>
</tr>
<tr>
<td>Variable</td>
<td>you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source List</td>
<td>Type or from the drop-down menu select the variable containing a list of</td>
</tr>
<tr>
<td>Source List</td>
<td>items.</td>
</tr>
<tr>
<td>Item Number</td>
<td>Type a literal value or from the drop-down menu select the variable</td>
</tr>
<tr>
<td>Item Number</td>
<td>containing the item number of the item you want.</td>
</tr>
</tbody>
</table>

**Length expression**

The Length expression counts the number of characters in a string and returns the count as a number. The following table describes the fields for the Length expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>Type or from the drop-down menu select the name of the variable to which</td>
</tr>
<tr>
<td>Variable</td>
<td>you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source String</td>
<td>Type the character string, or from the drop-down menu select the variable</td>
</tr>
<tr>
<td>Source String</td>
<td>containing the string whose characters are to be counted.</td>
</tr>
</tbody>
</table>

Note:
The variable must be a number variable.
 Parse expression

The Parse expression breaks a field or string into two smaller subfields. The source field is typically structured to consist of two or more subfields separated by one or more separator characters such as commas or spaces. When this expression is used in an application, the first subfield of the source string is cut and copied to the destination and any separators are skipped until the next non-separator character is encountered. The remaining characters are then moved to the start of the source field.

**Note:**
This expression alters the original source field or string.

For example, suppose a source field contains the characters "Columbus, Ohio 43213". The separators, as defined in the Parse expression, are commas, periods, and spaces. The first time this expression is used then, the application removes "Columbus" and assigns it to the destination variable. The comma and space are stripped out, and the new source string is "Ohio 43213". The return code contains the number 8, for the number of characters that were reassigned.

The following table describes the fields for the Parse expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Variable</td>
<td>Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source String</td>
<td>Type the character string, or from the drop-down menu select the variable containing the string to be parsed.</td>
</tr>
<tr>
<td>Parsing Separator</td>
<td>Type the characters or from the drop-down menu select the variable containing the characters to be used in separating the parsed elements of the string.</td>
</tr>
</tbody>
</table>

Substring expression

The Substring expression returns a selected portion of a character string. The following table describes the fields for the Substring expression type.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Variable</td>
<td>Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.</td>
</tr>
<tr>
<td>Source Substring</td>
<td>Type the substring, or from the drop-down menu select the variable containing the substring to be returned.</td>
</tr>
<tr>
<td>Starting at Position</td>
<td>Type or from the drop-down menu select the variable containing the character position to use as the starting position for the returned substring. If the source string does not contain enough characters to reach the specified starting position, the return value is null.</td>
</tr>
</tbody>
</table>
### Maximum Length

Type the number of characters, or from the drop-down menu select the variable containing the number of characters, to use in setting the maximum number of characters to be used from the selected substring.

A zero in this field instructs the application to return all characters after the specified starting position.

**Examples:**

Assuming that the source substring value is "Columbus", then if:

- "Starting at Position" value is 2, and "Maximum Length" value is 3, the result is "lum".
- "Starting at Position" value is 2, and "Maximum Length" value is 8, the result is "lumbus".
- "Starting at Position" value is 3, and "Maximum Length" value is 0, the result is "umbus".
- "Starting at Position" value is -4, and "Maximum Length" value is 3, the result is "bus".
- "Starting at Position" value is -4, and "Maximum Length" value is 0, the result is "mbus".
- "Starting at Position" value is 9 or -9, the result is " " (null string).

### Set Test Branch Condition Editor

The **Set Test Branch Condition Editor** opens whenever you click the button for a branch condition already defined in the selected Set and Test node.

The following table describes the fields for the Set Test Branch Condition Editor fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go To Branch</td>
<td>Type or from the drop-down menu select the name of the branch for which you want to set conditions.</td>
</tr>
<tr>
<td>Variable</td>
<td>Type a variable name or from the drop-down menu select the variable you want to use as the condition test variable.</td>
</tr>
<tr>
<td>Relation</td>
<td>From the drop-down menu select the relation operator to use in testing the variable. Options include: =, &lt;&gt; , &lt;, &lt;=, &gt;, and &gt;=.</td>
</tr>
<tr>
<td>Value</td>
<td>Type a literal value or variable name, or from the drop-down menu select the variable containing the value to use in testing the variable. This field allows double-byte text for VXML applications.</td>
</tr>
<tr>
<td>Plus icon</td>
<td>Expands the Set Test Branch Condition Editor so you can add additional test conditions. For additional test conditions, the relation operator can be set to AND or OR.</td>
</tr>
<tr>
<td>Minus icon</td>
<td>Collapses the Set Test Branch Condition Editor after it has been expanded.</td>
</tr>
</tbody>
</table>
Group checkbox

When checked, contiguous conditions are grouped together for testing. Checking the Group checkbox is equivalent to using parenthesis for a math expression. If conditions are not grouped, they are evaluated from first entered (top) to last entered (bottom).

---

**Time Branch node**

The **Time Branch** node is used to direct the call based on time of day, day of the week, or specific date. The following figure shows the Time Branch node.

For practice using the Time Branch node, see Branching based on time, day, or date in a VoiceXML application on page 212.

**Example**

ABC Automotive wants to transfer callers to a sales representative during normal business hours only. If the caller calls at any time outside of these hours, an announcement is played asking the caller to call back during normal business hours.

**Behavior**

- The branch that matches the actual time of day, day of the week, or specified date is taken.
- If there are multiple branches that match the actual time, day or date, the first matching branch is taken.
- Select the branch labeled **Other** to insert branches and set time conditions.
- If there are no branches that match the actual time, day, or date, the default **(Other)** branch is taken.

**Inserting branches**

When you insert a Time Branch node in a call flow, it creates one branch named (by default) **Other**. This branch cannot be renamed or deleted.

To use a Time Branch node, you must insert additional branches to set the time intervals for which you want the node to check.

To insert a branch on a Time Branch node:
1. Right-click the branch where you want the new branch to go (the branch you click moves down when the new branch is created).

2. Click the Insert Branch option.

   IVR Designer displays the Time Branch Events dialog box.

   By clicking the Time tab, you can assign a Start Time and an End Time for the branch to use in routing calls.

   By clicking on the Day tab, you can assign a Start Day and an End Day for the branch to use in routing calls.

   By clicking on the Variable tab, you can set your own Branch Name, Start Variable, and End Variable for the branch to use in routing calls.

Attributes

There are no user-configurable attributes for the Time Branch node.

---

**Transfer Call node**

The Transfer Call node is used to transfer a call to another number. The following figure shows the Transfer Call node.

![Transfer Call node diagram]

**Example**

The ABC Automotive Inquiry Line has a menu giving the caller the option to transfer to the Service Department or the Parts Department. When the caller makes a choice by entering the proper touchtone, the call is transferred to the appropriate department.

**Behavior**

- When the Transfer Call node is executed, the call is transferred to the specified phone number.
- If the transfer is completed successfully, the Pass branch is taken.
- If the transfer is not completed within the time specified, the Fail branch is taken.
- The node can be set to play a message to the caller while the transfer is being executed.
Specific node behavior for this node depends on the type of transfer you have selected. There are two types of transfers: Blind (default) and bridging.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Group</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer Options</td>
<td>Transfer Options</td>
<td>Phone Number</td>
<td>Type the phone number using alphanumeric characters, or from the drop-down menu select the variable containing the phone number to transfer the caller to.</td>
</tr>
</tbody>
</table>
|                      |                 | Transfer Type     | From the drop-down menu select one of the following types of call transfer: Bringsing  
|                      |                 |                   |  
|                      |                 |                   |  
|                      |                 | Transfer Status   | This attribute applies only if the Transfer Type attribute is set to Bridging. From the drop-down menu, select the variable that you want to use to store the transfer status result. |
|                      |                 | Connect Timeout   | This attribute applies only if the Transfer Type attribute is set to Bridging. Type or from the drop-down menu select the variable that you want to use to store the time (in seconds) to wait before returning the "no answer" condition. |
|                      |                 | Maximum Allowed Time of Call | This attribute applies only if the Transfer Type attribute is set to Bridging. Type or from the drop-down menu select the variable that you want to use to store the maximum amount of time (in seconds) the call is allowed to last. |
|                      |                 | Duration of successful call | This attribute applies only if the Transfer Type attribute is set to Bridging. From the drop-down menu, select the variable that you want to use to store the duration (in seconds) of a successful call. |
| Caller Options       |                 | Message to Caller | From the drop-down menu, select the prompt to play to the caller before transferring the call.                                              |
|                      |                 | Allow Interrupt   | To allow the caller to interrupt while a prompt is playing, select True from the drop-down menu. To play the prompt without interruption, select False from the drop-down menu. |
| Failure Options      |                 | Failure Message   | This attribute applies only if the Transfer Type attribute is set to Bridging. From the drop-down menu select the prompt to play if the call cannot be completed successfully. |
System variable descriptions

System variables are variables that are predefined and included as part of the IVR Designer package.

System variables are displayed whenever you open the Variables Manager. They are identified by the word "True" in the System column. They can be used in much the same way as any custom variable.

System variables cannot be renamed, and some of their attributes (such as Type) cannot be edited by the user.

System variables are used mostly behind-the-scenes or as part of prompts. Behind-the-scenes use, for example, includes using a Time Branch node to direct a call through the call flow. In prompts, use of variables is typically defined using the Prompts tabs.

The following table describes the system variables included with IVR Designer. For more information about a particular variable, see the topic for that variable.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANI (Automatic Number Identification) on page 381</td>
<td>Stores the calling party ID number, if available.</td>
</tr>
<tr>
<td>Attempts on page 382</td>
<td>Stores the number of times the caller tries to respond to a prompt.</td>
</tr>
<tr>
<td>Channel on page 382</td>
<td>Channel number on which the current call is being handled.</td>
</tr>
<tr>
<td>CollectedDigits on page 382</td>
<td>Value of the digit or digits pressed or spoken by the caller in responding to a prompt.</td>
</tr>
<tr>
<td>Date on page 383</td>
<td>Current date in yyyymmdd format.</td>
</tr>
<tr>
<td>DNIS (Dialed Number Identification Service) on page 383</td>
<td>Stores the calling party ID number dialed, if available.</td>
</tr>
<tr>
<td>HOURS CLOSED on page 384</td>
<td>If the call is received during non-business hours, the HOURS_CLOSED variable is set to one by a Time Branch node. If the call is received after business hours, this variable is set to zero, which is the default value.</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>iidigits on page 384</td>
<td>Stores the result of the Information Indicator Digit service that provides information about the originating line (for example, pay phone or cellular service) of the caller.</td>
</tr>
<tr>
<td>LANGUAGE on page 385</td>
<td>A character string describing the language of the application.</td>
</tr>
<tr>
<td>NoDigit on page 385</td>
<td>Value of the number for a caller to press or speak signifying a &quot;no&quot; response.</td>
</tr>
<tr>
<td>PivotYear on page 385</td>
<td>Number used to help convert character variables to date variables; used to assign the date to the correct century.</td>
</tr>
<tr>
<td>Time on page 386</td>
<td>Current time in hh24mmss format.</td>
</tr>
<tr>
<td>TRANSFER_RESULT on page 386</td>
<td>A character code representing the result of the last transfer attempt.</td>
</tr>
<tr>
<td>Tries on page 387</td>
<td>Represents the number of times to allow a caller to attempt a response to a prompt.</td>
</tr>
<tr>
<td>Uui on page 387</td>
<td>Stores the result of the User to User Information service that provides supplementary information as part of an ISDN call set-up from a calling party.</td>
</tr>
<tr>
<td>YesDigit on page 387</td>
<td>Value of the number for a caller to press or speak signifying a &quot;yes&quot; response.</td>
</tr>
</tbody>
</table>

### ANI (Automatic Number Identification)

As a variable, ANI stores the phone number of the calling party. This variable is only available and functional if you have purchased the Adjunct/Switch Application Interface (ASAI) optional feature package.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as digits.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th><strong>Attribute</strong></th>
<th><strong>Default Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>none*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.
Attempts

As a variable, Attempts stores the number of times the caller tried to respond to a prompt.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as a number.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Spoken As</td>
<td>number*</td>
</tr>
<tr>
<td>Default</td>
<td>0*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

Channel

As a variable, Channel stores the channel number on which the current call is being handled.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as a number.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Spoken As</td>
<td>number*</td>
</tr>
<tr>
<td>Default</td>
<td>1*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

CollectedDigits

As a variable, CollectedDigits stores the value of the digit(s) pressed by the caller in responding to a prompt.

When spoken by the CONVERSANT or Avaya IR system, the collected values are spoken as digits.
Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>none*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

---

**Date**

As a variable, Date stores the date on which the call was received, in \textit{yyyyymmdd} format.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as \textit{ymd}. For example, \textit{19980102} would be spoken as "nineteen ninety eight January second" unless you change the \textit{Spoken As} field value.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Date</td>
</tr>
<tr>
<td>Spoken As</td>
<td>date: ymd*</td>
</tr>
<tr>
<td>Default</td>
<td>date the application was created*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

---

**DNIS (Dialed Number Identification Service)**

As a variable, DNIS stores the number the calling party dialed. This variable is only available and functional if you have purchased the Adjunct/Switch Application Interface (ASAI) optional feature package.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as digits.

Default attributes are listed in the following table.
### Attribute Default Value

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>none*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

## HOURS_CLOSED

If the call is received during non-business hours, the HOURS_CLOSED variable is set to one by a Time Branch node. If the call is received after business hours, this variable is set to zero, which is the default value.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Spoken As</td>
<td>number*</td>
</tr>
<tr>
<td>Default</td>
<td>0*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

## lidigits

As a variable, lidigits stores the result of the Information Indicator Digit service that provides information about the originating line (for example, pay phone or cellular service) of the caller.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as a number.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Spoken As</td>
<td>number*</td>
</tr>
<tr>
<td>Default</td>
<td>0*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.
### LANGUAGE

As a variable, LANGUAGE is a character string describing the language of the application.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>none*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

### NoDigit

As a variable, NoDigit is the value of the number for a caller to press signifying a "no" response.

When spoken by the CONVERSANT or Avaya IR system, the value is spoken as a digit.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>2*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

### PivotYear

PivotYear is used to designate a year which the CONVERSANT or Avaya IR can use to determine how to assign character variable date strings to a particular century. Values that are less than the value of PivotYear are interpreted to be in the 21st century (20xx). Values that are greater than the value of PivotYear are interpreted to be in the 20th century (19xx).

This variable is used only when converting a character variable in the `mmddyy` format to a date variable.

Default attributes are listed in the following table.
Reference

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Spoken As</td>
<td>number*</td>
</tr>
<tr>
<td>Default</td>
<td>50*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

Time

As a variable, Time stores the current time in hh24mmss format. The current time is the time (on the target system) at which the Time variable is encountered in the application.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as hhmmss. For example, 155227 would be spoken as "three fifty two P M" unless you change the Spoken As field value.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Time</td>
</tr>
<tr>
<td>Spoken As</td>
<td>time:hms*</td>
</tr>
<tr>
<td>Default</td>
<td>Time the application was created*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

TRANSFER_RESULT

As a variable, TRANSFER_RESULT is a character code representing the result of the last transfer attempt.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as digits.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
</tbody>
</table>
**System variable descriptions**

<table>
<thead>
<tr>
<th>Default</th>
<th>none*</th>
</tr>
</thead>
</table>

* Indicates a value that you can edit or reset.

---

**Tries**

As a variable, Tries represents the number of times to allow a caller to attempt a response to a prompt.

When spoken by the CONVERSANT or Avaya IR system, the collected value is spoken as a number.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Spoken As</td>
<td>number*</td>
</tr>
<tr>
<td>Default</td>
<td>3*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

---

**Uui**

As a variable, Uui stores the result of the User to User Information service that provides supplementary information as part of an ISDN call set-up from a calling party.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>none*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.

---

**YesDigit**

As a variable, YesDigit is the value of the number for a caller to press signifying a "yes" response.
When spoken by the CONVERSANT system, the value is spoken as a digit.

Default attributes are listed in the following table.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>character</td>
</tr>
<tr>
<td>Spoken As</td>
<td>none*</td>
</tr>
<tr>
<td>Default</td>
<td>1*</td>
</tr>
</tbody>
</table>

* Indicates a value that you can edit or reset.
Glossary

A

active call flow
The call flow currently showing in the main window (design pad) and available for editing. See also design pad.

ANI
See Automatic Number Identification.

announcement
A message the system plays to the caller to provide information. The caller is not asked to give a response. See prompt.

application
The automated transaction (interactions) among the caller, the voice response system, and any databases or host computers required for your business. See also application script.

application resource
See global resource.

application script
The computer program that controls the application (the transaction between the caller and the system). See also application.

application simulation
A process in which IVR Designer simulates the behavior of an application as it is expected to behave on the CONVERSANT or Avaya IR system. It is useful as a debugging tool.

application transfer
An option that transfers files interactively or directly to/from another computer using the file transfer system (FTS).

application verification
A process in which IVR Designer verifies that all the components needed by an application are complete.
arguments
Variable values that are passed from one function, method, or application to another.

ASR
Automatic Speech Recognition.

asynchronous event
An event detected by the system that disrupts the normal flow of the application. At present, the only asynchronous event recognized by CONVERSANT or Avaya IR systems is "inbound call termination," encountered when the caller hangs up.

attributes
Properties or values assigned to or characteristic of variables.

Automatic Number Identification
Automatic Number Identification (ANI) is a service that tells the recipient of a telephone call the telephone number of the person making the call. This number can be passed to computer equipment to automatically retrieve associated information about the caller.

backing up an application
Using a utility to make an archive copy of a completed application or an interim copy of an application in progress. The backup copy can be restored to the system if the online version is damaged, or if you make revisions and want to go back to the previous version.

backup
The preservation of the information in a file in a different location, so that the data is not lost in the event of hardware or system failure.

barge-in
A capability provided by WholeWord speech recognition and Natural Language Speech Recognition (NLSR) that allows callers to speak or enter their responses during the prompt and have those responses recognized.

blind transfer protocol
A protocol in which a call flow is terminated as soon as the phone number is dialed, without waiting to see if the telephone is busy or if the called party answers.

bps
Bits per second.
branches
Connections to a node, indicated in IVR Designer by lines leading down and to the right from a node.

breakpoints
Designated places within a call flow where the application temporarily stops running during a simulation. A breakpoint is designated in a call flow with the icon.

bridging
The process of connecting one telephone network connection to another. Bridging decreases the processing load on the system since an active bridge does not require speech processing, database access, host activity, and so on, for the transaction.

buttons
See icons.

Call Classification Analysis
Call Classification Analysis (CCA) is a process that enables application designers to use information available within the system to classify the disposition of originated and transferred calls.

call flow
1. A series of nodes linked together. 2. The “path” that a caller is directed through when interacting with a voice response system such as a CONVERSANT or Avaya IR system. See also transaction.

call progress tone
Standard telephony sounds that indicate the status of the call. These sounds include busy, fast busy, ringback, reorder, etc.

called party number
The number dialed by the person making a telephone call. Telephone switching equipment can use this number to selectively route an incoming call to a particular department or agent.

caller
The party who calls for a service, gets connected to the target voice response system, and interacts with it. As the target system can also make outbound calls for service, the caller can also be the person who responds to those outbound calls.

CCA
See Call Classification Analysis.
**code generation**
The process in IVR Designer of generating the code necessary to transfer and install an application to the target voice response system. See also Transaction Assembler Script (TAS) language.

**command**
An instruction or request the user issues to the system software to make the system perform a particular function. An entire command in UNIX consists of a command name and options.

**configuration**
The arrangement of the software and hardware in a computer system or network.

**contiguous nodes**
Any selected series of nodes in a call flow, in which the selected nodes are all directly adjacent to one another without any unselected nodes in between. Any unselected node that breaks up the series makes the series discontiguous. See also branches.

**control**
A component of a prompt used to alter the way a prompt is played to the caller. Controls can be used to regulate the speed at which a phrase is played, for example, or to insert a silence or pause between phrases.

**control bar**
The area at the top of the Globals Manager where tabstops are created and placed.

**custom speech**
Unique words or phrases to be used in CONVERSANT or Avaya IR system voice prompts that Avaya custom records on a per-customer basis.

**debug**
The process of locating and correcting errors in computer programs; also referred to as troubleshooting.

**default**
A particular setting or value for a variable that is assigned automatically by an operating system and remains in effect unless canceled or overridden by the operator.

**design pad**
The area in the main window that opens after you open an application in IVR Designer. It is where you do most of the work in creating call flow applications.
**dial ahead**
The ability to collect and process touch-tone inputs in sequence, even when they are received before the prompts.

**Dial Pulse Recognition**
Dial Pulse Recognition (DPR) is a feature that allows the voice response system to recognize digits dialed by callers using rotary dial telephones.

**dial through**
A capability provided by touchtone recognition that allows callers to enter responses during the prompt and have those responses recognized (similar to the Speak with Interrupt capability). See also barge-in.

**Dialed Number Identification Service**
Dialed Number Identification Service (DNIS) is a service that allows incoming calls to contain information about the telephone number for which it is destined.

**dialog box**
A secondary window in which you are asked to fill in information and/or make choices before closing it and returning to the main window.

**DIP**
See data interface process.

**directory**
A type of file used to group and organize other files or directories. Also called a folder.

**discontiguous nodes**
Two or more nodes selected and grouped together that have other unselected nodes in between. See also contiguous nodes.

**DNIS**
See Dialed Number Identification Service.

**docking**
The act of attaching a palette or window to another window so that it appears to become part of the second window. In IVR Designer, the term docking applies to both palettes and global resource managers.

**DPR**
See Dial Pulse Recognition.
Glossary

E

EBS
Enhanced basic speech.

empty branch
A node branch that has no node connected to it. See also branches.

Enhanced Basic Speech
Enhanced Basic Speech (EBS) is pre-recorded speech available from Avaya in several languages.

error message
A message indicating that something is wrong, often with a suggestion for correcting it.

event
The notification given to an application when some condition occurs that is generally not encountered in normal operation.

F

file
A collection of data treated as a basic unit for storage.

file transfer
An option that lets you transfer files interactively or directly to/from another computer.

File Transfer Protocol
File Transfer Protocol (FTP) is a standard protocol for transferring files using a modem and phone lines.

filename
Alphanumeric characters used to identify a particular file.

flash
See switch hook flash.

FlexWord speech recognition
A type of speech recognition based on subword technology that recognizes phonemes, or parts of words, in a specific language. Compare to phoneme.

folder
A type of file used to group and organize other files or directories. Also called a directory.
FTP
See File Transfer Protocol.

Full CCA
A feature package that augments the types of call dispositions that Intelligent CCA can provide. See also intelligent CCA.

G

global
See global resource.

global resource
Any of several types of components that can be used in building IVR Designer applications. Types include but are not limited to phrases, prompts, database tables, variables, host definitions, and external functions. See also shared resource.

global resource manager
Any of several windows that can be opened to display information about global resources in an application. For example, information about variables can be viewed in the Variables Manager. Right-clicking in a global resource manager brings up a popup menu which presents a different set of actions and options for each global resource manager. See also global resource.

grammar
The set of inputs that a speech recognizer can match (identify) from a caller.

graphical user interface
A user interface based on graphics (icons and pictures and menus) instead of text; uses a mouse as well as a keyboard as an input device.

group/tab
On the external functions palette, a set of external functions that are grouped together and whose icons appear grouped together.

GUI
See graphical user interface.
H

hard disk drive
A high-capacity data storage/retrieval device that is located inside a computer. A hard disk drive stores data on nonremovable high-density magnetic media based on a predetermined format for retrieval by the system at a later date.

hardware
The physical components of a computer system, such as keyboards, monitors, and media disk drives.

I

icons
A picture on a screen that represents a specific file, directory, window, option, or program.

independent software vendor
Independent software vendor (ISV) is a company that has an agreement with Avaya to develop software to work with the CONVERSANT or Avaya IR system to provide additional features required by customers.

Integrated Services Digital Network
Integrated Services Digital Network (ISDN) is a network that provides end-to-end digital connectivity to support a wide range of voice and data services.

intelligent CCA
A type of Call Classification Analysis (CCA) in which the line is monitored after dialing is complete, to determine whether a busy, reorder (fast busy), or other failure has been encountered. It also recognizes when the extension is answered or if the extension is not answered after a specified number of rings. The monitoring capabilities are dependent on the network interface circuit card and protocol used. See also call classification analysis.

Interactive Voice Response
A telecommunications system that uses prerecorded voice messages to present options to a user, typically over a telephone line.

interface
The point of interaction or communication between a computer and any other entity, such as a printer or human operator.

interrupt
The termination of voice and/or telephony functions when a particular condition occurs. For example, if an interrupt option is turned on, and a caller speaks a response before a prompt is done playing, the prompt terminates immediately, and the caller is moved to the next step in the call flow. See also barge-in.
ISDN
See Integrated Services Digital Network.

ISV
See independent software vendor.

IVR
See Interactive Voice Response.

IVR Designer
Optional software that provides a graphical interface to assist in the development of voice response applications.

K

Kbyte
kilobyte

keyboard mapping
The method used to assign values to keys on a computer keyboard. Keyboards can be mapped for a particular type of computer interface emulation or for a particular language's alphabet.

L

LAN
See local area network.

local area network
Local area network (LAN) is a data communications network in a limited geographical area. The LAN provides communications between computers and other electronic office equipment.

logging on/off
Entering or exiting the target voice response system software.

M

main topic window
The window that appears on the screen when the online help is first accessed. Most of the online help topics appear in this window.

main window
The window that appears when you first start IVR Designer.
Mbyte
megabyte

menu
Options presented to a user on a computer screen or with voice prompts.

menu bar
A permanently displayed menu spread horizontally across the top of the screen or window.

name bar
A field in a global resource manager which contains the name of the global resource for that resource manager. The name bar is distinguishable from other fields by its magenta color.

Natural Language Speech Recognition
An advanced type of speech recognition. Like WholeWord speech recognition and FlexWord speech recognition, NLSR can recognize particular words and phrases, but it can also interpret and assign meaning to those words and phrases. NLSR can also recognize natural numbers and currency amounts.

node
A graphic object considered a basic "building block" for IVR Designer call flows. Each node represents a portion of code that is to be generated in the final application.

non-modal window
A window in the Windows operating system that lets other windows to simultaneously be open and active. This lets you easily move back and forth between multiple windows in an application.

null value
An entry containing no value. A field containing a null value is normally displayed as blank and is different from a field containing a value of zero.

on-line help
Messages or information that appear on the user's screen when the F1 "function key" is pressed or when the Help menu is accessed.

open branch
See empty branch.
option
1. Something chosen or available as a choice. 2. An argument used in a command line to modify program output by modifying the execution of a command. When you do not specify any options, the command executes according to its default options.

palette
An area, usually at the top of a window, in which icons that perform a variety of functions are displayed. In IVR Designer, there are three palettes, each with its own set of icons: the Toolbar, the Node Palette, and the External Functions Palette.

parent nodes
Nodes that have one or more subordinate nodes connected to them, which appear below them in the call flow. See also subordinate nodes.

PBX
See private branch exchange.

PC
Personal computer.

peripheral device
Equipment, such as a keyboard or monitor, that is added to the basic processing device in a computer system.

phrase
In IVR Designer, recorded speech used in constructing prompts. A phrase in IVR Designer may be a single word, a group of words, or a complete sentence or group of sentences. See also prompt.

phrase list
A list of phrases contained in a phrase table in IVR Designer. Each phrase list entry consists of two parts: the phrase number and the phrase tag. These phrase lists become the basis of talkfiles on the target voice response system. See also talkfile.

phrase numbers
Numbers used to identify individual phrases. IVR Designer assigns phrase numbers automatically and consecutively when you create new phrases.

phrase table
A set of phrases grouped together as a set.
phrase tag
A string of up to 50 characters that identifies the contents of a speech phrase used by an application script.

phrase text
The text used to designate a particular phrase. The first 50 characters of the phrase text become the phrase tag. See also phrase tag.

pop-up
A window or menu that appears on the screen when a button or pop-up topic link is clicked.

pop-up menu
A menu that appears on the screen when the secondary, usually the right, mouse button is clicked. See also right-click menu.

pop-up topic link
A hyperlink in an online help system that causes a pop-up window to appear. It is usually distinguished by a dashed underline under a term or phrase.

pop-up topic window
A window that appears on the screen when a pop-up topic link is clicked. Pop-up topics usually explain or define terms.

port
A connection to a device that allows data or information to travel into and out of the device.

PPP
Point-to-Point Protocol

PRI
See Primary Rate Interface.

Primary Rate Interface
Primary Rate Interface (PRI) is an ISDN term for connections over E1 or T1 facilities that are usually treated as trunks.

Private Branch Exchange
Private Branch Exchange (PBX) is a private telephone switching system, usually located on a company's premises, which provides a central switching point for a number of individual telephones. Often found in hotels, businesses, government agencies, etc.
prompt
A message played to a caller that gives the caller a choice of selections in a menu and asks for a response. Compare to announcement.

prompt and collect
A message played to a caller that asks the caller for a response. The caller input is collected and the application progresses.

proxy server
A server external to the CONVERSANT or Avaya IR system used in a client/server configuration to perform processor-intensive functions, such as Natural Language Speech Recognition (NLSR) or text-to-speech. See also Proxy Text-to-Speech.

Proxy Text-to-Speech
The capability to do text-to-speech processing using one or more auxiliary computers that are connected to the CONVERSANT or Avaya IR system in a client/server configuration.

pull-down menu
A menu that can be accessed by clicking a menu options icon (a down arrow) to reveal the menu items.

recognizer
The part of the system that compares caller input to a grammar in order to correctly match (identify) the caller input.

recovery
The process of trying to establish or re-establish a working connection between the target voice response system and a remote host.

resource
See global resource.

Resource manager
See global resource manager.

return values
Variable values that are passed back to a function, method, or application script from another function, method, or script that has been invoked.
right-click menu
In IVR Designer, the popup menu called by clicking on a window or element with the secondary, usually the right, mouse button.

Note:
Although we recognize that some computer mouse devices may be configured so that the left mouse button is the secondary button, we have chosen in this documentation to refer to a secondary mouse click as a right-click, according to popular usage and convention.

S

script
The set of instructions for the target voice response system to follow during a transaction.

software
The set or sets of programs that instruct the computer hardware to perform a task or series of tasks.

speech file
A file containing a digitally recorded speech phrase.

speech phrase
A continuous speech segment encoded into a digital string. See also phrase.

speech recognition
The ability of a system to understand spoken input from callers.

status bar
The box at the bottom of the main window that displays help and status messages to aide in creating applications.

string
A contiguous sequence of characters treated as a unit. Strings are normally bounded by white spaces, tabs, or a character designated as a separator. A string value is a specified group of characters symbolized by a variable.

subordinate nodes
Nodes that are connected to one or more "parent" nodes, which appear above them in the call flow. See also parent nodes.

switch hook flash
A signaling technique in which the signal is originated by momentarily depressing the switch hook.
**system administrator**
The person assigned the responsibility of monitoring all computer system software processing, performing daily system operations and preventive maintenance, and troubleshooting errors as required.

**system variables**
Variables that are predefined and included as part of the IVR Designer package.

**T**

**talkfile**
An ASCII file that contains the speech phrase tags and phrase tag numbers for all the phrases of a specific application. The speech phrases are organized and stored in groups. Each talkfile can contain up to 65,535 phrases, and the speech file system can contain multiple talkfiles. See also phrase list.

**target voice response system**
The system on which the IVR Designer application is to be installed (for this release, CONVERSANT V8 and Avaya IR systems).

**TAS**
See Transaction Assembler Script language.

**TCP/IP**
Transfer Control Protocol/Internet Program.

**Text-to-Speech**
Text-to-Speech (tts or TTS) is an optional feature that lets an application play US English speech directly from ASCII text by converting that text to synthesized speech. The text can be used for prompts or for text retrieved from a database or host, and can be spoken in an application with prerecorded speech.

**trace**
The trace command that can be used to monitor the execution of a script.

**transaction**
The interactions (exchanges) between the caller and the voice response system. A transaction can involve one or more telephone network connections and voice responses from the target voice response system. It can also involve one or more of the system optional features, such as speech recognition, 3270 host interface, FAX Actions, etc.

**Transaction Assembler Script language**
Transaction Assembler Script (TAS) language is a proprietary language developed by Avaya, designed especially for use on CONVERSANT and Avaya IR systems.
troubleshooting
The process of locating and correcting errors in computer programs. This process is also referred to as debugging.

tts or TTS
See Text-to-Speech.

U

UK
United Kingdom

undocking
The act of removing a palette or window from another window so that it becomes a separate window. In IVR Designer, the term undocking applies to both palettes and global resource managers.

Uniform Resource Identifier
Uniform Resource Identifier (URI) is the generic set of all names and addresses which are short strings that refer to objects (typically on the internet). The most common kind of URI is a Uniform Resource Locator (URL).

UNIX Operating System
A multiuser, multitasking computer operating system.

URI
See Uniform Resource Identifier.

US
United States of America

V

variable
An application-specific information holder. Variables may be used to collect and hold data, act as counters, or make use of system performance information.

W

window
A rectangular area on the screen that displays its own file or message independently of the other areas of the screen.
word list
A set of words available for FlexWord recognition by an application during a prompt and collect action step.
Index

A

A single touchtone and speech recognition application • 53
About Avaya IVR Designer • 9
About Help • 24
About node branches • 63
About the Help window • 25
active call flow • 389
Advanced VoiceXML application development tours • 271, 272
ANI • 389
ANI (Automatic Number Identification) • 380, 381
announcement • 389
Announcement node • 350
Announcements • 42
application • 389
Application errors • 49
Application errors, checking for • 139
Application examples • 51
application resource • 389
application script • 389
application simulation • 389
application transfer • 389
Application Transfer window menus • 154
Application usability • 37
application verification • 389
arguments • 390
ASR • 390
Assigning a VoiceXML application to channels or numbers • 162
Assigning an application on the target system • 21, 162
Assigning URIs • 111
Assigning VoiceXML applications to channels or numbers • 162, 207
asynchronous event • 390
Asynchronous Event node • 351, 368
Attempts • 380, 382
attributes • 390
Audio recording tips • 113
Automatic Number Identification • 390
Avaya IVR Designer icons • 342

B

backing up an application • 390
backup • 390
barge-in • 390
Basic VoiceXML application development tours • 164, 165
blind transfer protocol • 390
bps • 390
branches • 391
Branching based on time, day, or date in a VoiceXML application • 211, 212, 220, 224, 256, 377
breakpoints • 391
bridging • 391
buttons • 391

C

Call Classification Analysis • 391
call flow • 391
Call Flow node • 352
Call flows, working with • 75
call progress tone • 391
called party number • 391
caller • 391
CallScript node • 353
CCA • 391
Channel • 380, 382
Checking an application for errors • 139
Checking for the SCI tools on the target system • 15
Choosing an Alternate View for the Prompt Text field • 94
code generation • 392
Code Generation/Application Transfer tool icons • 347
CollectedDigits • 380, 382
Collecting information in VoiceXML applications • 233, 362
command • 392
configuration • 392
Configuring the VoiceXML IVR Designer desktop • 17, 169, 170, 171, 173
Configuring VoiceXML interface elements • 174
contiguous nodes • 392
cert • 392
collection • 392
Creating New (Custom) Variables • 86
Creating an embedding call flow in a VoiceXML application • 263
Creating an external VoiceXML grammar • 122, 123, 127
Creating and speaking variables in VoiceXML • 251, 277
Creating custom VoiceXML grammars • 122, 304, 306
Creating a basic VoiceXML application • 18, 181
Creating a local VoiceXML grammar • 122, 127
Creating a new phrase within a prompt • 95
Creating a voice response application • 17, 18, 19, 61
Creating a VoiceXML application • 185, 193
Creating a VoiceXML phrase table • 212, 256
Creating a VoiceXML Time Branch application • 214
Creating an application using Avaya IVR Designer • 18
Creating new phrases • 105
Creating new prompts • 89
Creating phrases in VoiceXML • 104, 256, 259
custom speech • 392

D
Date • 380, 383
debug • 392
default • 392
Defining asynchronous events in VoiceXML • 288
Defining successful voice response applications • 30
Delete operation • 133
Deleting a prompt element • 102
design pad • 392
Designing a voice response application • 37
Diagram your application design • 35
dial ahead • 393
Dial Pulse Recognition • 393
dial through • 393
Dial through and barge-in • 56
Dialed Number Identification Service • 393
dialog box • 393
digit input • 45
DIP • 393
directory • 393
Disconnect Call node • 355
discontiguous nodes • 393
Displaying the Used By view • 77
Displaying the Uses view • 76
DNIS • 393
DNIS (Dialed Number Identification Service) • 380, 383
docking • 393
Docking and undocking resource managers • 82
Docking/undocking palettes • 339
DPR • 393
Dragging a call flow to create a copy • 78

E
EBS • 394
Edit Menu icons • 343
Edit menu options • 331
Editing a phrase in a prompt • 98
Editing a VoiceXML grammar • 125
Editing existing phrases • 106
Editing node attributes • 67
Editing prompts in VoiceXML • 89, 259
empty branch • 394
Encapsulating a VoiceXML call flow • 265
Enhanced Basic Speech • 394
Entering digit sequences • 46
error message • 394
event • 394
Examining the simulation script • 144

F
file • 394
File Menu icons • 342
File menu options • 330
file transfer • 394
File Transfer Protocol • 394
filename • 394
flash • 394
FlexWord speech recognition • 394
folder • 394
FTP • 395
Full CCA • 395

G
Generate the source code • 20
Generating a VoiceXML application • 20, 146
Generating and assigning VoiceXML applications • 146,
   202, 240
Generating and transferring an application • 146
Generating the VoiceXML application files • 148
Generating VoiceXML code • 202
Getting started with IVR Designer • 17
Getting the most out of Text-to-Speech • 58
global • 395
global resource • 395
global resource manager • 395
Globals, working with • 79
grammar • 395
graphical user interface • 15, 395
group/tab • 395
GUI • 395

H
Handling strings in VoiceXML applications • 272
hard disk drive • 396
hardware • 396
Help Menu icon • 345
Help menu options • 335
Help topics, printing • 25
HOURS_CLOSED • 380, 384
icons • 396
idigits • 381, 384
Importing text • 116
independent software vendor • 396
Inline Code node • 127, 356
Insert operation • 131
Inserting a Control into a prompt • 99
Inserting a new line (alternate view) • 97, 99, 102
Inserting a phrase break • 99
Inserting a variable into a prompt • 98
Inserting an existing phrase into a prompt • 97
Install the application on the target system • 21
Installing Avaya IVR Designer on your PC • 11
Integrated Services Digital Network • 396
intelligent CCA • 396
Interactive Voice Response • 396
interface • 396
Internationalization and globalization • 13, 126
interrupt • 396
System variable descriptions

Introduction to Avaya IVR Designer • 9
ISDN • 397
ISV • 397
IVR • 397
IVR (Interactive Voice-Response) node icons • 346
IVR Designer • 397

K
Kbyte • 397
keyboard mapping • 397

L
LAN • 397
LANGUAGE • 381, 385
Load Variables node • 357, 368
local area network • 397
Log/Trace node • 127, 358
logging on/off • 397

M
main topic window • 397
main window • 397
Maneuvering in the Globals Manager • 80
maxnbest • 364
Mbyte • 398
menu • 398
menu bar • 398
Menu node • 359
Menus • 43
Menus and menu options • 16, 330
Menus and menu options, descriptions • 330
Menus in the Code Generation window • 149
Miscellaneous node icons • 346
Modify a touch-tone-only application to include spoken input • 55
More about tabstops • 80, 81

N
name bar • 398
Name conflicts during paste actions • 65
Natural Language Speech Recognition • 398
Nbest • 364
N-Best • 364
Next operation • 130
node • 398
Node descriptions • 61, 67, 349
Node editing guidelines • 64
Node Inspector • 70
Node Palette icons • 345
Nodes, working with • 61
NoDigit • 381, 385
non-modal window • 398
null value • 398

O
Offer a user guide • 33
Offer caller training • 34
on-line help • 398
open branch • 398
Opening a Palette Configuration window • 337
option • 399

P
Pace the application • 48
palette • 399
Palettes • 17, 336
parent nodes • 399
Passing data between VoiceXML applications • 292, 300
PBX • 399
PC • 399
Performing calculations in VoiceXML • 241
peripheral device • 399
phrase • 399
Phrase Editor window elements • 108
phrase list • 399
phrase numbers • 399
phrase table • 399
phrase tag • 400
phrase text • 400
Phrases, creating • 105
Phrases, working with • 104
PivotYear • 381, 385
Placing a node in a call flow • 62
Plan for disabled callers • 34
Plan the application design • 18
Planning a voice response application • 18, 32
pop-up • 400
pop-up menu • 400
pop-up topic link • 400
pop-up topic window • 400
port • 400
PPP • 400
Presenting menus in VoiceXML applications • 219, 224, 359
PRI • 400
Primary Rate Interface • 400
Printing Help topics • 25
Printing reports • 24
Private Branch Exchange • 400
prompt • 401
prompt and collect • 401
Prompt and Collect node • 362
Prompt and Collect node attributes • 127, 364
Prompts • 40
Prompts, creating • 89
Prompts, Response, and Confirmation in VoiceXML • 89, 235
Prompts, working with • 89
Provide an attendant • 34
proxy server • 401
Proxy Text-to-Speech • 401
pull-down menu • 401
R
Rearranging icons on a palette • 337
recognizer • 401
recovery • 401
Reference • 329
Renaming a VoiceXML grammar • 125
Renaming custom variables • 87
Renaming prompts • 92
Reports, printing • 24
resource • 401
Resource manager • 401
Return node • 367
return values • 401
Reusing and Copying VoiceXML call flows • 268
right-click menu • 402
Right-click menu options for Globals Manager • 82
Right-Click menu options for nodes and empty node branches • 67
Right-click menu options for palettes • 340
Right-click menu options for the Call Flow Inventory tool • 78, 353
Right-click menu options for the Node Inspector • 72
Right-click menu options for the Phrases Manager • 106
Right-click menu options for the Prompt Editor • 102
Right-click menu options for the Prompts Manager • 93
Right-click menu options for the Simulation tool • 144
Right-click menu options for the Variables Manager • 88
Right-click menu options for the Verification Results window • 140
Right-click menu options for the VXML Grammars Manager • 126
Running an application simulation • 142
Running TAS applications from VoiceXML • 291, 292, 353
S
Save and Exec node • 368
script • 402
Searching for information in applications • 72
Select operation • 128
Selecting a node to view or edit attributes • 71
Selecting multiple nodes for cut, copy, and paste • 65
Separate touchtone and speech recognition applications • 52
Set and Test node • 370
Set Test Branch Condition Editor • 127, 376
Set Test Expression Editor • 127, 372
Setting application transfer options • 150
Setting or editing a variable's attributes • 87, 88
Setting user preferences • 15, 21
Setting VoiceXML preferences • 178
Simulating a VoiceXML application • 141, 200, 228
Simulating an application • 141
Simulation tool icons • 347
software • 402
Specifying parameters for a VoiceXML application • 192, 193
Specifying VoiceXML Prompt and Collect parameters • 236
Specifying VXML menu error prompts • 230
Specifying VXML menu response handling • 232
speech file • 402
speech phrase • 402
speech recognition • 402
status bar • 402
Stored Procedure • 136
string • 402
subordinate nodes • 402
switch hook flash • 402
system administrator • 403
System requirements • 10
System variable descriptions • 83, 380
system variables • 403
T
talkfile • 403
target voice response system • 403
TAS • 403
TCP/IP • 403
Telephony node icons • 346
Testing a VoiceXML application • 139, 198, 199
Testing and Troubleshooting VoiceXML applications • 326, 327
Testing Text-to-Speech applications • 59
Text-to-Speech • 403
Text-to-Speech in applications • 57
The Application Parameters tab • 22
The code generation process • 148
The Environment tab • 22
The Find Text tab • 75
The Globals Manager • 79
The graphical user interface • 15
The Named tab • 73
The New Phrase Assignments window • 95
The Node Inspector • 70
The Phrase Editor • 108
The Phrase Table Editor • 113
The Printing tab • 24
The Prompt Editor • 93
The Unused tab • 74
The Uses Of tab • 74
The Variables Manager window elements • 84, 127
Time • 381, 386
Time Branch node • 377
Toolbar icons • 169, 342
Tools Menu icons • 344
Tools menu options • 334
Touch-tone and speech recognition • 50
Touchtone input used with spoken input • 52
Tours, VoiceXML application development • 164
trace • 403
Tracing a VoiceXML application • 327
transaction • 403
Transaction Assembler Script language • 403
Transfer Call node • 378
Transfer the application to the target system • 21
TRANSFER_RESULT • 381, 386
Transferring an application to the target system • 159
Transferring application source files • 154
Transferring files from floppy disks to the target system • 15, 153
Transferring files using a modem • 151, 155
Transferring the application • 21, 150
Transferring VoiceXML applications • 203, 204
Tries • 381, 387
troubleshooting • 404
tts or TTS • 404
UK • 404
undocking • 404
Uniform Resource Identifier • 109, 111, 404
UNIX Operating System • 404
Update operation • 134
URI • 404
US • 404
Use flowcharts • 35
Use human factors and usability engineering resources • 32
Use outlines • 36
Use prerecorded speech • 33
Use touch-tone input when speech recognition fails • 54
Using asynchronous events in VoiceXML applications • 287, 288, 351
Using both Text-to-Speech and prerecorded speech prompts and announcements • 58
Using calculations in VoiceXML applications • 240, 273
Using database operations in VoiceXML applications • 128, 313
Using double-byte data • 14, 126, 372
Using drag-and-drop to cut, copy, and paste • 66
Using embedded call flows in VoiceXML applications • 76, 262, 263, 353
Using https
// in URIs • 112
Using Inline Code nodes to execute stored procedures • 322
Using Inline Code nodes to perform database operations • 313, 314, 356
Using Inline Node to perform operations using VoiceXML 2.1 tags • 321, 322
Using menus in VXML applications • 224
Using Prompt and Collect in VoiceXML applications • 234
Using stored procedures in VoiceXML applications • 128, 321, 322
Using Text-to-Speech for prompts and announcements • 57
Using the Allow Scrolling option • 338
Using the Call Script node • 292, 297
Using the Create Default Phrases dialog box • 120
Using the Getarg and Ret2vxml nodes • 292, 294
Using the Phrase Editor • 110
Using the Phrase Text Import dialog box • 117

Using the Save and Exec node and Load Variables node in VoiceXML • 300
Using the Tabs option • 339
Using the VoiceXML Concat and ItemOf expressions • 279
Using the VoiceXML ItemCount and ItemOf expressions • 276
Using the VoiceXML Length expression • 273
Using the VoiceXML Parse expression • 286
Using the VoiceXML Substring expression • 283
Using variables and speech in VoiceXML applications • 83, 237, 247
Using voice input in VoiceXML applications • 122, 304
Using VoiceXML 2.1 tags in VoiceXML applications • 321
Using VoiceXML grammars for voice input • 306
Uui • 381, 387

V
variable • 404
Variable formats in VoiceXML • 248
Variables, descriptions • 380
Variables, working with • 83
Verifying a VoiceXML application • 139, 199, 228
Verifying and testing an application • 139
View Menu icons • 343
View menu options • 333
Voice application basics • 28
Voice response application uses • 28
Voice response system versus an attendant • 30
VoiceXML application design overview • 181
VoiceXML application development tours • 164
VoiceXML database operations • 128
VoiceXML Globals • 79, 170, 171
VoiceXML IVR Designer Interface • 15, 165, 166
VoiceXML menus • 166
VoiceXML Node Palette • 169
VoiceXML Substring and Parse comparison • 282
VoiceXML Toolbar • 168, 169
VXML menu error prompts and response handling • 230

W
window • 404
Windows 2000 PPP configuration • 155, 156
word list • 405
Working with call flows • 75
Working with nodes • 19, 61
Working with phrases • 104
Working with prompts • 89
Working with variables • 83
Working with VoiceXML grammars • 121
Writing node notes • 71

Y
Yes/no questions • 47
YesDigit • 381, 387