



Avaya IVR Designer

December 2008

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document might be incorporated in future releases.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all the time and we have no control over the availability of the linked pages.

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 Web site: <http://www.avaya.com/support>

Licenses

USE OR INSTALLATION OF THE PRODUCT INDICATES THE END USER'S ACCEPTANCE OF THE TERMS SET FORTH HEREIN AND THE GENERAL LICENSE TERMS AVAILABLE ON THE AVAYA WEB SITE <http://support.avaya.com/LicenseInfo/> ("GENERAL LICENSE TERMS"). IF YOU DO NOT WISH TO BE BOUND BY THESE TERMS, YOU MUST RETURN THE PRODUCT(S) TO THE POINT OF PURCHASE WITHIN TEN (10) DAYS OF DELIVERY FOR A REFUND OR CREDIT.

Avaya grants End User a license within the scope of the license types described below. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the Documentation or other materials available to End User. "Designated Processor" means a single stand-alone computing device. "Server" means a Designated Processor that hosts a software application to be accessed by multiple users. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone Products or pre-installed on Hardware. "Hardware" means the standard hardware Products, originally sold by Avaya and ultimately utilized by End User.

License types

Concurrent User License (CU). End User may install and use the Software on multiple Designated Processors or one or more Servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A "Unit" means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Product that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

Shrinkwrap License (SR). With respect to Software that contains elements provided by third party suppliers, End User may install and use the Software in accordance with the terms and conditions of the

applicable license agreements, such as "shrinkwrap" or "clickwrap" license accompanying or applicable to the Software ("Shrinkwrap License"). The text of the Shrinkwrap License will be available from Avaya upon End User's request (see "Third-party Components" for more information).

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Third-party components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: <http://support.avaya.com/ThirdPartyLicense/>

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 is not working on your company's behalf). Be aware that there can 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 Support Web site: <http://support.avaya.com>

Suspected security vulnerabilities with Avaya Products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

Avaya, the Avaya logo, Avaya Interactive Response, Avaya IVR Designer, Avaya Communication Manager, CONVERSANT, and Avaya Dialog Designer are either registered trademarks or trademarks of Avaya Inc. in the United States of America and/or other jurisdictions.

All other trademarks are the property of their respective owners.

Downloading documents

For the most current versions of documentation, see the Avaya Support Web site: <http://www.avaya.com/support>

Contact Avaya Support

Avaya Inc. provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: <http://www.avaya.com/support>

Contents

Chapter 1: What's New in IVR Designer 6.1	11
MrcpSetParam external function	11
TAS2VXI node enhancements	12
Privacy property for TAS applications	12
Chapter 2: About IVR Designer	13
About Avaya IVR Designer	13
Introduction to Avaya IVR Designer	13
System requirements	15
Installing Avaya IVR Designer on your PC	16
Installing the SCI tools on the target system	18
Feature Compatibility Warnings	19
Internationalization and globalization	22
The graphical user interface	23
Getting started with Avaya IVR Designer	25
Converting Voice@Work and IVR Designer applications	25
Creating an application using Avaya IVR Designer	26
Plan the application design	26
Create the call flows and other application components	26
Generate the source code	29
Transfer the application to the target system	29
Install the application on the target system	30
Setting user preferences	30
The Application Parameters tab	31
The Environment tab	33
The Printing tab	35
About Help	35
About the Help window	36
Chapter 3: Voice application basics	39
Voice response application uses	39
Voice response system versus an attendant	41
Defining successful voice response applications	41
Chapter 4: Planning a voice response application	43
Use human factors and usability engineering resources	43
Use prerecorded speech	43
Offer a user guide	44
Offer caller training	44
Provide an attendant	45
Plan for disabled callers	45
Diagram your application design	45
Use flowcharts	46
Use outlines	46
Chapter 5: Designing a voice response application	49
Application usability	49
Prompts	52
Announcements	53

Menus.....	54
Digit input.....	56
Entering digit sequences.....	56
Validate a digit sequence entry.....	57
Confirming digit entries with callers.....	58
Yes/no questions.....	59
Pace the application.....	59
Application errors.....	60
Touch-tone and speech recognition.....	62
Application examples.....	62
Use touch-tone input when speech recognition fails.....	63
Modify a touch-tone-only application to include spoken input.....	64
Dial through and barge-in.....	65
Bilingual, multilingual, and non-US English applications.....	66
Bilingual applications.....	66
Multilingual applications.....	67
Text-to-Speech applications.....	67
Using Text-to-Speech for prompts and announcements.....	67
Using both Text-to-Speech and prerecorded speech prompts and announcements.....	68
Getting the most out of Text-to-Speech.....	68
Testing Text-to-Speech applications.....	69
Chapter 6: Creating a voice response application.....	71
Working with nodes.....	71
Placing a node in a call flow.....	71
About node branches.....	72
Node editing guidelines.....	73
Name conflicts during paste actions.....	75
Selecting multiple nodes for cut, copy, and paste.....	75
Using drag-and-drop to cut, copy, and paste.....	76
Editing node attributes.....	77
Right-click menu options for nodes and empty node branches.....	77
The Node Inspector.....	82
Selecting a node to view or edit attributes.....	83
Writing node notes.....	83
Right-click menu options for the Node Inspector.....	84
Searching for information in applications.....	85
The Named tab.....	85
The Uses Of tab.....	86
The Unused tab.....	87
The Find Text tab.....	87
Touchtone input used with spoken input.....	88
Separate touchtone and speech recognition applications.....	88
A single touchtone and speech recognition application.....	89
Working with call flows.....	90
Displaying the Uses view.....	91
Displaying the Used By view.....	92
Dragging a call flow to create a copy.....	93
Right-click menu options for the Call Flow Inventory tool.....	93
The Globals Manager.....	94
Maneuvering in the Globals Manager.....	96
More about tabstops.....	96
Docking and undocking resource managers.....	97
Right-click menu options for the Globals Manager.....	98

DPR Barge-in.....	99
Enabling DPR Barge-in.....	99
Working with variables.....	99
The Variables Manager window elements.....	101
Creating new (custom) variables.....	116
Renaming custom variables.....	117
Setting or editing a variable's attributes.....	118
Right-click menu options for the Variables Manager.....	118
Variables and arithmetic operations.....	119
Converting variable values.....	121
Variable comparisons.....	128
Working with speech recognition.....	129
Importing FlexWord grammars (word lists).....	131
Creating a FlexWord list.....	131
Creating a Natural Language Speech Recognition (NLSR) word list.....	132
Creating an NLSR word list for a TDD application.....	133
Creating stub grammars.....	133
Renaming a Word List.....	134
Editing a word list.....	134
Right-click menu options for the FlexWords Manager.....	135
Working with external functions.....	136
Importing external functions.....	137
Creating new external functions.....	138
Renaming an external function.....	139
Editing an external function.....	140
Right-click menu options for the External Functions Manager.....	140
The External Function Editor.....	141
Setting the external function parameters.....	143
Setting the function arguments attributes.....	144
Setting the Function Return Values attributes (optional).....	145
Defining the external function's appearance.....	146
Right-click menu options for the External Function Editor.....	147
The External Function Source Editor.....	148
Right-click menu options for the External Function Source Editor.....	149
Working with prompts.....	150
Creating new prompts.....	151
Renaming prompts.....	153
Right-click menu options for the Prompts Manager.....	154
The Prompt Editor.....	155
Working with phrases.....	169
Importing phrase tables.....	171
Creating new phrases.....	172
Editing existing phrases.....	172
Right-click menu Options for the Phrases Manager.....	173
The Phrase Editor.....	175
The Phrase Table Editor.....	179
Working with hosts.....	185
Importing Host Definitions.....	186
Creating new Host Definitions.....	187
Renaming a Host Definition.....	188
Editing a Host Definition.....	189
Right-click menu options for the Hosts Manager.....	189
The Host Definition Editor.....	190

Host Flows.....	201
Capturing host screens.....	203
Working with database tables.....	209
Working with database tables.....	210
Creating database tables.....	211
Making backup copies of database tables.....	212
Renaming database tables.....	213
Editing a table's schema.....	213
Editing a table's data.....	213
Right-click menu options for the Tables Manager.....	214
The Table Schema Editor.....	215
The Table Editor.....	223
Working with application resources.....	226
About sharing application resources.....	227
Guidelines for sharing resources.....	228
Importing additional resources.....	229
Viewing shared resources.....	232
Sharing-out an application resource.....	233
Updating shared-out resources.....	233
Creating a voice response application.....	234
Refreshing shared-in resources.....	238
Right-click menu options for the Application Resources tool.....	238
Chapter 7: Verifying and testing an application.....	241
Checking an application for errors.....	241
Right-click menu options for the Verification Results window.....	242
Simulating an application.....	243
Running an application simulation.....	245
Examining the simulation script.....	246
Right-click menu options for the Simulation tool.....	247
Chapter 8: Generating and transferring an application.....	249
Generating TAS instruction code.....	249
Setting Code Generation options.....	250
Generating the source code files.....	252
Code generation file types.....	253
The code generation process.....	255
Menus in the Code Generation window.....	256
Adjusting the volume of generated audio files.....	258
Database Table Provisioning window.....	259
Transferring the application.....	261
Setting application transfer options.....	261
Transferring an application to the target system.....	265
Transferring files from floppy disks to the target system.....	267
Transferring application source files.....	267
Application Transfer window menus.....	268
Transferring files using a modem.....	269
Windows 2000 PPP configuration.....	271
Chapter 9: Converting Script Builder applications.....	275
Prepare application for conversion.....	275
Convert an application.....	278
Resolving conversion problems.....	279
Simulation halted.....	280

Remote database tables do not appear.....	282
SCI variables not found.....	283
Variables are not spoken correctly.....	284
Discard Previous Touchtones always set to True.....	284
Seasonal greetings do not work.....	285
New database tables are not the most current.....	285
Time Branch nodes show redundant date branches.....	285
Custom phrase table names.....	286
Host definition issues.....	286
Breaking a long application into sections.....	288
Chapter 10: Installing an application on the target system.....	289
Preparing to install an application.....	290
Installing the application.....	290
Using the Application Install tab.....	290
Using the sci command.....	292
Assigning the installed application using the Application Install tab.....	293
Removing an application.....	294
Importing or removing a professionally recorded speech file.....	295
Transferring an application from floppy disks.....	296
Chapter 11: TAS application development tours overview.....	297
Basic TAS application development tours.....	297
TAS IVR Designer interface.....	298
Configuring the TAS IVR Designer desktop.....	307
Creating a basic TAS application.....	312
Testing a TAS application.....	330
Generating code and transferring and installing a TAS application.....	333
Sharing TAS resources.....	342
Branching based on time, day, or date in TAS applications.....	348
Copying TAS nodes and applications.....	355
Presenting menus to callers in TAS applications.....	359
Collecting information in TAS applications.....	368
Using calculations in TAS applications.....	379
Using variables and speech in TAS applications.....	385
Using embedded call flows in TAS applications.....	400
Creating database applications.....	408
Advanced TAS application development tours.....	436
Handling strings in TAS applications.....	436
Using asynchronous events in TAS applications.....	451
Using external functions in TAS applications.....	456
Passing data between TAS applications.....	480
Using stored procedures in TAS applications.....	483
Capturing voice from callers in TAS applications.....	487
Creating host applications.....	496
Avaya Interaction Center integration.....	508
Predictive Dialing System (PDS) integration.....	514
Testing and troubleshooting TAS applications.....	521
Securing user data.....	526
Chapter 12: Reference.....	529
Menus and menu options.....	529
File menu options.....	529
Edit menu options.....	531

View menu options.....	533
Window menu options.....	534
Tools menu options.....	535
Sharing menu options.....	537
Help menu options.....	537
Avaya IVR Designer icons.....	538
Toolbar icons.....	538
Node Palette icons.....	545
Control node icons.....	545
Telephony Node icons.....	546
Miscellaneous Node icons.....	547
IVR (Interactive Voice-Response) Node icons.....	548
Database Node icons.....	549
Host Node icons.....	549
External Function palette icons.....	550
Miscellaneous icons.....	550
Code Generation/Application Transfer tool icons.....	551
Events String Editor icons.....	552
Simulation tool icons.....	552
Palettes.....	554
Opening a Palette Configuration window.....	554
Rearranging icons on a palette.....	555
Using the Allow Scrolling option.....	556
Using the Tabs option.....	556
Docking/undocking palettes.....	557
Right-click menu options for palettes.....	558
Node descriptions.....	559
Add Record node.....	559
Announcement node.....	561
Answer Call node.....	562
Asynchronous Event node.....	562
Automenu node.....	563
Call Flow node.....	570
Delete Record node.....	573
Disconnect Call node.....	574
Host Connection node.....	574
Host Send node.....	576
Host Send/Receive node.....	577
Inline Code node.....	579
Load Variables node.....	580
Menu node.....	581
Prompt and Collect node.....	588
Query Table node.....	598
Return node.....	601
Save and Exec node.....	601
Set and Test node.....	603
TAS2VXI node.....	612
Time Branch node.....	614
Trace Variables node.....	615
Transfer Call node.....	617
Update Table node.....	621
Voice Capture node.....	623
System variable descriptions.....	626

ANI (Automatic Number Identification).....	629
Attempts.....	630
Channel.....	630
CI_MODE.....	631
CI_NO_DIGS_GOT.....	632
CI_RECOG.....	632
CollectedDigits.....	633
Date.....	634
Default.....	634
DNIS (Dialed Number Identification Service).....	635
Errno.....	635
F_HOST_ERROR.....	636
F_HOST_SCREEN.....	637
FINDBEST.....	637
HOURS_CLOSED.....	638
LANGUAGE.....	638
NoDigit.....	639
PivotYear.....	640
RecordCounter.....	641
RecordsFound.....	641
Time.....	642
TRANSFER_RESULT.....	642
Tries.....	643
UNIX_TIME.....	644
YesDigit.....	644
Standard external functions.....	645
CLEO external functions.....	658
ASAI external functions.....	658
Control external functions.....	665
CRM Central external functions.....	671
CTI DIP external functions.....	671
Date/Time external functions.....	681
Dialer external functions.....	682
FAX external functions.....	696
Form Filler external functions.....	699
UCID external functions.....	699
GeoTel external functions.....	700
Interaction Center external functions.....	700
NICE external functions.....	714
Phrase external functions.....	724
PRI external functions.....	725
NLSR external functions.....	726
Speech Features external functions.....	729
Telephony external functions.....	732
Vonetix external functions.....	733
Chapter 13: Glossary.....	737
Index.....	753

Chapter 1: What's New in IVR Designer 6.1

This section lists what is new in the Avaya IVR Designer 6.1 release.

- [MrcpSetParam external function](#) on page 11
- [TAS2VXI node enhancements](#) on page 12
- [Privacy property for TAS applications](#) on page 12

MrcpSetParam external function

MrcpSetParam is an NLSR external function that represents a node when imported to IVR Designer. It provides a comprehensive interface to set the following MRCP parameters on the speech server at the start of each recognition request:

- confidencelevel
- sensitivity
- speedvsaccuracy
- completetimeout
- incompletetimeout
- maxspeechtimeout
- maxnbest
- savewaveform
- recognitionmode

The timeout values for completetimeout, incompletetimeout, and maxspeechtimeout are measured in milliseconds.

The valid range for confidencelevel, sensitivity, and speedvsaccuracy is from 0 to 1.0. If the parameter value is set to -1, then this value is not set on the speech server. However, the default value available on the speech server comes into effect.

You can either use an MrcpSetParam external function once or the LreqPROXY external function nine times for each of the MRCP parameters. Avaya recommends you to use MrcpSetParam external function as using LreqPROXY external function nine times can be a time-consuming task.



Caution:

Do not use the `MrcpSetParam` external function and the `RequestPROXY` external function in the same application. The application does not compile correctly and generates the `multiple define` error message.

For more information, see [NLSR external functions](#) on page 726.

TAS2VXI node enhancements

In the TAS2VXI Node Inspector dialog box, you can now:

- Enter URL to access the called VXML application or select the character variable representing the URL from the **URL** drop-down list.
- Select a variable that contains the parameter name to be passed from the **Input Variables** drop-down list.
- Select an input value variable that contains the parameter value to be passed from the **Input Value Variables** drop-down list.
- Select a variable that stores the value returned from the called VXML application from the **Output Variables** drop-down list.

For more information, see [TAS2VXI node attributes](#) on page 612.

Privacy property for TAS applications

You can use the privacy property feature to show or hide personal details such as passwords or Personal Identification Numbers (PIN) and protect them from being revealed through the application flow traces.

When you disable the privacy property value, you can view the user input data in the IVR-Designer application traces.

You can use the following commands in the **Inline Code** node:

- To turn on privacy property:

```
#include <irParams.h>
setIRAPIparam(IRP_INPUT_DISPLAY, imm.0)
setIRAPIparam(IRP_PRIVACY_ON, imm.1)
```

- To turn off privacy property:

```
#include <irParams.h>
setIRAPIparam(IRP_INPUT_DISPLAY, imm.1)
setIRAPIparam(IRP_PRIVACY_ON, imm.0)
```

For more information, see [Securing user data](#) on page 526.

Chapter 2: About IVR Designer

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)
- 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
- An Inline Code node that lets you insert TAS code into your application "as is"
- The ability to share resources with other developers
- The ability to specify the use of more than one local or remote databases in applications
- Comprehensive language support
- Powerful Code Generation and Application Transfer tools
- External functions for Universal Call Identification (UCID), Interaction Center, Avaya Fax, and many other functions are included as part of the standard external functions set
- An extensive online Help system

Related topics:

[System requirements](#) on page 15

[Installing Avaya IVR Designer on your PC](#) on page 16

[Installing the SCI tools on the target system](#) on page 18

[Feature Compatibility Warnings](#) on page 19

[Internationalization and globalization](#) on page 22

[The graphical user interface](#) on page 23

[Getting started with Avaya IVR Designer](#) on page 25

[Converting Voice@Work and IVR Designer applications](#) on page 25

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
- 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.

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.

Option	Description
Typical	Installs all required IVR Designer files, all supported languages, and sample applications. Recommended for most users.
Compact	Installs basic IVR Designer files, but installs only the US English-female language package. Other languages must be installed separately.
Custom	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.

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.

Installing the SCI tools on the target system

The Service Creation Installation (SCI) tools are used to install TAS applications created using IVR Designer on a CONVERSANT® Version 7, Version 8, or Release 9 system, or on a Avaya Interactive Response system. For details about using the SCI tools, see [Installing an application on the target system](#) on page 289.

Checking for the SCI tools on a CONVERSANT system

If the SCI tools are not present on the CONVERSANT system, you must install them before you can install your IVR Designer applications.

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

If the SCI tools are installed on the CONVERSANT, the system displays the following message:

```
intuity sc INTUITY Service Creation Integration Package [Release number]
```

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

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

Checking for the SCI tools on a Avaya IR system

If the SCI tools are not present on the Avaya IR system, you must install them before you can install your IVR Designer applications.

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
```

Installing the SCI tools

To install the SCI tools on a CONVERSANT or Avaya IR system:

1. At the UNIX prompt, enter `pkgadd -d diskette1`
The system displays the following message:

```
Insert diskette into Floppy Drive 1.  
Type [go] when ready, or [q] to quit: (default: go)
```
2. Insert the IVR Designer floppy diskette labeled "Intuity Service Creation Integration Package".
3. Press Enter.
The system displays the following message:

```
Installation in progress. Do not remove the diskette.
```

The following packages are available:

1 sc INTUITY Service Creation Integration Package

Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]:

4. Press Enter.

After a series of information messages, the system displays the following message:

Installation of INTUITY Service Creation Integration Package (sc) was successful.

Insert diskette into Floppy Drive 1.

Type [go] when ready, or [q] to quit: (default: go) q

5. Enter q

6. Remove the floppy from the drive.

Feature Compatibility Warnings

A feature compatibility warning tells you that the feature or function you have selected is not supported on one of the target voice response platforms. To continue using the selected feature or function, click the Continue button. To stop using the feature or function, click the Cancel button.

The following table shows the IVR Designer features and functions that are not currently supported on the Avaya Interactive Response platform.

Feature or Function	Description
Voice recording rates	The following voice recording rates are not currently supported on Avaya Interactive Response : • CELP • ADPCM16 • ADPCM32 • PCM64
Text-to-Speech	Only Proxy Text-to-Speech is supported on Avaya Interactive Response. The following Text-to-Speech (TTS) controls are not currently supported on Avaya Interactive Response: • tts • tts slow • tts fast

	• tts normal • tts address • tts fraction • tts proper name • tts phone number
FlexWord voice recognition	Not currently supported on Avaya Interactive Response.
Dial Pulse Recognition (DPR)	Not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.
LASR	Lucent Automatic Speech Recognition (LASR), which is one of the speech recognition resources for Natural Language Speech Recognition (NLSR), is not currently supported on Avaya Interactive Response.
External functions	The following external functions are not currently supported on Avaya IR Release 1.0 through Release 1.2, but are supported on Avaya IR Release 1.2.1, and on CONVERSANT R9 and earlier releases. NLSR • GetdcPROXY • RmdcPROXY

	DPR DPR_Disable The following external function is supported on Avaya IR Release 1.2.1 and later releases. DPR DPR_Reset The following external functions are not currently supported on Avaya Interactive Response : Fax • FAX_CovrPage • FAX_CNG • BT_FAX_Get • BT_FAX_CovrPage • BT_FAX_Current • BT_FAX_Combine • BT_FAX_CNG • BT_Concat5 • BT_Exec_UNIX • BT_FAX_Send NLSR • NLI_OpenSes • NLI_GetToken • NLI_GetVar • NLI_CloseSes Phrase • Background • Background_Tag GeoTel IcrTrans Telephony • Transfera • Reconnect

	PRI ISDN_billing PTTS Proxytts_on
Transfer Call node	The Answer Message attribute is not currently supported on Avaya Interactive Response. The Hold option is not supported for the Answer Connection attribute, and the Hold and Complete options are not supported for the Busy Connection, Error Connection, and No Answer connection attributes.

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.

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 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
- 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 30.

The graphical user interface

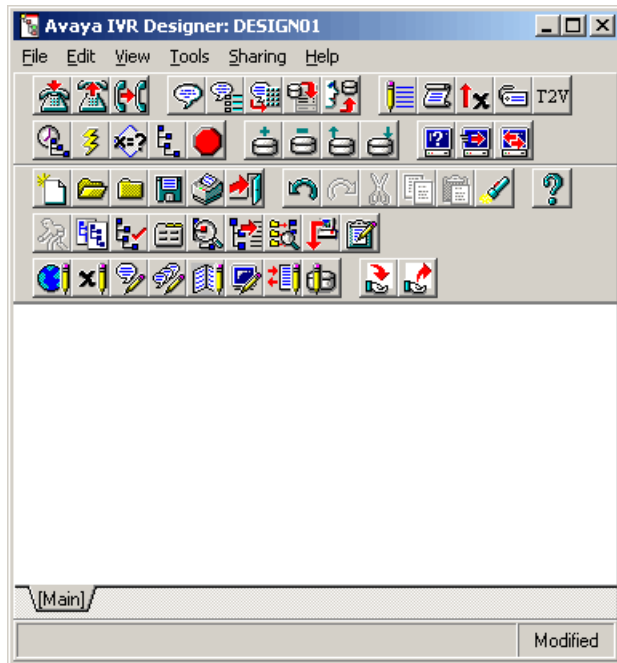
The graphical user interface (GUI) is, for the most part, based upon standard Windows conventions and practices. This includes the use of clickable icons or buttons, keyboard shortcuts, and right-click menus. For more information about the user interface, see [TAS IVR Designer interface](#) on page 298.

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 capability to create new external functions groups and tabs
- 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](#).

Just below the menu bar are the palettes. There are three palettes, each with its own set of icons. These palettes are:

- The Toolbar
- The Node Palette
- The External Function 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 554.

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](#).

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

Getting started with Avaya 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 TAS IVR Designer desktop](#) on page 307.

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.
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. Install the application on the target system.

For more information about creating an application, see [Creating a voice response application](#).

For practice creating an application, see [Creating a basic TAS application](#) on page 312.

Related topics:

[Plan the application design](#) on page 26

[Create the call flows and other application components](#) on page 26

[Generate the source code](#) on page 29

[Transfer the application to the target system](#) on page 29

[Install the application on the target system](#) on page 30

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](#).

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 (for the procedure, see [Working with nodes](#) on page 71). 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](#).

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.

Resource type	Description
Phrases and phrase tables	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.
Variables	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.
Prompts	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.
Database tables	IVR Designer applications can be designed to make use of database tables that reside on the target system or on machines other than the target system. These tables be created and populated from within IVR Designer.
External functions	External functions are used when you need IVR Designer to do something that no currently existing node does.
Host definitions	Host definitions are used to specify an interface between the target system and a host computer that is remote to the target system.
FlexWord grammars	FlexWord grammars are based on a type of speech recognition that

	recognizes phonemes (parts of spoken words) in a specific language.

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
Application Resources	The Application Resources tool lets you: • See at a glance what resources are in use with the current application • Import additional resources into your application as needed (as Private Copies or Shared) • View which resources have been shared in or shared out (Sharing tab)
Call Flow Inventory	The Call Flow Inventory tool lets you see: • What call flows exist for your application • Which call flows use other subordinate call flows (Uses tab) • Which call flows are used by or are subordinate to other call flows (Used By tab)
Code Generation / Application Transfer	The Code Generation tool generates the code necessary to transfer and install the application onto the target system. The Application Transfer tool transfers the application files to the target system so they can be installed.
Events String Editor	The Events String Editor displays both a list of all the variables defined for the application and a list of the variables selected to be treated as Call Data Events on the target system.
Node Inspector	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.
Preference Editor	The Preference Editor sets application-wide preferences for using IVR Designer.
Screen Mapping Tool	Screen Mapping captures screens from an IBM host machine and transfers them back to the PC. You can then display the screens and map the field positions to the HostEdit display within IVR Designer.
Simulation	The Simulation tool runs your application in a simulated environment. It is often helpful in finding problems within the application.
Verify Design	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.

Sharing resources

IVR Designer also offers options for sharing resources with other developers. For details about sharing resources, see [Working with application resources](#) on page 226.

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.

Use the Code Generation/Application Transfer tool to generate the code for the target system. For more information about generating the application source code, see [Generating the source code files](#) on page 252.

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 261.

Install the application on the target system

Before you can run the application you have transferred to the target system, you must install it on the target system. If the PC with the Avaya IVR Designer application has network connectivity to the target voice response system, the application can be installed from Avaya IVR Designer. If the PC does not have network connectivity to the target system, install the application using the SCI tool. For more information about installing the application using the SCI tool, see [Installing an application on the target system](#) on page 289.

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.



Note:

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

Related topics:

[The Application Parameters tab](#) on page 31

[The Environment tab](#) on page 33

[The Printing tab](#) on page 35

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.

Parameter	Description
Default Application Directory	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.
Application-Wide Language Selection	This parameter lets you select from the drop-down menu the language you want to use in your applications. 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: If you try to select a language that does not support Text-to-Speech options or FlexWord speech recognition, and you have used these resources in your application, IVR Designer displays a message instructing you to remove all uses of or references to these resources before you can select the new language. If you see such a message, you must remove the appropriate resources before continuing with the language change. When you change the language for an application, IVR Designer automatically changes the following: • The EBS phrase table • Variables to defaults When changing an application's language, you are responsible for: • Importing custom phrase tables for the new language • Modifying FlexWord lists  Note: All resources used in the application (Text-to-Speech, FlexWord lists, and EBS) must be available in the language you want to select. If not, IVR Designer does not let you change the language.
Application Feature Compatibility	This parameter lets you control whether warnings are shown for unsupported features on the target platform. The default setting is to show warnings for all target platforms. To hide warnings for Avaya Interactive Response only, clear the show warnings checkbox for that platform. To hide warnings for CONVERSANT R9 or earlier platforms only, clear the show warnings checkbox for those platforms.

	To hide all warnings, make sure the Hide all warnings for all applications checkbox has a check mark.

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.

Option group	Option (and type)	Description
Opening/Saving	Save desktop upon exit (checkbox)	When checked, saves your design pad (desktop) options so they are used whenever you reopen IVR Designer.
	Save desktop when saving application (checkbox)	When checked, saves your design pad (desktop) options whenever you save an application.
	Automatically open last application (checkbox)	When checked, upon starting IVR Designer, automatically opens whatever application you were working on the last time you had IVR Designer open.
	Automatically save application (checkbox and drop-down menu)	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.
Spell Check	Automatically Spell Check Phrases on Exit (checkbox)	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.  Note: This feature works only if you have both Microsoft Word and the Word Spell Checker installed on your PC.
	Language for Spell Checker (drop-down menu)	Defaults to the designated language for the Microsoft Word Spell Checker.
Call Flow Editor	Automatically name nodes (checkbox)	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.
	Multiple Document Interface (radio button)	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.  Note: You must restart IVR Designer before this option takes effect.
	Tabbed Interface (radio button)	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.

		 Note: You must restart IVR Designer before this option takes effect.

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 TAS application open and you

press F1 or select Contents... from the Help menu, the system displays the Avaya IVR Designer Help file for TAS 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.

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.

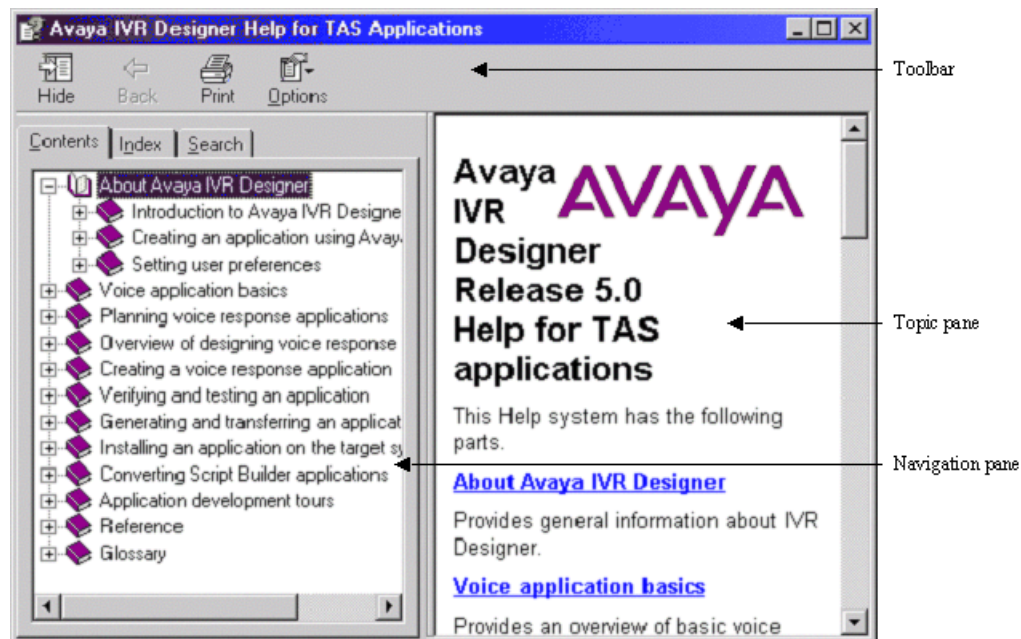
A PDF version of each Help file is available on the Avaya IVR Designer CD in the docs folder, and from the Product Documentation area of the following web site: <http://support.avaya.com/>

Related topics:

[About the Help window](#) on page 36

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:

Button name	Description

Hide/Show	If the navigation pane is open, click the Hide button to close the pane. If the navigation pane is closed, click the Show button to open the pane.
Back	Click the Back button to open the previously viewed Help topic in the topic pane.
Print	Click the Print 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.
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:

Tab name	Description
Contents	The Contents 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.
Index	The Index 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 Display button. You can also type a word into the text box above the list and press Enter or click the Display button when an index entry is highlighted. The selected Help topic is displayed in the topic pane.
Search	Using the Search 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 List Topics 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.

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.

Chapter 3: 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.

Related topics:

[Voice response application uses](#) on page 39

[Voice response system versus an attendant](#) on page 41

[Defining successful voice response applications](#) on page 41

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.

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 and trade show information, such as a 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 ticket rates, concert information, and sporting event times and results.

Voice response for database information requests

A voice response system can be connected to a system containing enterprise-specific information to access or store information in a host computer. Enterprise-specific information includes: published materials, bank accounts, and product descriptions. Database information requests include:

- Bank or investment accounts
- Magazine articles
- News wire stories
- Public records, such as deeds or licences
- Telephone directory listings
- Government records
- Catalog item descriptions

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.
- 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.

Chapter 4: Planning a voice response application

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
- 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](#) in the Avaya IR system help.

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.

Related topics:

[Use flowcharts](#) on page 46

[Use outlines](#) on page 46

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.

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.

Chapter 5: Designing a voice response application

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:

Information	Comments
Who are your callers?	Your callers have personal attributes that affect how they interact with the application you are designing. 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. 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. 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. Note any other personal attributes that could affect how callers will interact with your application, such as regional slang, accents, or education level.
How often do they call?	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. 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.
How well do they know the subject matter?	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.
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.

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.

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...	prompt...
To listen to information on travel to California, say 1.	For California travel information, say 1.
If you are calling to register for classes, press 3.	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. This example requires 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...	prompt...
Say 2 to receive the information by fax.	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...	prompt...
To hear sales information, press 1.	For sales, press 1.
For services, press 2.	For services, press 2.
If you would like to speak to an attendant, press 0.	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..."

Menus

Use the following guidelines when designing your applications that use menus.

Number of menu choices

Too many menu choices presented at once may confuse callers. In general, do not offer more than four or five choices in a single menu.

If you have more than four or five choices, separate them into small related groups and present more than one menu. For example, if you are giving information on seven parks, do not list them all in one menu. Your first menu could separate them into groups, and say:

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

If the caller presses 1: "For Stony Ridge, press 1 For Cantwell Cliffs, press 2 For Old Man's Cave, press 3"

If the caller presses 2: "For Seneca Rocks, press 1 For Rialto Beach, press 2 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?"

Related topics:

[Entering digit sequences](#) on page 56

[Validate a digit sequence entry](#) on page 57

[Confirming digit entries with callers](#) on page 58

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

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, 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.

Validate a digit sequence entry

When you know that a certain group of digit sequences are valid, use whatever techniques you can to validate caller input. This is especially important with spoken input, but can also be done for touchtone input. These validation methods help you increase the recognition accuracy for connected-digit sequences. The following sections describe three ways to validate an entry.

Custom grammars

You can limit the results of the recognizer to valid sequences by using a custom grammar obtained from Avaya. A grammar is the set of rules the recognizer uses to understand data, such as caller input.

A number of grammars are already provided with IVR Designer (a list can be seen in the Input ASR Mode field on the Prompt & Collect Response tab). Custom grammars are special

additions to this standard set, designed for your specific needs. For example, if you know that all valid callers will have telephone numbers from only a few telephone area codes, request a custom grammar that allows only the area codes you want. This helps the recognizer make accurate judgments about caller speech.

Checksums

Another way to validate an entry is to use a checksum. Some numbers (like credit card numbers) have a checksum built into the structure of the number, so that the number can be checked with a formula to determine if it is valid.

If the digit series you are accepting has a built-in checksum, use a data interface process (DIP), IVR Designer Set and Test node, or external function to evaluate the checksum. Once the system determines that the entry is valid, it can speak the entry back for confirmation, if necessary.

Database lookups

You can also validate an entry by looking for it in a database of valid numbers. For example, when accepting a telephone number, locate the number in the database of all customer telephone numbers. If the number is found, speak it back to the caller for confirmation, if necessary. If it is not found, the system can search the database for a telephone number that matches the second-best recognition result. If the second choice is found, the system assumes that this is what the caller entered, and can speak it back for confirmation.

Confirming digit entries with callers

It is important to confirm digit entries with callers.

Confirming single digit entries

In order for the system to speak back digits for confirmation, you must ensure that there is a recorded phrase associated with each digit. IVR Designer will supply the phrase tags for these digits, and allow you to record them. You can also use enhanced basic speech, in the language of your choice, to speak back digits.

Confirming digit sequences

With non-US English languages, the only reliable way to speak caller numeric entries back to them is as a series of digits, rather than a natural number. For example, a caller that speaks "one-hundred, twenty-three" will hear their input read back as "one two three." This should be adequate for most applications.

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...	prompt...
If this is correct, say 1, if it is not correct, say 2	Is this correct? Please say yes or no, now.

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...	Use phrases such as...
We are sorry, but your speech cannot be understood. Please repeat the category you want.	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:
"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.

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

Related topics:

[Application examples](#) on page 62

[Use touch-tone input when speech recognition fails](#) on page 63

[Modify a touch-tone-only application to include spoken input](#) on page 64

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?"
- 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.

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:

Word	Maps to option
listen to homework	Listen to homework option
homework	Listen to homework option
listen	Listen to homework option
leave a message for the teacher	Leave a message for the teacher option
leave a message	Leave a message for the teacher option
teacher	Leave a message for the teacher option
choose a different class	Choose a different class option
different class	Choose a different class option
different	Choose a different class option
class	Choose a different class option

If you plan to use 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.

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 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.

Bilingual, multilingual, and non-US English applications

Your Avaya IR system can support up to two different speech recognition software packages in several available languages. Use the guidelines in Bilingual, multilingual, and non-US English applications to effectively use speech recognition for languages other than US English.

Related topics:

[Bilingual applications](#) on page 66

[Multilingual applications](#) on page 67

Bilingual applications

The Avaya IR system allows you to provide callers with bilingual service. The system plays prompts and announcements, and recognize speech, in several languages. In some cases you can provide this type of service with a single application. In other cases, you will need to use a separate application for each language.

Selecting a language

All bilingual services, whether simple or complex, should ask callers for their preferred language at the beginning of the application. The system then selects the application (or part of the application) that speaks and recognizes speech in the language the caller chooses.

Use a language gate

The part of the application that asks the caller to pick one of two languages is called a language gate. Since the system can recognize only a single language for each Prompt & Collect statement, the language gate must be written carefully. The speech recognition language gate illustrated below includes two questions, one in each language. Each question asks the caller to say a vocabulary word in one of the languages. The application then branches to a different application, depending on the caller's response.

You can also have a language gate for a touchtone-only application. First, greet the callers in each language. Then say, for example, "For English, press one. Para Español, marque dos." The code, then executes a different application, based on caller input.

Dates, times, and monetary amounts

If your application does not need to speak dates, times, monetary amounts, or numbers to the caller, you can use a single application to handle the interactions in both languages. Write the application in two different parts, with each part prompting and recognizing speech in a single language. When the caller chooses a preferred language in the beginning of the application, branch to the correct section.

If your application must speak dates, times, monetary amounts, or numbers to the caller, each language must be processed in a separate application. A single application can have only one set of enhanced basic speech phrases, the phrases that are used to speak numbers. Therefore, each application can speak only numbers in a single language.

Enhanced basic speech allows the system to play commonly used words to callers from professional recordings made by Avaya. The best way to speak a numeric entry back to a caller

is to use enhanced basic speech for the local language. You can install as many of these languages as you have space for on your system disks. One language will be available for each application, but your system can support several different applications.

Multilingual applications

If you have at least two Avaya IR systems, you can process callers response in different languages. You need to configure different languages on the speech servers for recognition. You could start with the above language gate. Instead of transferring to the attendant if the caller does not say "yes" or "sí" at one of the expected times, you could transfer the call to an application on another system. The second system could include a language gate in the other two languages, then branch to the correct application (or to the attendant) depending on the caller's response.

Text-to-Speech 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.

Related topics:

[Using Text-to-Speech for prompts and announcements](#) on page 67

[Using both Text-to-Speech and prerecorded speech prompts and announcements](#) on page 68

[Getting the most out of Text-to-Speech](#) on page 68

[Testing Text-to-Speech applications](#) on page 69

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) < John Smith > "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.

Chapter 6: Creating a voice response application

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 559.

Related topics:

[Placing a node in a call flow](#) on page 71

[About node branches](#) on page 72

[Node editing guidelines](#) on page 73

[Name conflicts during paste actions](#) on page 75

[Selecting multiple nodes for cut, copy, and paste](#) on page 75

[Using drag-and-drop to cut, copy, and paste](#) on page 76

[Editing node attributes](#) on page 77

[Right-click menu options for nodes and empty node branches](#) on page 77

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.

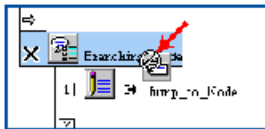
Drop the icon onto any open branch. Or, if you do not want to connect the node to the call flow, drop the icon below the last node.

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.

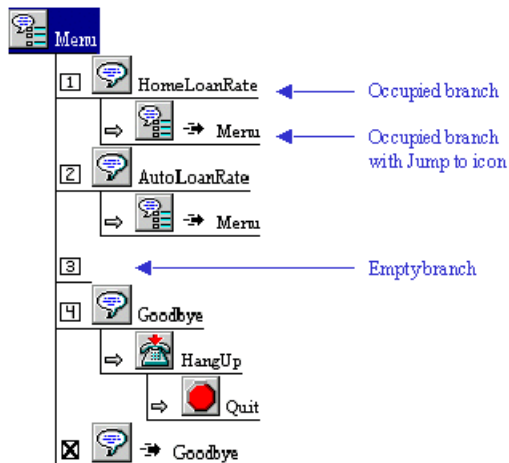
IVR Designer also lets you select multiple nodes within a call flow, but they must be contiguous nodes. That is, they must have the same parent node and be adjacent to each other on the call flow. As with single nodes, all subordinate nodes are copied or moved with the parent nodes.

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 for include:

- Menu Node
- Automenu Node
- 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
- Database tables (includes both table schemas and table data)
- Host definitions
- FlexWord lists
- External functions
- Encapsulated call flows

For practice copying nodes and applications, see [Copying TAS nodes and applications](#) on page 355.

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. Therefore, if you are attempting to paste nodes with shared resources, use the Refresh From Shared Resources tool before pasting. This lets you make sure that all resource pathnames are current before pasting.

 Note:

Keep in mind during paste operations that IVR Designer lets you overwrite a private copy of a shared resource by a shared version, or lets you overwrite a shared version with a private copy.

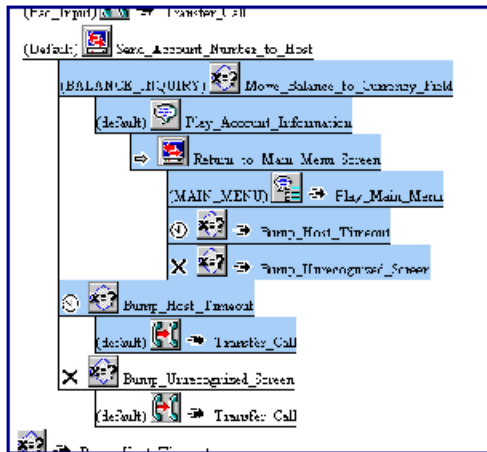
If the target application already has a resource with the same name as one in the nodes you are trying to paste, IVR Designer prompts you to specify whether you want to keep the original resource on the target system or overwrite it with the new one.

Selecting multiple nodes for cut, copy, and paste

IVR Designer lets you cut/copy/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.

 Note:

As shown in the following figure, you cannot, for example, select only the Set and Test nodes `Move_Balance_to_Currency_Field` and `Bump_Unrecognized_Screen` without also selecting the `Bump_Host_Timeout` node between them.



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. For example, in the figure above you cannot shift-click the `Transfer_Call` node to select the highlighted nodes.

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.



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



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 559.

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.

Menu option	Description
Cut	Removes the selected node and any subordinate nodes and copies them to the clipboard.

Copy	Copies the selected node and any subordinate nodes to the clipboard.
Paste	Inserts any nodes on the clipboard at the point where the cursor is. This option is available only after you have done a cut or copy action.
Branch Notes	Opens the Branch Note 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.
(Delete Branch)	When available, this option deletes the entire branch from the selected node.  Note: This action cannot be undone.
(Insert Branch)	When available, this option inserts a new branch into the call flow.
(Rename Branch)	When available, this option lets you rename the branch.
(Receive Field Mappings)	When available (Host Send and Host Send/Receive nodes only), this option maps fields received from the remote host to the appropriate fields in the selected Host Definition. This option is available only if: • At least one Host Definition Editor --> Field Definition tab --> Direction column field value is set to From Host or Both. • A new branch is inserted under the appropriate node.
Delete Node	Removes the selected node and any subordinate nodes from the call flow.  Note: This action cannot be undone.
Disconnect Node	Separates the selected node and any subordinate nodes from the call flow.
(Edit Flow)	(Active only with the Call Flow node) Opens the "encapsulated" call flow, making it the active call flow.

Edit Node	Opens or activates the Node Inspector, where you can edit the selected node's attributes.
(Find Jumped To Node)	(Active only for nodes that have the  icon following the node icon) Finds the node that the selected node jumps to.
Encapsulate	Converts the selected node to a Call Flow node and encapsulates that and any subordinate nodes, making them into a separate call flow.
(Expand Node)	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.
(Hide / Show Execution)	When available, hides or shows the execution of the node during Simulation.
Print Node	Prints out a summary of the selected node's attributes.
Rename Node	Lets you rename the node.
Set / Clear Breakpoint	Stops the execution of a Simulation at the node where it is set. A breakpoint is designated in a call flow with the  icon.
Verify Node	Opens and runs the Verify Design tool on only the selected node.
Print Call Flow	Prints out the active call flow similar to the design pad display of the call flow.
Verify Call Flow	Opens and runs the Verify Design tool on only the active call flow.
Close	Displays a list of all the currently open call flows, selecting a call flow from the list closes it.  Note: This option is available only when Tabbed Interface is selected on the Environment tab of the Preference Editor.

Open	Displays a list of all currently closed call flows, selecting a call flow from the list opens it.  Note: This option is available only when Tabbed Interface is selected on the Environment tab of the Preference Editor.

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.

Menu option	Description
Cut	Grayed out (inactive) on an empty branch.
Copy	Grayed out (inactive) on an empty branch.
Paste	Inserts any nodes on the clipboard at the point where the cursor is. This option is available only after you have done a cut or copy action.
Branch Notes	Opens the Branch Note 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.
(Delete Branch)	When available, deletes the entire branch from the selected node.  Note: This action cannot be undone.
(Insert Branch)	When available, inserts a new branch into the call flow.
(Rename Branch)	When available, lets you rename the branch.

Connect Node	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.
Copy Call Flow	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.  Note: The [Main] call flow and [HostFlow] call flows cannot be copied.
Edit Node	Opens the Node Inspector, where you can edit the selected node's attributes.
Jump to Node	Opens a dialog box that displays a list of nodes in the call flow to which you may jump from the currently selected node.
Reuse Call Flow	Similar to Copy Call Flow, except that the currently active call flow is not displayed and cannot be selected. Provides a link to the selected call flow, but does not make a copy of it
Print Call Flow	Prints out the active call flow similar to the design pad display of the call flow.
Verify Call Flow	Opens and runs the Verify Design tool on only the active call flow.
Close	Displays a list of the currently open call flows. Selecting a call flow from the list closes it.  Note: This option is available only when Tabbed Interface is selected on the Environment tab of the Preference Editor.
Open	Displays a list of the currently closed call flows. Selecting a call flow from the list opens it.

	 Note: This option is available only when Tabbed Interface is selected on the Environment tab of the Preference Editor.

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](#) on page 559.

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.

Element	Description
Node selection drop-down menu	The drop-down menu at the top of the Node Inspector lets you select which node to edit.
Node attribute fields	The fields below the Node selection drop-down menu let you set attributes for the selected node.
Node attribute tabs	When available, the tabs at the bottom of the Node attributes area let you select groups of attributes for editing.
Node Notes area	The Node Notes area at the bottom of the Node Inspector lets you type comments about the selected node.

Related topics:

[Selecting a node to view or edit attributes](#) on page 83

[Writing node notes](#) on page 83

[Right-click menu options for the Node Inspector](#) on page 84

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.

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, the Trace Variables node, and the Load Variables node.

Menu option	Description
Show Hints	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.
Show Node Notes	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.
Stay on Top	Toggles on/off. When checked (toggled on), causes the Node Inspector window to remain on top of other open IVR Designer windows.
New Element	Adds a new element to the variable definition.
Delete Element	Deletes an element from a variable definition.
Move Element Up	Moves an element upward in the variable definition table.
Move Element Down	Moves an element downward in the variable definition 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:

Tab name	Description
Named	Searches for text strings contained within the names of selected items.
Uses Of	Searches for locations where items are used within the application.
Unused	Searches for items that have been created, but not used, within the application.
Find Text	Searches for text strings located anywhere within the application.

Related topics:

[The Named tab](#) on page 85

[The Uses Of tab](#) on page 86

[The Unused tab](#) on page 87

[The Find Text tab](#) on page 87

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.

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.

Touchtone input used with spoken input

You can use 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.

Related topics:

[Separate touchtone and speech recognition applications](#) on page 88

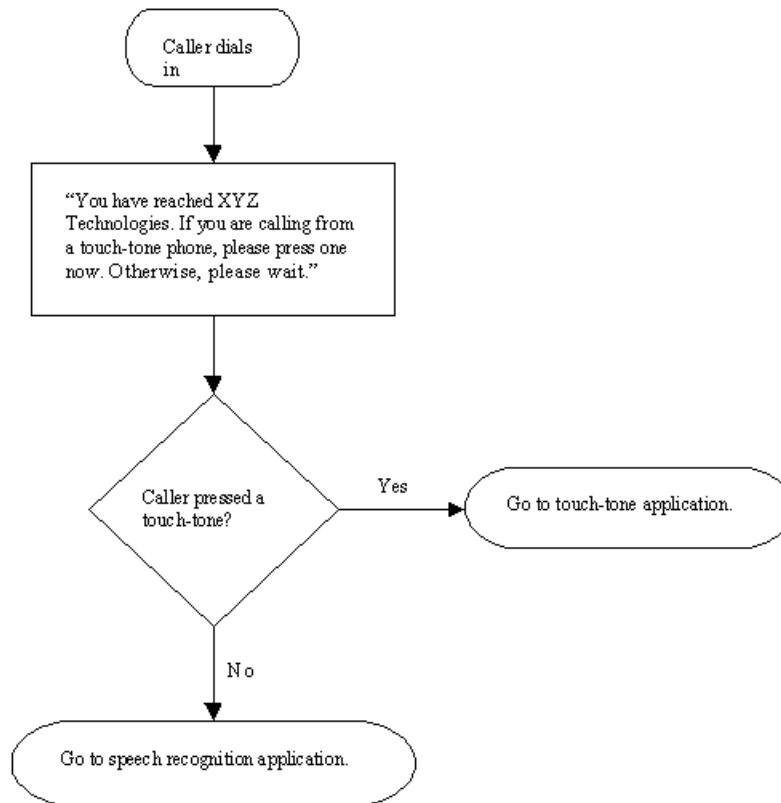
[A single touchtone and speech recognition application](#) on page 89

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:

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."

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

Note:

Host call flows differ from voice call flows, because they use only a certain set of nodes, and they are not linked to voice call flows. Host flows are dedicated to establishing and maintaining required communication with remote hosts.

For practice using call flows, see [Using embedded call flows in TAS applications](#) on page 400.

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.

Related topics:

[Displaying the Uses view](#) on page 91

[Displaying the Used By view](#) on page 92

[Dragging a call flow to create a copy](#) on page 93

[Right-click menu options for the Call Flow Inventory tool](#) on page 93

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.

Element	Description
	Indicates that the designated call flow is defined but contains no other call flows.
	Indicates that the designated call flow contains other call flows, but that the directory for viewing them is closed.
	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.
Uses tab	Lets you view the call flows defined for the current application and see what other call flows they reference (use).
Used by tab	Lets you view call flows and see which other call flows reference (use) them.

Expanding a call flow

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

Element	Description
Display pane	Displays the call flows currently defined for the application.
	At the primary level, signifies that the call flow is defined, but is not used, by any other call flow. At a subordinate level, signifies that the call flow uses the call flow in the level above.
	Indicates that the call flow is used by other call flows, but that the call flows using it are hidden (collapsed view).
	Indicates that the call flow is used by other call flows, which are displayed at a secondary level underneath (expanded view) The number in parentheses indicates how many times the parent call flow had used that subordinate call flow at the time the subordinate call flow was created.  Note: If one or more references to the subordinate call flow are later deleted, IVR Designer does not renumber the remaining references to it.

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: 

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: 

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.

Option	Description
Edit	Displays the selected call flow in the design pad, where you can edit it.

Delete	Deletes the selected call flow from the application. 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. You cannot delete a host flow except by deleting its host definition.
Expand	Expands the selected call flow and displays its contents.
Collapse	Collapses the selected call flow and hides its contents.
Expand All	Expands all call flows and displays their contents.
Collapse All	Collapses all call flows and hides their contents.

The Globals Manager

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

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
- FlexWord speech recognition
- External functions
- Remote hosts
- Phrases
- Prompts
- Database tables

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.

Element	Description
Title bar	Located at the top of the window. Contains the buttons for minimizing, maximizing, restoring, and closing the Globals Manager window.
Control bar	Located immediately below the title bar. Contains the tabstops, Show More button, and Show Less button (see the next three elements).
Show More icon	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.
Show Less icon	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.
Tabstop	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.
Resource manager name bar	Used to identify the individual global resource managers. Is distinguishable from other fields in a globals window by its magenta color.
Resource manager header bar	Located adjacent to the name bar (see above). Identifies the data columns within each global resource manager.
Data displays	Fields located underneath the name bar and header bars. Used to display the appropriate data for each column.

Related topics:

[Maneuvering in the Globals Manager](#) on page 96

[More about tabstops](#) on page 96

[Docking and undocking resource managers](#) on page 97

[Right-click menu options for the Globals Manager](#) on page 98

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 96.

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
- 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 the Globals 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 for the Globals Manager

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

Location	Menu option	Description
In the control bar	New Tabstop	Creates a new tabstop and places it on the control bar.
	Stay On Top (toggle option)	When checked, causes the Globals Manager to stay on top of all other IVR Designer windows. When unchecked, allows the Globals Manager to move to the background when another window is selected as the active window.
On an existing tabstop	New Tabstop	Creates a new tabstop and places it on the control bar.
On an existing tabstop	Delete Tabstop	Deletes the tabstop under the cursor.
On an individual resource manager's name bar	Open/Close (toggle option)	Opens or closes the individual resource manager.  Note: This action is inactive if the resource manager has been undocked from the Globals Manager.
On other resource managers' fields:		Calls up a choice menu. The choices available depend on which resource manager you are in. For more details on the specific menu choices available, see the

		description of the pertinent resource manager.

DPR Barge-in

Barge-in is a capability that allows callers to enter their responses during the prompt and have those responses recognized. IVR Designer has an enhanced TAS based Dial Pulse Recognition (DPR) to support barge-in while playing a prompt.

Related topics:

[Enabling DPR Barge-in](#) on page 99

[Automenu node attributes](#) on page 566

[Menu node attributes](#) on page 584

[Prompt and Collect node attributes](#) on page 591

Enabling DPR Barge-in

Barge-in can be enabled to accept the DPR input from the callers by using the Prompt and Collect, Menu, and Automenu nodes of IVR Designer. To enable DPR Barge-in, use the Node Inspector to do the following:

1. Set the **Allow Interrupt** attribute to **True**.
2. Set the **Input DPR Mode** attribute to a value other than **None**.



Note:

For DPR Barge-in to work correctly, there should be no noise in the analog line and echo cancellation should be in place. The DPR parameters can be tuned for better performance.

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 or variables, and compare them with other variables. For more information about system variables, see [System variable descriptions](#) on page 626. For practice using variables, see [Using variables and speech in TAS applications](#) on page 385.

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.
- 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.

Related topics:

[The Variables Manager window elements](#) on page 101

[Creating new \(custom\) variables](#) on page 116

[Renaming custom variables](#) on page 117

[Setting or editing a variable's attributes](#) on page 118

[Right-click menu options for the Variables Manager](#) on page 118

[Variables and arithmetic operations](#) on page 119

[Converting variable values](#) on page 121

[Variable comparisons](#) on page 128

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 [].

Value column

The value displayed in this field is the current value of the variable. This field remains blank until you run the Simulation tool, when it becomes populated with values generated during the simulation.

Type column

The Type chosen for a variable influences several of its other attributes. The six basic variable types and the default values and accepted input for each attribute are listed in the following table. The options in the following table are for US English only. For information about options specific to other EBS language packages, refer to the product documentation for your system.

Type	Spoken As	Input As	Length	Default	Notes
Character	A (tts) C (recorded)	Char	24		Accepts as input all standard ASCII characters, including special characters and punctuation. (The extended ASCII character set is not supported.)
Currency	AN\$D0 (tts) N\$ (recorded)	#*00	15	0.00	Accepts as input numerals (0-9), a dollar sign, a single leading minus sign, and a

					decimal point.
Date	ADMDYY (tts) D (recorded)	mmddyy	8	19980101	Accepts as input numerals (0-9), hyphens between numerals, slashes, and spaces.
Number	AN (tts) N (recorded)	#	10	0	Accepts as input numerals (0-9) and a leading minus sign.
Real	AND1 (tts) ND0 (recorded)	###	10	0.0	Accepts as input numerals (0-9), a leading minus sign, and a decimal point.
Time	ATHMAM (tts) T (recorded)	hhmm(AP) M	6	000000	Accepts as input numerals (0-9) and the letters A, P, and M.

Speak Using column

The displayed value tells what method will be used by the target voice response system to speak the variable. Text-to-Speech (TTS) options are available and appear only for languages that support TTS. Note that you can use TTS in IVR Designer only if you have purchased it as an option for your CONVERSANT or Avaya IR system. In most cases, you will probably find it preferable to use Recorded Speech. The following table describes the types of speech available from the Speak Using column.

Speech type	Description
Recorded	Digitally recorded spoken speech

tts (Text-to-Speech)	Synthesized speech produced according to predetermined algorithms

To change a Speak Using attribute to another value, click the field for the variable you want to change and select the type of speech you want from the drop-down menu.

Spoken As column

The value that appears in the Spoken As column of the Variables Manager depends on what type of variable it is and what value you set for it. For information about the Spoken As values for a particular type of variable, see of the following topics:

- [Spoken as options for character variables](#) on page 108
- [Spoken as options for currency variables](#) on page 109
- [Spoken as options for date variables](#) on page 111
- [Spoken as options for number variables](#) on page 112
- [Spoken as options for real variables](#) on page 113
- [Spoken as options for time variables](#) on page 115

When you select a new Type for the variable, IVR Designer automatically assigns it a default value in the Spoken As column. If you change the value in the Speak Using column from tts to recorded or from recorded to tts, the value in the Spoken As field automatically changes to the default value for the new Speak Using option.

To set or reset the values in the Spoken As field, click the field and select the option you want from the drop-down menu.

When you open the drop-down menu, a context-sensitive pop-up help message appears when you place the cursor over an option on the menu. This pop-up message gives you a brief description of what that option does.

Phrase Table column

A value appears in the Phrase Table column of the Variables Manager only if the Type of variable is number, the Speak Using speech type is "recorded," and the Spoken As value is NX. Otherwise, this field is empty.

When you select number as the Type, recorded as the Speak Using speech type, and NX as the Spoken As value, IVR Designer populates this field with the Phrase Table that is currently active.

To set or reset the values in the Phrase Table field, click the field and select the Phrase Table you want from the drop-down menu.

Input As column

The value that appears in the Input As column of the Variables Manager depends on what type of variable it is and what value you set for it.

When data is received into the variable, the way the data is input must match the format you specify for the field.

When you select a new Type for the variable, IVR Designer automatically assigns it a default value in the Input As column.

The following table describes the options available for each variable type.

Variable type	Default value	Other options	Description
Character	char		Accepts all standard ASCII characters as input. Stores input as a string without making any changes to it.
Currency	#*00		Accepts numerals, dollar signs (\$), minus signs (-), decimal points (.), and commas (.). If no characters are input after a decimal point (or no decimal point exists), IVR Designer automatically assigns zeroes.
Date	mmddyy	yyyymmdd	Accepts numerals, hyphens (-), slashes (/), and spaces. The option you select here determines how incoming data is interpreted by the target system.
Number	#		Accepts only numerical input and disregards any nonnumeric characters, such as commas or hyphens. The only exception is a leading minus sign (-), which indicates a negative number. All input is treated as integer values.

			If you want to preserve hyphens or other nonnumeric characters within a number, for example social security numbers or phone numbers, use the Character variable type.
Real	###		Accepts numerals, decimal points (.), and minus signs (-). May be input as any number of digits before and after the decimal point.
Time	hhmm(AP)M	hhmm hh24mm hhmmss	Accepts numerals, colons (:), decimal points (.), and the letters "A", "P", and "M". Strips out characters other than numerals and the letters "A", "P", and "M" when storing values.

To set or reset the values in the Input As field, click the field and select the option you want from the drop-down menu.

Length column

The purpose of the Length column is to set the maximum length that the selected variable can have. If the length of the variable input is greater than the set value, the input is truncated to fit within the allocated space.

The value that appears in the Length column depends on what type of variable it is and what value you set.

When you select a new Type for the variable, IVR Designer automatically assigns it a default value in the Length column.

You can enter any integer value you want in the Length field. (It may not always make good sense to do so, however.) The following table describes the default values for each Type of variable.

Variable type	Default length	Description
Character	24	
Currency	15	When setting this value, you must include the decimal point (.), the dollar sign (\$), and the minus sign (-).
Date	8	This value cannot be reset by the user.
Number	10	Values may range from -2,147,483,648 to 2,147,483,647.
Real	10	When setting this value, you must include the decimal point (.) and the minus sign (-).
Time	6	This value cannot be reset by the user.

To set or reset a value in the Length field, click the field and type the new value you want. You cannot edit or reset the Length values for system variables.

Default column

The value that appears in the Default column depends on what type of variable it is and what value you set for it.

When you select a new Type for the variable, IVR Designer automatically assigns it a default value in the Default column.

The following table describes the default values for the Default column for each variable type.

Variable type	Default value	Description
Character	(none)	All character values in the standard ASCII character set are accepted, including spaces.
Currency	0.00	Only numerical values are accepted. All decimal values are rounded to the nearest cent. The value is always displayed as dollars and cents.

		Any characters other than numerals (including spaces) are ignored and removed.
Date	19980101	Any new value must be a valid date, using the yyyymmdd format. If you attempt to enter an invalid date value, IVR Designer rejects it and returns the field value to the last valid value that was entered into the field.
Number	0	Only numerical values and leading minus symbols are accepted. A decimal point is ignored in any value containing one (such as 234.56).
Real	0.0	Only numerical values and the decimal point and minus sign are accepted. All values are rounded to the number of places specified in the Spoken As column and displayed with the appropriate number of decimal places. If you attempt to enter an invalid (non-numeric) value, IVR Designer rejects it and returns the field value to the last valid value that was entered into the field.
Time	000000	Any new value must be a valid time, using the hhmmss format. If you attempt to enter an invalid time value, IVR Designer rejects it and returns the field value to the last valid value that was entered into the field.

To set or reset the values in the Default column, click the field for the desired variable and type the new value you want.

System column

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

Related topics:

[Spoken as options for character variables](#) on page 108

[Spoken as options for currency variables](#) on page 109

[Spoken as options for date variables](#) on page 111

[Spoken as options for number variables](#) on page 112

[Spoken as options for real variables](#) on page 113

[Spoken as options for time variables](#) on page 115

Spoken as options for character variables

For recorded speech, individual characters in the string are spoken in succession, with the specified inflection, if any. For Text-To-Speech (TTS), the string may either be treated as individual characters or read as words.

For both types, a null character field is spoken as nothing. If a character within a field is not valid for that type, it is ignored.

The following table summarizes the Spoken As options available when the variable is of type character.

The following options are for US English only. For information about options specific to other EBS language packages, refer to the product documentation for your system.

Speech type	Option	Description
Recorded	Crmf	Speaks the first character with rising inflection, the last character with falling inflection, and all other characters with medial inflection.
	Crmm	Speaks the first character with rising inflection and all other characters with medial inflection.
	Cmmf	Speaks the last character with falling inflection and all other characters with medial inflection.

	Cmmm	Speaks all characters with medial inflection.
	C (default)	Speaks all characters with medial inflection (same as Cmmm).
TTS	AC	Spells out the characters in the string, for example, the character string "Avaya" is spoken as "A - V- A -Y -A ".
	A (default)	Speaks the character string as words, if possible, for example, the character string "Avaya" is spoken as "Avaya".

Spoken as options for currency variables

Currency variables are actually specialized forms of number variables. All currency variable Spoken As formats begin with N\$ (for recorded speech) or AN\$ (for TTS). In most cases, this is followed by a D and a number. This combination represents the number of decimal places to assign to the number. The following tables shows how the currency input 12397 is spoken using each option.

Recorded speech type

Option	Description
N\$ (default)	Speaks the number as whole dollars (same as N\$D0).
N\$D0	Speaks the number as whole dollars, for example, "Twelve thousand three-hundred ninety-seven dollars".
N\$D2	Speaks the number as dollars and cents (treating the last two digits as decimal places), for example, "One hundred twenty-three dollars and ninety-seven cents".
N\$D1	Speaks the number as dollars and tenths of dollars; for example, "One thousand two-hundred thirty-nine point seven dollars".
N\$D3	Speaks the number as dollars and thousandths of dollars, for example, "Twelve point three nine seven dollars".
N\$D4	Speaks the number as dollars and ten-thousandths of dollars, for example, "One point two three nine seven dollars".
N\$D5	Speaks the number as dollars and hundred-thousandths of dollars, for example, "Point one two three nine seven dollars".

Option	Description
N\$D6	Speaks the number as dollars and millionths of dollars, for example, "Point zero one two three nine seven dollars".
N\$D7	Speaks the number as dollars and ten-millionths of dollars, for example, "Point zero zero one two three nine seven dollars".
N\$D8	Speaks the number as dollars and hundred-millionths of dollars, for example, "Point zero zero zero one two three nine seven dollars".
N\$D9	Speaks the number as dollars and billionths of dollars, for example, "Point zero zero zero zero one two three nine seven dollars".

TTS speech type

Option	Description
AN\$D2	Speaks the number as dollars and cents (treating the last two digits as decimal places), for example, "One hundred twenty-three dollars and ninety-seven cents".
AN\$D0 (default)	Speaks the number as rounded to the nearest whole dollar, for example, "Twelve thousand three-hundred ninety-seven dollars".
AN\$D1	Speaks the number as dollars and tenths of dollars, for example, "Twelve hundred thirty-nine point seven dollars".
AN\$D3	Speaks the number as dollars and thousandths of dollars, for example, "Twelve point three nine seven dollars".
AN\$D4	Speaks the number as dollars and ten-thousandths of dollars, for example, "One point two three nine seven dollars".
AN\$D5	Speaks the number as dollars and hundred-thousandths of dollars, for example, "Point one two three nine seven dollars".
AN\$D6	Speaks the number as dollars and millionths of dollars, for example, "Point zero one two three nine seven dollars".
AN\$D7	Speaks the number as dollars and ten-millionths of dollars, for example, "Point zero zero one two three nine seven dollars".
AN\$D8	Speaks the number as dollars and hundred-millionths of dollars, for example, "Point zero zero zero one two three nine seven dollars".
AN\$D9	Speaks the number as dollars and billionths of dollars, for example, "Point zero zero zero zero one two three nine seven dollars".

Spoken as options for date variables

Date formats always begin with a D (for recorded speech) or AD (for TTS). These are appended by a combination of the following:

- M - The month spoken as a number
- MSP - The month spoken by name
- D - The day spoken as a number
- Y - The year spoken as a two-digit number
- YY - The year spoken as a four-digit number

The following table shows how the date input 112397 is spoken using each option.

Speech type	Option	Spoken as
Recorded	DMDYY	"Eleven (pause) twenty-three (pause) nineteen ninety-seven"
	DMDY	"Eleven (pause) twenty-three (pause) ninety-seven"
	DMD	"Eleven (pause) twenty-three"
	DMSPDYY	"November twenty-third (pause) nineteen ninety-seven"
	DMSPDY	"November twenty-third (pause) ninety-seven"
	DMSPD	"November twenty-third"
	D (default)	Same as DMSPDY
TTS	ADMDYY (default)	"Eleven (pause) twenty-three (pause) nineteen ninety-seven"
	ADMSPDYY	"November twenty-third (pause) nineteen ninety-seven"
	ADMDY	"Eleven (pause) twenty-three (pause) ninety-seven"

	ADMSPDY	"November twenty-third (pause) ninety-seven"
	ADMD	"Eleven (pause) twenty-three"
	ADMSPD	"November twenty-third"

Spoken as options for number variables

Number variables are spoken as numbers, with the specified inflection, if any. A negative number is spoken with the word minus in front of the number.

IVR Designer does not support speaking numbers in the billions or trillions, because those numbers are generally outside the range of a number variable (numbers greater than 2,147,483,647 and less than -2,147,483,648). The phrases billion and trillion, however, are included in the Enhanced Basic Speech (EBS) packages.

If your application requires speaking numbers in the billions or trillions, you can:

1. Write an external function that accepts a larger number in the form of an ASCII string.
2. Parse the string (using a Set and Test node) getting the amounts of billions and trillions as substrings.
3. Convert the resulting substrings to integer values.
4. Speak the resulting integer values, inserting the phrases for billion or trillion where appropriate.

The following table shows how the number input 1522 is spoken using each option.

Speech type	Option	Spoken as
Recorded	Nrmf	"One thousand five hundred twenty-two" (The "One" is spoken with rising inflection, the "two" with falling inflection, and all other digits with medial inflection.)
	Nrmm	"One thousand five hundred twenty-two" (The "One" is spoken with rising inflection and all other

		digits with medial inflection.)
	Nmmf	"One thousand five hundred twenty-two" (The "two" is spoken with falling inflection and all other digits with medial inflection.)
	Nmmm	"One thousand five hundred twenty-two" (All digits are spoken with medial inflection.)
	N (default)	Same as Nmmm.
	NX	Speaks the phrase having this number. (For example, for the number 1522, phrase number 1522 from the designated phrase table is spoken, not the number.) If you select the NX option, you must also designate a phrase table in the Phrase Table column.
TTS	AN (default)	Speaks the number as a number. ("One thousand five hundred twenty-two")
	AC	Speaks the number as individual digits. ("One five two two")

Spoken as options for real variables

Real variables are specialized forms of number variables. For this reason, all real variable Spoken As formats begin with N for recorded speech, or with AN for TTS, followed by a D and a number. This combination represents the number of decimal places to assign the

number. The decimal point is spoken as "point". The following tables show how a literal real input of 123.97 is spoken using each option.

Recorded speech type

Option	Description
ND	Speaks the number as rounded to the nearest whole number, for example, "One hundred twenty-four"
ND1 (default)	Speaks the number rounded to the nearest tenth, for example, "One hundred twenty-four point zero"
ND2	Speaks the number rounded to the nearest hundredth, for example, "One hundred twenty-three point nine seven"
ND3	Speaks the number rounded to the nearest thousandth, for example, "One hundred twenty-three point nine seven zero"
ND4	Speaks the number rounded to the nearest four decimal places, for example, "One hundred twenty-three point nine seven zero zero"
ND5	Speaks the number rounded to the nearest five decimal places, for example, "One hundred twenty-three point nine seven zero zero zero"
ND6	Speaks the number rounded to the nearest six decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero"
ND7	Speaks the number rounded to the nearest seven decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero zero"
ND8	Speaks the number rounded to the nearest eight decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero zero zero"
ND9	Speaks the number rounded to the nearest decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero zero zero zero"

TTS speech type

Option	Description
ND	Speaks the number as rounded to the nearest whole number, for example, "One hundred twenty-four"
ND1 (default)	Speaks the number rounded to the nearest tenth, for example, "One hundred twenty-four point zero"
ND2	Speaks the number rounded to the nearest hundredth, for example, "One hundred twenty-three point nine seven"
ND3	Speaks the number rounded to the nearest thousandth, for example, "One hundred twenty-three point nine seven zero"

Option	Description
ND4	Speaks the number rounded to the nearest four decimal places, for example, "One hundred twenty-three point nine seven zero zero"
ND5	Speaks the number rounded to the nearest five decimal places, for example, "One hundred twenty-three point nine seven zero zero zero"
ND6	Speaks the number rounded to the nearest six decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero"
AND7	Speaks the number rounded to the nearest seven decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero zero"
AND8	Speaks the number rounded to the nearest eight decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero zero zero"
AND9	Speaks the number rounded to the nearest nine decimal places, for example, "One hundred twenty-three point nine seven zero zero zero zero zero zero zero"

Spoken as options for time variables

There is only one Spoken As format for time variables. This format speaks the variable using a 12-hour clock and the abbreviations "AM" or "PM".

For example, a 24-hour clock time of 14:30 is spoken as "Two-thirty (pause) P M".

The following table describes the Time Spoken As options.

Speech type	Option	Description
Recorded	THMAM	Speaks the variable using a 12-hour clock and the abbreviations a.m. or p.m.
	T (default)	Same as THMAM
TTS	ATHMAM (default)	Speaks the variable using a 12-hour clock and the abbreviations a.m. or p.m.

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 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.

When you create a new variable, the Variables Manager does the following:

- Places the new variable (along with all other variables) in the grid in alphabetical order by name.
- Gives the new variable the following default values:
Value (blank) Type character Spoken As char Speak Using (see Note below) Phrase Table (blank) Input As char Length 24 Default (blank) System (blank)



Note:

The Speak Using default value depends on the language you have selected. If the application language supports TTS, TTS is selected as

the default. If the application language does not support TTS, recorded is selected as the default.

6. Set the attributes for the new variable as desired. For the procedure, see [Setting or editing a variable's attributes](#) on page 118.

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.
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.

Option	Description
New...	Lets you create a new (custom) variable.
Delete	Deletes the selected variable from the Variables Manager. System variables cannot be deleted.
Rename...	Lets you rename the selected custom variable without altering its attributes.
Find Uses of	Opens the Find window, searches for Uses Of the selected variable, and presents the results of the search in the Find window results field.
Sharing...	Inactive option in the Variables Manager.
Stay on Top	(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 "Stay on Top," the window actually on top will be the last window you clicked in. All windows toggled to "Stay on Top," however, will stay on top of any other IVR Designer windows that are open.
Use TTS only	When checked, changes all values in the Speak Using column to "tts" (Text-to-Speech). Checking this box also changes all Spoken As formats to the appropriate tts defaults.
Use Recorded only	When checked, changes all values in the Speak Using column to "recorded" (Recorded Speech). Checking this box also changes all Spoken As formats to the appropriate recorded defaults.

Variables and arithmetic operations

The following is a summary of the rules governing arithmetic operations between any two variables of the same or different classes:

- Addition and subtraction can be performed only on variables of the same class (Currency, Number, and Real variable classes only).

The exceptions to this rule are addition and subtraction involving Date and Time variables. A Number variable can be added to or subtracted from a Date or Time variable.

- Multiplication and division must involve either a Number or a Real variable.
- No arithmetic operations can be performed on any Character variable.

Rules for currency variables

- If both variables are Currency variables, they can be added or subtracted.
- Multiplication or division can be performed between a Currency variable and a Number or a Real variable, provided the destination is a Currency variable.

Rules for date variables

- Date values are stored internally in the yyyyymmdd format.
- A Number variable may be added to or subtracted from a Date variable. For example, adding 32 to a Date value of 19980112 (January 12, 1998), results in a value of 19980213 (February 13, 1998).
- When adding or subtracting a Number variable from a Date variable, the Number variable is treated as days. For example, to add 2 months to the date March 12, 1998, you must populate a Number variable with the value 61 (19 days left in March + 30 days for April + 12 days for May = 61 days).
If you populate the Number variable with a value of 2, the resulting Date value will be March 14, 1998 (19980314).

Rules for number variables

- The maximum and minimum values that a Number variable can have are 2,147,483,647 and -2,147,483,648, respectively. Values outside this range will produce errors and unpredictable results.
- If both variables are Number variables, they can be added, subtracted, multiplied, or divided.
- Multiplication or division can be performed between a Number and a Currency variable only if the destination is a Currency variable.
- A Number and a Real variable may only be multiplied or divided (not added or subtracted). If the destination is a Number variable, the resulting value is always an integer. Decimal places are ignored and discarded. For example, $3 * 12.15$ results in a value of 36.45, which is stored and used as 36. Also, $3 + 12.15$ is not a valid operation in IVR Designer.

Rules for real variables

- If both variables are Real variables, they can be added, subtracted, multiplied, or divided.
- Multiplication or division can be performed between only a Real and a Currency variable if the destination is a Currency variable.

- A Real and a Number variable can only be multiplied or divided (not added or subtracted). For example, $3 * 12.15$ results in a value of 36.45. Also, $3 + 12.15$ is not a valid operation in IVR Designer.
- When setting the field length for a Real variable used as a destination for arithmetic operations, remember the following:
 - One place must be reserved for the decimal point and one for the minus symbol. Make sure you allow enough places to include possible decimal points and minus symbols in any computation results you may get.
 - If you do not allow for enough digits in the destination variable field length, the MathDIP returns only the digits that will fit in the allotted space.

Rules for time variables

- Time values are stored internally in a 24-hour hhmmss format.
- A Number variable can be added to or subtracted from a Time variable. For example, adding 600 to a Time variable of 205500 (8:55 p.m.) results in a value of 210500 (9:05 p.m.).
- When adding or subtracting a Number value from a Time value, the Number value is treated as seconds. For example, to add 10 minutes to the time 12:31, you must populate a Number variable with the value 600 (10 minutes times 60 seconds = 600 seconds). If you populate the Number variable with a value of 10, the resulting Time display is unchanged, because the minutes portion of the time value remains unchanged.



Note:

: The seconds portion of the Time variable is not currently supported in the Spoken As formats in IVR Designer.

Converting variable values

When data of one field type is assigned to another field type, IVR Designer converts the value to the type of the field receiving the value. How IVR Designer handles this depends on the type of data in the original field.



Note:

: If a conversion cannot be done, IVR Designer populates the target variable with the previous value, if there is one. Otherwise, IVR Designer uses the default value for the target variable type.

Related topics:

[Converting character variable values](#) on page 122

[Converting currency variable values](#) on page 123

[Converting date variable values](#) on page 124

[Converting number variable values](#) on page 125

[Converting real variable values](#) on page 126

[Converting time variable values](#) on page 127

Converting character variable values

IVR Designer converts character variable values to other variable types according to the rules in the following table.

Type of field converting to	Rules and comments	Examples
Character	If the length of the field receiving the value is less than of the original field, IVR Designer truncates the string to the length of the new field.	If the string "This string is 34 characters long." is converted to a field limited to 24 characters, the result is: "This string is 34 charac"
Currency	IVR Designer strips out all characters except numbers, dollar signs (\$), and leading minus symbols (-) and converts the resulting string to a number.	The string "12-31-97" is converted to: 123197 The string "This is a string." cannot be converted, so the Currency default is used: 0.00
Date	IVR Designer converts the original value to the date format yyyymmdd.	The string "12-31-97" is converted to: 19971231 The string "800-555-1536" cannot be converted, so the Date default is used: 19980101
Number	If the string being converted contains only numeric characters, IVR Designer converts the string to an integer value. IVR Designer ignores any leading spaces. If there is a leading minus sign (-), IVR Designer assigns the integer a negative value. All non-numeric characters except the minus symbol (-) are stripped out.	The string "-1235ba-6" is converted to: -12356 The string "This is a string." cannot be converted, so the Number default is used: 0

Real	IVR Designer strips out all except number and leading negative sign (-) characters and converts the resulting string to a number.	The string "12-31-97" is converted to: 123197 The string "This is a string." cannot be converted, so the Real default is used: 0.0
Time	IVR Designer converts the original value to the time format hhmmss.	The string "1231pm" is converted to: 123100 The string "12-31-97" cannot be converted, so the Time default is used: 000000

Converting currency variable values

IVR Designer converts currency variables to other variable types according to the rules in the following table.

Type of field converting to	Rules and comments	Examples
Character	Since the currency variable is stored as a special type of character variable, currency values convert intact.	The value \$123.45 is converted to: "\$123.45"
Currency	If the length of the field receiving the value is less than of the original field, IVR Designer truncates the value to the length of the new field. Dollar signs are stripped out before conversion.	The value \$123.45 is converted to: 123.45 If the variable \$12345.67 is converted to a field limited to 7 characters, the result is: 12345.6
Date	IVR Designer cannot convert currency values to date values because of the leading dollar sign. The previous value or default date value is used.	The variable \$123.45 cannot be converted, so the Date default is used: 19980101
Number	IVR Designer cannot convert currency values	The variable \$123.45 cannot be converted, so

	to number values because of the leading dollar sign. The previous value or default number value is used.	the Number default is used: 0
Real	IVR Designer strips out all except numbers, decimal points, and leading minus symbols and converts the resulting string to a number.	The variable \$123.45 is converted to: 123.45
Time	IVR Designer cannot convert currency values to time values because of the leading dollar sign. The previous value or default time value is used.	The variable \$123.45 cannot be converted, so the Time default is used: 000000

Converting date variable values

IVR Designer converts date variables to other variable types according to the rules in the following table.



Note:

: IVR Designer stores date variables in the yyyyymmdd format and converts them from there.

Type of field converting to	Rules and comments	Examples
Character	IVR Designer converts the date variable value to a character string using the yyyyymmdd format.	The date 12-31-97 is converted to: "19971231"
Currency	IVR Designer converts the date variable value to a number.	The date 12-31-97 is converted to: 19971231
Date	IVR Designer converts date variables intact.	The date 12-31-97 is converted to: 19971231
Number	Since date variables as stored are composed of eight numerical	The date 12-31-97 is converted to: 19971231

	characters, IVR Designer converts date variables to numbers easily.	
Real	IVR Designer converts the date variable to a number.	The date 12-31-97 is converted to: 19971231
Time	Because of the format in which date variable values are stored (yyyymmdd), IVR Designer is unable to convert dates to time variables.	The date 12-31-97 cannot be converted, so the Time default is used: 000000

Converting number variable values

IVR Designer converts number variables to other variable types according to the rules in the following table.

Type of field converting to	Rules and comments	Example
Character	IVR Designer converts the number variable value to a character string.	The number 123197 is converted to: "123197" The number 13579246 is converted to: "13579246"
Currency	IVR Designer converts the number to a number.	The number 123197 is converted to: 123197
Date	IVR Designer tries to interpret the number variable value to see if it can convert it to a date format. If it can, it converts it to the yyyymmdd format.	The number 123197 is converted to: 19971231 The number 13579246 cannot be converted, so the Date default is used: 19980101
Number	Number variables are converted "as is".	The number 123197 is converted to: 123197
Real	IVR Designer converts the number variable to a number.	The number 123197 is converted to: 123197

Time	IVR Designer tries to interpret the number variable value to see if it can convert it to a time format. If it can, it converts it to the hhmmss format.	The number 123157 is converted to: 123157 The number 13579246 cannot be converted, so the Time default is used: 000000

Converting real variable values

IVR Designer converts real variables to other variable types according to the rules in the following table.

Type of field converting to	Rules and comments	Examples
Character	Since the real variable is stored as a special type of character variable, real values convert intact.	The value 123.4567 is converted to: "123.4567"
Currency	IVR Designer strips out all except numbers, leading minus symbols, and decimal points and converts the resulting value to a number.	The variable 123.4567 is converted to: 1234567
Date	IVR Designer strips out all except numbers and converts the result to a date value using the yyyyymmdd format.	The variable 12.3197 is converted to: 19971231 The variable 123.4567 cannot be converted, so the Date default is used: 19980101
Number	IVR Designer strips out all but number characters and leading minus symbols.	The variable 123.4567 is converted to: 1234567
Real	If the length of the field receiving the value is less than of the original field, IVR Designer truncates the value to the length of the new field.	If the variable 12345.6789 is converted to a field limited to 7 characters, the result is: 12345.6 (the decimal point counts as one character)

Time	IVR Designer strips out all except number characters and converts the result to a time value using the hhmmss format.	The variable 12.3157 is converted to: 123157 The variable 123.4567 cannot be converted, so the Time default is used: 000000

Converting time variable values

IVR Designer converts time variables to other variable types according to the rules in the following table.



Note:

: Time variables are stored using the hhmmss format. All conversions are made using those values.

Type of field converting to	Rules and comments	Examples
Character	Since the time variable is really a special type of character variable, time values convert intact.	The value 12:34 (PM) is converted to: "123400"
Currency	IVR Designer converts the time variable value to a number.	The variable 12:34 (PM) is converted to: 123400
Date	IVR Designer converts the result to a date value using the yyymmdd format.	The variable 12:19:57 (PM) is converted to: 19571219 The variable 12:34 (PM) cannot be converted, so the Date default is used: 19980101
Number	IVR Designer converts the time variable value to an integer.	The variable 12:34 (PM) is converted to: 123400
Real	IVR Designer converts the time variable value to a number.	The variable 12:34 (PM) is converted to: 123400
Time	IVR Designer converts the time variable value intact.	The variable 12:34 (PM) is converted to: 123400

Variable comparisons

When comparing variables, IVR Designer uses the following general rules:

- Only variables of the same class can be compared in operations involving variables. For example, a character variable can only be compared with another character variable, a number variable can only be compared with another number variable, and so on.
- You can use either a variable or a literal value when performing variable comparison operations. However, if you use a literal value, it must take a form appropriate for the class of variable you want to compare it with. For example, you cannot compare a literal value of "abc" with a number variable, because "abc" is a character string.
- When comparing character variables, case is taken into account according to standard rules for character variable comparison.

Comparison operands

When comparisons are performed between two values, the operation returns a value of "true" or "false", depending on the type of comparison being performed. The following table describes the actions of operands used in variable comparisons.

Operand	Comparison action
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to

Comparisons with character values

You can perform comparisons between two character variables or between literal values and character variables. You can also make comparisons using a null character value.

IVR Designer compares character values using a lexicographical comparison. This means that each character of the two fields, starting from the left, is compared one at a time, until one field has no more characters or a mismatch occurs.

If no mismatch occurs and the fields are of the same length, they are considered equal. If a field has fewer characters and all previous characters have matched, the longer field is considered greater. If a mismatch occurs, the field with the character having the greater value at the mismatch is considered the greater variable.

Characters are compared according to the standard character ordering scheme for programming languages. Thus, the following comparisons would all return a value of "true":

"" < any other string

"12" < "2"

"0123" != "123"

"ABC" < "AD"

"000" < "999"

"999" < "AAA"

"AAA" < "AAB"

"AAA" < "AAAA"

"AAA" < "A.A."

"AAA" < "aaa"



Note:

: Two character variables can contain numbers that are numerically equal but not lexicographically equal, as in the third example above. If you want a numeric comparison, you must first convert both variables to number variables, and then do the comparison.

Comparisons with other values

IVR Designer performs straightforward arithmetic comparisons on all other variable classes.

Working with speech recognition

What is Natural Language Speech Recognition (NLSR)?

Within the context of interactive voice response (IVR) systems and applications, the term speech recognition, also called automatic speech recognition or advanced speech recognition (ASR), is the ability of an IVR system to recognize spoken responses from a caller and either convert it to text or use the results to initiate some system action. IVR Designer supports FlexWord speech recognition, which uses phonemes to recognize longer words and phrases.

Various forms of ASR are very good at what they do, and in many applications they are sufficient. They do, however, share some limitations, both in the number of words or phrases that can be recognized, and also in their inability to take into account grammatical sentence structure. While these ASR technologies can recognize specific words or phrases, even when extraneous words or phrases are thrown in by the caller, they have no ability to recognize what part the recognized speech plays in the overall statement. In other words, these ASR technologies are all designed to recognize specific words or phrases, but not to interpret what they recognize.

Natural Language Speech Recognition (NLSR) takes the speech recognition process several steps further, by providing a more natural conversational interface with IVR systems. Not only can NLSR be used to recognize particular words and phrases, it can also interpret and assign meaning to the speech it recognizes.

For example, under the more basic forms of ASR, a caller can respond only to specific prompts, such as "Say 'one' if you want information about..." or "Say 'yes' if this is correct." NLSR enables

you to write applications that ask the caller more open-ended questions, such as a banking application that presents the caller with a list of options and then asks "What would you like to do?" Then, when the caller responds "I'd like to know the balance of my checking account, please," the system can recognize what kind of information the caller is asking for (the balance in a checking account) and can automatically direct the call to a new prompt that would ask for the caller's checking account number.

NLSR is able to take into account grammatical structures. This allows NLSR, for instance, to recognize and deal appropriately with differences in statements, like the following caller responses:

"I would like to fly from Chicago to LAX."

"I need to get from LAX to Chicago."

NLSR is also capable of understanding natural numbers ("seventy-six" instead of "seven six"), natural dates ("July 26th" instead of "zero seven two six") and natural currency ("25 dollars" instead of "two five zero zero").

You can access NLSR resources through the FlexWords Manager.

NLSR options can be used only if NLSR is available for the application language selected. NLSR options can be implemented on the target voice response system only if you have purchased the NLSR package for your system.

What is FlexWord speech recognition?

FlexWord speech recognition recognizes words by matching phonemes (parts of spoken words). Individual words are grouped together in word lists, also called grammars. FlexWord lists are grouped together and accessed through the FlexWords Manager.

FlexWord options can be used only if FlexWord speech recognition is available for the application language selected. FlexWord options can be implemented on the target voice response system only if you have purchased the FlexWord speech recognition package for your system.



Note:

FlexWord is not supported on Avaya IR.

The FlexWords Manager

The FlexWords Manager displays the contents of all FlexWords grammars, which are word lists that are defined for the application, and the recognition resource type for NLSR grammars.

Opening the FlexWords Manager

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

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

Related topics:

[Importing FlexWord grammars \(word lists\)](#) on page 131

[Creating a FlexWord list](#) on page 131

[Creating a Natural Language Speech Recognition \(NLSR\) word list](#) on page 132

[Creating an NLSR word list for a TDD application](#) on page 133

[Creating stub grammars](#) on page 133

[Renaming a Word List](#) on page 134

[Editing a word list](#) on page 134

[Right-click menu options for the FlexWords Manager](#) on page 135

Importing FlexWord grammars (word lists)

You may sometimes want to use FlexWord lists that you or other developers have created for other applications. IVR Designer lets you import FlexWord lists from outside the current application. This saves you the trouble of having to create new FlexWord lists each time you want to use one in your application.

To import a FlexWord list into your application:

1. Open the Application Resources tool.
2. Click the Import tab at the bottom of the window.
3. (Optional) From the drop-down Drive menu, select the mapped network drive containing available resources.
4. Select FlexWord Grammar from the Application Resources drop-down menu at the top of the window.
IVR Designer displays a list of available FlexWord lists and their locations.

**Note:**

The FlexWords option is available only if it exists for the language you have selected.

5. Select whether you want to import the resource as Shared or as a Private Copy by clicking the appropriate option.
6. Do one of the following:
 - Highlight the line entry for the FlexWord list you want, then click the Import button.
 - Double-click the line entry for the word list you want.
IVR Designer imports the selected FlexWord list and displays it as an entry in the FlexWords Manager.

Creating a FlexWord list

IVR Designer makes it easy for you to create new FlexWord lists to use in your applications. Once created, these word lists become part of the application.



Note:

The FlexWord option is available only if it exists for the language you have selected. FlexWord is not supported on Avaya IR.

To create a new word list:

1. Open the Flexwords Manager.
2. Right-click the Flexwords Manager somewhere other than on the name bar.
3. Select New... from the menu.
4. Select FlexWord Grammar...
IVR Designer displays a dialog box and asks you to name the new word list.
5. Enter a name for the FlexWord list in the space provided.



Note:

When naming FlexWord lists, keep in mind that the name should be unique in the first 14 characters. If the first 14 characters are not unique, any FlexWord list on the target system with a matching name will be overwritten during installation of the application.

6. Click OK.
IVR Designer displays the FlexWord Editor window.
7. Enter the words you want in the FlexWord list in the Word List field.
8. When you are finished adding words, click OK.
IVR Designer displays the new FlexWord list in the FlexWords Manager.

Creating a Natural Language Speech Recognition (NLSR) word list

To create a NLSR word list (grammar):

1. Open the Flexwords Manager.
2. Right-click the Flexwords Manager somewhere other than on the name bar.
3. Select New... from the menu.
4. Select NLSR Grammar...
IVR Designer displays a dialog box and asks you for a name for the new word list.
5. Enter a name for the NLSR word list in the space provided.
6. Click OK.
The Natural Language Grammars dialog box displays.
7. Select a recognition resource type from the list.



Note:

If the target system for the application is CONVERSANT, select either OPSR4 or LASR. If the target system for the application is Avaya Interactive Response, select any recognition resource type except LASR. The association between a

recognition resource type (OPSR4 through OPSR9) and a specific recognition resource is administered on the target Avaya IR system.

8. Click OK.
The FlexWord Editor displays the selected recognition resource.
9. Click OK.
IVR Designer displays the new NLSR word list in the FlexWords Manager.

Creating an NLSR word list for a TDD application

IVR Designer can be used to create applications that use the Telecommunications Device for the Deaf (TDD) Modem feature to allow communications with the hearing-impaired.

To create a NLSR word list (grammar) for a TDD application:

1. Open the Flexwords Manager.
2. Right-click the Flexwords Manager somewhere other than on the name bar.
3. Select New... from the menu.
4. Select NLSR Grammar...
IVR Designer displays a dialog box and asks you for a name for the new word list.
5. Enter a name for the NLSR word list in the space provided.
6. Click OK.
The Natural Language Grammars dialog box displays.
7. Select Open Speech Recognition [External Vendor - OPSR8] as the recognition resource type.



Note:

The association between a recognition resource type and a specific recognition resource is administered on the target Avaya IR system.

8. Click OK.
The FlexWord Editor displays the selected recognition resource.
9. Click OK.
IVR Designer displays the new NLSR word list in the FlexWords Manager.

Creating stub grammars

In order to call an NLSR grammar from within a IVR Designer application, you must first create a stub grammar. This stub grammar allows the application to make sure that caller input is sent to

the correct NLSR engine and that the correct NLSR grammar is used by the Prompt and Collect action calling it.

Creating a stub grammar in IVR Designer

To create a stub grammar for a IVR Designer application, do the following from within the IVR Designer application:

1. Open the FlexWords Manager.
2. Right click the FlexWords Manager.
3. From the popup menu, select New, then NLSR grammar...
4. When prompted, enter the name of the NLSR grammar and select OK.
IVR Designer displays the Natural Language Grammars window.
5. Select Open Speech Recognition (External Vendor - OPSR4)
6. Select OK.
IVR Designer displays the Natural Language Grammar Editor window. This window displays the name of the grammar and the grammar type in non-editable fields.
7. Select OK.
The Natural Language Grammar is added to the FlexWords Manager list of FlexWord grammars. During Code Generation (assuming you have used the grammar in your application), the appropriate grammar file is generated.

Later, when you transfer and install the application to the target system, IVR Designer directs the system to create the appropriate stub grammar automatically, as part of the application transfer and installation processes.
8. If the stub grammar is for any existing application, re-verify and re-install all existing applications on the target system.

Renaming a Word List

To rename a word list:

1. Open the Flexwords Manager.
2. Right-click the name of the FlexWord list you want to rename.
3. Select Rename... from the menu.
IVR Designer displays a dialog box and prompts you for a new name for the word list.
4. Enter the new name you want for the word list in the space provided.
5. Click OK.
The name of the FlexWord list changes in the FlexWords Manager window.

Editing a word list

To edit a word list:

1. Open the Flexwords Manager.
2. Right-click the name of the FlexWord list you want to edit.
3. Select Edit FlexWord Grammar... from the menu.
IVR Designer displays the FlexWord Editor window.
4. Edit your FlexWord list in the Word List field.
5. When you are finished editing the word list, click OK.
The edited FlexWord list appears in the FlexWords Manager window.

Right-click menu options for the FlexWords Manager

The following table describes the options available from the FlexWords Manager right-click menu.

Menu option	Description
New...	Lets you create a new FlexWord list.
Flexword Grammar...	An option under New.. that brings up the Flexword editor.
NLSR Grammar...	An option under New.. that brings up the Natural Language Grammar Window, where you select the recognition resource type of your choice, which then brings up the Natural Language Grammar Editor.
Delete	Deletes the FlexWord list from the FlexWords Manager, if the list is not in use in the application.  Note: If you have used the FlexWord list in your application, you must first remove its use from the application before you can delete it.
Rename...	Lets you rename the selected FlexWord list.
Find Uses of	Opens the Find window, searches for Uses Of the selected FlexWord list, and presents the results of the search in the Find window results field.

Sharing...	Lets you designate a FlexWord list as "Shared", so that you or other developers can use it in other applications. When a FlexWord list is marked as "Shared", it can be imported into other applications using the Application Resources tool.
Stay on Top	(Available only when the FlexWords Manager is undocked) Toggles on and off. When checked (toggled on), it causes the FlexWords Manager window to remain on top of other open IVR Designer windows.  Note: 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 in. All windows toggled to "Stay on Top", however, will stay on top of any other IVR Designer windows that are open.
Edit FlexWord Grammar...	Lets you edit the FlexWord list.

Working with external functions

External functions provide an interface for cases that require special handling in an application. In such cases, you can either import a predefined (standard) external function node provided with IVR Designer using the Application Resources tool, or create your own external function node to do what you need. For more information about standard external functions, see [Standard external functions](#). For practice using external functions, see [Using external functions in TAS applications](#) on page 456.

External functions are grouped together and accessed through the External Functions Manager.

The External Functions Manager

Most of the actions you can perform with external functions are done from the External Functions Manager.

The External Functions Manager displays the names of and brief descriptions for all external functions defined for the current application.

The actual external function resides on the target voice response system, not in IVR Designer. To make use of an external function residing on the target system, you must create (or import) an "external function node" to act as an interface with the external function itself. In Help,

however, the external function nodes are referred to simply as "external functions." As part of the CONVERSANT systems, Avaya IR, and IVR Designer packages, Avaya has made available a set of commonly used standard external functions for your use. These, in effect, are "shared-out" by Avaya so that you can import them into ("share them in") your application. If you want to create your own custom external functions, you can use the External Function Source Editor to create them, or you can create them directly on the target system. Once the external function resides on the target system, you must know the name of that external function before you can create the external function node in IVR Designer to call it.

Opening the External Functions Manager

To open the External Functions Manager, do one of the following:

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

External Functions Manager window elements

In the External Functions Manager window, the left column lists in alphabetic order, the names of all external functions defined for the current application. The right column displays a brief description of each external function. This description can only be typed in using the External Function Editor.

Related topics:

[Importing external functions](#) on page 137

[Creating new external functions](#) on page 138

[Renaming an external function](#) on page 139

[Editing an external function](#) on page 140

[Right-click menu options for the External Functions Manager](#) on page 140

[The External Function Editor](#) on page 141

[Setting the external function parameters](#) on page 143

[Setting the function arguments attributes](#) on page 144

[Setting the Function Return Values attributes \(optional\)](#) on page 145

[Defining the external function's appearance](#) on page 146

[Right-click menu options for the External Function Editor](#) on page 147

[The External Function Source Editor](#) on page 148

[Right-click menu options for the External Function Source Editor](#) on page 149

Importing external functions

The IVR Designer package includes a variety of standard external functions. You may often want to use these external functions or others that you or other developers have created. IVR Designer lets you import such external functions from outside the current application. This can save you the trouble of having to create new external functions each time you want to use one in your application.

To import an external function:

1. Open the Application Resources tool.
2. Select External Functions from the Application Resources Options menu.
3. Select the Drive to import from and the external function you want to import.
4. Select one of the following:
 - Create Private Copy
 - Share
5. Do one of the following:
 - Highlight the line entry for the external function you want and click the Import button.
 - Double-click the line entry for the external function you want.
IVR Designer imports the selected external function and displays it as an entry in the External Functions Manager.

Creating new external functions

When no external functions exist that perform the action you need, you can use IVR Designer to create new external functions for use in applications. For more information about creating custom external functions, see [Creating custom external functions](#) on page 476.

To create a custom external function:

1. Open the External Functions Manager.
2. Right-click the External Functions Manager somewhere other than on the name bar.
3. Select New... from the menu.
IVR Designer displays a dialog box and prompts you for a name for the new external function.
4. Enter the name you want for the external function in the space provided.
When naming external functions, keep the following in mind:
 - There is no limit on the length for external function names, but you should keep them as short as possible.
 - Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the external function name.
 - External function names are case sensitive. IVR Designer preserves cases as you assign them when naming the external function. 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_External_Function as the external function name, IVR Designer converts it to New_External_Function when creating the external function. 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. If, for example, you have an external function

called `New_External_Function`, you cannot also create an external function called `NEW_EXTERNAL_FUNCTION`.

- External function names must begin with an alphanumeric character. IVR Designer ignores and deletes all other initial characters when creating the external function name. You may use the underscore (`_`) symbol after the initial character of a external function name.
5. Select OK.
IVR Designer displays the External Function Editor window.
 6. Use the External Function Editor to define the arguments for the new external function.



Note:

There is no limit to the number of arguments an external function may contain.

7. Select OK.
The External Function Editor closes and the new external function name appears as an entry in the External Functions Manager.

Renaming an external function



Note:

If you want to rename an external function defined as a "system function" (such as the standard external functions that come packaged with IVR Designer), you must first clear the "System Function" checkbox in the External Function Editor. For details about defining external functions as "system functions," see [The External Function Editor](#) on page 141.

To rename an external function:

1. Open the External Functions Manager.
2. Right-click the name of the external function you want to rename.
3. Select Rename... from the menu.
IVR Designer displays a dialog box and prompts you for a new name for the external function.
4. Enter the new name you want for the external function in the space provided. Keep in mind the external function name conventions.
When naming external functions, keep the following in mind:
 - There is no limit on the length for external function names, but as a matter of practicality, you should keep them as short as possible.
 - Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the external function name.
 - External function names are case sensitive. IVR Designer preserves cases as you assign them when naming the external function. 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_External_Function as the external function name, IVR Designer converts it to New_External_Function when creating the external function. 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. If, for example, you have an external function called New_External_Function, you cannot also create an external function called NEW_EXTERNAL_FUNCTION.

- External function names must begin with an alphanumeric character. IVR Designer ignores and deletes all other initial characters when creating the external function name. You may use the underscore (_) symbol after the initial character of a external function name.
5. Click OK.
The name of the external function changes in the External Functions Manager window.

Editing an external function

To edit an external function:

1. Open the External Functions Manager.
2. Right-click the name of the external function you want to edit.
3. Select Edit External Function from the menu.
IVR Designer displays the External Function Editor window.
4. Edit the External Function Definition as needed.
5. Click OK.

Right-click menu options for the External Functions Manager

The following table describes right-click menu options for the External Functions Manager.

Menu option	Description
New...	Lets you create a new external function.
Delete	Deletes the external function definition from the External Functions Manager, if it is not in use in the application.
Rename...	Lets you rename the selected external function without altering its definition.

Find Uses of	Opens the Find window, searches for Uses Of the selected external function definition, and presents the results of the search in the Find window results field.
Sharing...	Lets you designate a custom external function as Shared, so that you or other developers can use it in other applications. When an external function is marked as Shared, it can be imported into other applications using the Application Resources tool.
Stay on Top	(Available only when the External Functions Manager is undocked) Toggles on and off. When checked (toggled on), it causes the External Functions Manager window to remain on top of other open IVR Designer windows. 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 in. All windows toggled to "Stay on Top", however, will stay on top of any other IVR Designer windows that are open.
Edit External Function Definition	Lets you edit the selected external function definition.

The External Function Editor

Using the External Function Editor, you can define and edit the attributes of any external function defined for the application.

From the External Function Editor, you can:

- Type a brief description of the external function for display in the External Functions Manager.
- Name or rename the source file for the external function.
- Define the external function as a system function.
- Set the Function Arguments attributes.

- Set the Function Return Values attributes and define custom return branches.
- Define the external function's appearance on the External Functions palette.
- Use a right-click menu to perform various operations in the External Function Editor.

Opening the External Functions Editor

The External Function Editor is accessed through the External Functions Manager.

To open the External Function Editor:

1. Open the External Functions Manager.
2. Do one of the following:
 - Create and name a new external function.
 - Right-click the External Functions Manager window on the name of the external function you want to edit, and then select Edit External Function Definition... from the menu.
 - Double-click the name of the external function you want to edit.

IVR Designer displays the External Function Editor window.

External Functions Editor elements

The following table describes the elements of the External Function Editor.

Element	Description
Function Name	Displays the name of the external function being defined or edited.
System Function checkbox	When checked, it defines the external function as a system function. For more information, see Setting the external function parameters on page 143.
Description field	(Optional) Type a brief description of the external function for display in the External Functions Manager.
Source File Name	Displays the filename of the source file for the external function residing on the target voice response system. Typically, this is the same as the external function name with a .t extension.
Function Arguments box	(Optional) Lets you define the function arguments for the external function. For more information, see Setting the function arguments attributes on page 144.
Function Return Values box	(Optional) Lets you define function return values for the external function.

	For more information, see Setting the Function Return Values attributes (optional) on page 145.
Palette Appearance box	Lets you select a bitmap image for use on the External Functions Palette and assign the external function to a group or tab on that palette.

Setting the external function parameters

The External Function Parameters section includes fields for identifying the external function and controlling its attributes.

System Function Status checkbox

When checked, this parameter causes the external function to be defined as a system function. When an external function is defined as a system function, you cannot edit most attributes of the external function. This includes the function name, description, arguments, and most return value attributes.

The predefined external functions provided with IVR Designer are all defined as system functions.

To customize system functions, you must clear the System Function Status checkbox. You should then rename the external function.

Defining an external function as a system function after you have defined it in the External Function Editor is a good way to protect the external function definition from inadvertently being overwritten on the PC.

Description field

This parameter displays a brief description of the external function. Include what the external function is used for, what arguments it takes, and so forth. This description is displayed in the External Functions Manager Description field. This description can be typed in and edited only in the External Function Editor.

Source File Name field

This parameter displays the filename of the source file for the external function residing on the target voice response system.

Typically, this is the same as the external function name with a .t extension. The external function must have the .t extension to be recognized by a target voice response system.

This source file name is automatically assigned whenever you create or import an external function and in most cases should not be changed.

Setting the function arguments attributes

Function arguments determine how the target system uses the external function in the application. There is no limit to the number of arguments an external function may contain.

Argument field

Type the name of the argument as you want it to be display in the Node Inspector window.

When naming external function arguments, keep the following in mind:

- The only purpose for names is for display in the Node Inspector window. Names can be any length. However, keep argument names as short as possible.
- Do not use spaces in external function argument names. IVR Designer strips out spaces when creating the external function argument.
- Argument names are case sensitive. IVR Designer preserves cases as you assign them when naming the argument.

Type field

From the drop-down menu, select which type you want to assign to the argument. The following table describes the Type options.

Type	Description
Character	Accepts as input all standard ASCII characters, including special characters and punctuation. The extended ASCII character set is not supported.
Currency	Accepts as input numerals (0-9), dollar signs, minus signs, and decimal points.
Date	Accepts as input numerals (0-9), minus signs, slashes, and spaces.
Number	Accepts as input numerals (0-9) and minus signs.
Real	Accepts as input numerals (0-9), minus signs, and decimal points.

Time	Accepts as input numerals (0-9) and the letters A, P, and M.

Direction field

The Direction field determines which way the external function will direct the argument.

Select from the drop-down menu one of the following options:

- To Func – The argument is passed to the external function.
- From Func – The argument is received from the external function.
- To/From Func – The argument is either passed to or received from the external function or both.

Help Message field (optional)

Type a message to be displayed as a pop-up "balloon" message when the cursor rests over the argument in the Node Inspector window.

When writing pop-up Help messages, keep the following in mind:

- Do not use commas in your message. IVR Designer strips out all commas in pop-up Help messages.
- Keep your messages as short as possible. When a message pops up, it only displays for approximately 2.5 seconds. If your message is too long, the reader will not have time to read it before it disappears.

Required field

When set to True, this field defines this argument as a required argument. This means that a variable or literal value must be assigned to it in the Node Inspector window whenever you use this external function in an application. You cannot leave it blank.

Setting the Function Return Values attributes (optional)

IVR Designer lets you create new branches to take when specified values are returned by the external function. Use the Function Return Values area to define those branches and return values.

If you do not define custom branches in this area, IVR Designer sends returns to the default branch.

Return Code Description field

Type a brief description of the return code values.

This description appears as a pop-up "balloon" Help message whenever the cursor rests over the Put Return Code In cell of the Node Inspector window for the external function.

Branch field

The Branch, Operator, and Value fields are used in conjunction with each other. Type the name you want to assign to the branch to be taken if the specified value is returned by the external

function. When assigned, this creates a new branch on the external function node with the name entered here.

Operator field

The Branch, Operator, and Value fields are used in conjunction with each other. Select from the drop-down menu the operator to be used in evaluating values returned from the external function.

Value field

The Branch, Operator, and Value fields are used in conjunction with each other. Type a specific value that the external function may return.

Help Message field

This field is used only to document the use of the return values, and is viewed only in the External Function Editor. It does not appear anywhere else, such as in a pop-up Help message.

Go to Fail branch... checkbox

When the Go to Fail... checkbox is checked, the application goes automatically to the Fail branch whenever a return value is less than zero.

Defining the external function's appearance

In the Palette Appearance box, you can specify how the external function displays on the External Functions Palette.

Bitmap field

Click the  icon to specify a file containing a 20 x 20 pixel bitmap to be used as the icon for this external function.

Each standard external function included in the IVR Designer package has its own icon, which appears on the External Functions palette when you import the external function into your application.

The  icon is the default bitmap icon for External Functions and will be used if no other bitmap image is selected.

Bitmap icons should be exactly 20 x 20 pixels. If you attempt to use larger or smaller bitmaps, IVR Designer resizes the bitmap file to fit in a 20 x 20 pixel space.

Group/Tab Name field

Use the drop-down menu to assign the external function to a group/tab on the External Functions palette.

This is the only place in IVR Designer you can add new groups/tabs for external functions or assign an external function to a different group/tab.

To create a new group/tab for example, because the external function does not logically fit within any defined group/tab, clear the field and type the name for a new group/tab. The new group/tab appears as a separate group/tab on the External Functions palette, and you can further configure the group/tab using the Configure External Functions Palette... option.

To assign an external function to a different group/tab, select the name of a group/tab from the drop-down menu in the Palette Appearance area of the External Function Editor, or create a new group/tab.

You cannot change the name of an existing group/tab once it has been created. You cannot delete a group/tab unless you first delete or move all external functions currently residing in that group/tab. Once you delete or move all external functions from a group/tab, the group/tab is automatically deleted. You cannot delete the default External Functions group/tab.

Right-click menu options for the External Function Editor

The External Function Editor has two different right-click menus. The menu that pops up depends on which area of the External Function Editor you are in.

Right-click menu for fields

The following table summarizes the options on the right-click menu for fields. This is the menu that appears when you right-click the Function Name, Description, Source File Name, or Group/Tab Name fields.

Menu option	Description
Undo	Undoes the last action done in the selected field.
Cut	Removes highlighted text and places it on the clipboard.
Copy	Makes a copy of highlighted text and places it on the clipboard.
Paste	Pastes whatever is on the clipboard at the selected location.
Delete	Removes highlighted text without placing it on the clipboard.
Select All	Highlights the entire contents of the selected field. This option is active only if you have first placed the cursor within the text you want to select.

Right-click menu for grids

The following table summarizes the options on the right-click menu for grids. This is the menu that appears when you right-click either the Function Arguments or the Function Return Values grids.

Menu option	Description

Insert	Inserts a new argument line just above the selected argument, moving everything else down.
Delete	Deletes the selected argument from the grid. This action cannot be undone.
Rename...	Renames the selected argument without altering its attributes.
Add Line	Appends a new argument line at the end of all defined argument lines.
Move Up	Moves the selected argument up one line.
Move Down	Moves the selected argument down one line.

The External Function Source Editor

The External Function Source Editor lets you create an external function for the target system by writing or editing TAS code from within IVR Designer, rather than having to write it directly on the target system using the vi editor or another text editing tool.

Using the External Function Source Editor

To use the External Function Source Editor:

1. From the Code Generation/Application Transfer tool window click the External Function Source Code icon on the Code Generation tab.
IVR Designer displays the External Function Source Editor.

 Note:

Only external functions not defined as system functions are displayed in the External Function Source Editor. Standard external functions and others defined as system functions are not displayed.

2. Right-click the line of the external function you want to create or edit.

 Note:

Before you can create or edit an external function for the target system using the External Function Source Editor, you must first create the external function definition. See [Creating new external functions](#) on page 138.

3. Select Create File (or Edit File) from the right-click menu.

Notepad opens, allowing you to edit or write your own TAS code for the external function. IVR Designer incorporates this code during the Code Generation process and transfers it to the target system using the Application Transfer tool.

Make sure you save your code before closing Notepad, or your code will be lost.

External Function Source Editor column descriptions

The following table describes the columns in the External Function Source Editor.

Column name	Description
Name	Displays the names of external functions defined for the application. Only external functions not defined as system functions are displayed in the External Function Source Editor. Standard external functions and external functions defined as system functions are not displayed.
File	Displays the name of the external function's *.t file.
Used	Displays whether or not the external function is used in the current application.
Last Modified	Displays the date and time the external function was last modified using the External Function Source Editor. If a field in this column is blank, no external function exists in IVR Designer for the definition. The information in this column only reflects modifications made to the external function code using the External Function Source Editor. Modifications made to the external function definition in the External Function Source Editor are not reflected here. External functions already residing on the target system are also not reflected here.

Right-click menu options for the External Function Source Editor

The following table lists and briefly describes the right-click menu options available from the External Function Source Editor.

Option	Description
Create File	(Available only if you have not already used the External Function Source Editor to create external function source code) Lets you create the external function TAS code for the selected external function definition.
Recreate File	(Available only if you have already used the External Function Source Editor to create external function source code) Lets you recreate the external function TAS code for the selected external function definition. When you select this option, IVR Designer deletes any code you have written, before creating the new code. Use this option to "clear the slate" and start new with the external function.
Delete File	Deletes the external function file.
Edit File	Lets you edit existing external function code without deleting the original code.
Refresh	Redraws all columns in the External Function Source Editor in cases where one or more columns are not visible.

Working with prompts

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.

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.

For more information about and practice using prompts, see Prompts, Response, Confirmation, and Validation and [Editing prompts in TAS applications](#) on page 397.

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.

Related topics:

[Creating new prompts](#) on page 151

[Renaming prompts](#) on page 153

[Right-click menu options for the Prompts Manager](#) on page 154

[The Prompt Editor](#) on page 155

Creating new prompts

There are two ways you can create a new prompt in IVR Designer:

- 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.
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 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.

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.

Menu option	Description
New...	Lets you create a new prompt.
Delete	Deletes the selected prompt from the Prompts Manager, if it is not in use. If the prompt is used an application, you must first remove it from the application before you can delete it.
Rename...	Lets you rename an existing prompt.
Find Uses of	Opens the Find window, searches for Uses Of the selected prompt, and presents the results of the search in the Find window results field.

Sharing...	Inactive in the current release of IVR Designer.
Stay on Top	Available only when the Prompts Manager is undocked. Toggles on and off. When checked (toggled on), it causes the Prompts Manager window to remain on top of other open IVR Designer windows. 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 in. All windows toggled to Stay on Top, however, will stay on top of any other IVR Designer windows that are open.
Edit Prompt...	Opens the Prompt Editor so you can edit the selected prompt.

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
- 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
- Choose to use all selected records for a prompt – Automenu node 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.

Related topics:

[Choosing an alternate view for the Prompt Text field](#) on page 156

[Creating a new phrase within a prompt](#) on page 157

[The New Phrase Assignments window](#) on page 157

[Inserting an existing phrase into a prompt](#) on page 159

[Editing a phrase in a prompt](#) on page 160

[Inserting a variable into a prompt](#) on page 160

[Inserting a phrase break](#) on page 161

[Inserting a control into a prompt](#) on page 162

[Deleting a prompt element](#) on page 164

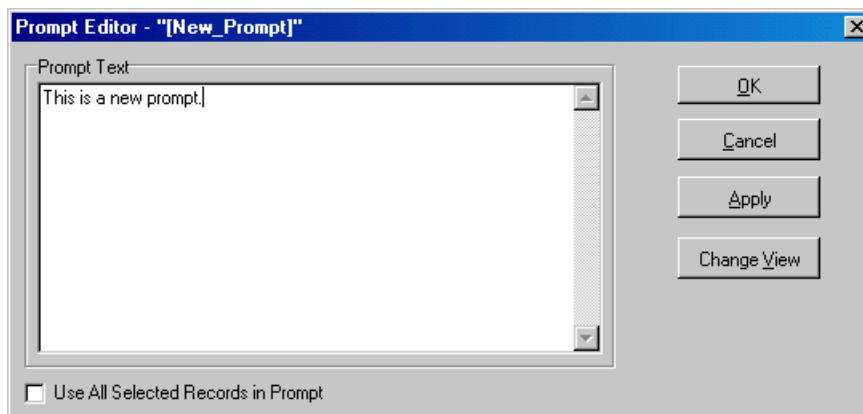
[Inserting a new line \(alternate view\)](#) on page 164

[Right-click menu options for the Prompt Editor](#) on page 164

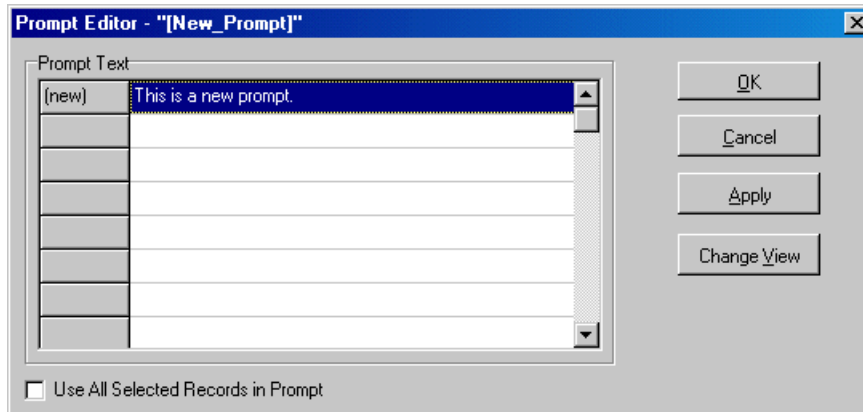
[The Use All Selected Records in Prompt option](#) on page 166

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.

You can also use the alternate view with the Use All Selected Records in Prompt checkbox checked.

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.

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.

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

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 164).

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 164).

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:

- Insert silences of specific lengths
- Adjust the rate at which Text-to-Speech (TTS) speaks the phrase text
- Select risky modes for TTS phrases

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:

- 0.05 sec silence
- 0.1 sec silence
- 0.2 sec silence
- 0.3 sec silence
- 0.4 sec silence
- 0.5 sec silence
- pause (indefinite length)

Text-to-Speech options

Text-to-Speech offers a variety of options for how it may be used. The following table describes the TTS options that are available for the CONVERSANT system. These TTS options are not available for the Avaya IR system.



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.

TTS option	Description
tts	Speaks the text at a normal speed.
tts address	Determines how certain abbreviations, such as "St." or "Dr.", are spoken. Example: "St." is spoken as "street," not "saint." "Dr." is spoken as "drive," not "doctor."

tts fast	Speaks the text at a speed faster than normal.
tts fraction	Determines how numbers with slashes in them are spoken. Example: The number 1/4 is spoken as "one fourth" and not as "one slash four", as it would be, for instance, as part of a date.
tts normal	Speaks the text at a normal speed.
tts phone number	Determines how numbers are spoken. Numbers are spoken as individual digits, except for "800" which, because of its common use in toll-free calls is spoken as "eight hundred" when used in the area code positions. Example: The number 5551368 is spoken as "five five five (pause) one three six eight" and not as "five million (pause) five hundred fifty-one thousand (pause) three hundred sixty-eight"
tts proper name	Determines how certain abbreviations, such as "St." or "Dr.", are spoken. Example: "St." is spoken as "saint," not "street." "Dr." is spoken as "doctor," not "drive."
tts slow	Speaks the text at a speed slower than normal.

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... from the menu.
 - 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.
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.

Menu option	Description
Cut	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.
Copy	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.
Paste	Pastes whatever is on the clipboard to the cursor's location.
Delete	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.
Select All	Selects or highlights all elements in the Prompt Text field.
Insert Variable...	Inserts a variable at the cursor's location.
Insert Phrase...	Inserts a phrase at the cursor's location.
Insert Phrase Break	Inserts a phrase break at the cursor's location.
Insert Control	Inserts a control at the cursor's location.

Alternate view

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

Menu option	Description
Insert Line	Inserts a new line just above the currently selected (highlighted) line.
Add Line	Adds a line at the end of the element list.
Delete Line	Deletes the currently selected (highlighted) line. The Delete action cannot be undone and does not copy the element to the system clipboard.

Move Up	Moves the currently selected line up one line, exchanging places with the line above it.
Move Down	Moves the currently selected line down one line, exchanging places with the line below it.

The Use All Selected Records in Prompt option

The Use All Selected Records in Prompt option is typically used only with the Automenu node. When selected, this option lets you:

- Select the prompt components to be played as an Introduction.
- Select the prompt components to be played as the Body.
- Select the prompt for playing More Records.
- Set the Loop Control Options for playing More Records.
- Select the prompt components to be played as the Conclusion.

Normal view

The following figure shows the normal view of the Prompt Editor with the Use All Selected Records in Prompt option checked.

The screenshot shows the 'Prompt Editor - "[New_Prompt]"' dialog box. It has four main text input areas on the left: 'Introduction', 'Body', 'More Records', and 'Conclusion'. The 'More Records' area contains the text 'To hear more options, press the pound key.' On the right side, there are four buttons: 'OK', 'Cancel', 'Apply', and 'Change View'. Below these buttons is a 'Loop Control Options' section with three dropdown menus: 'Record Counter' (set to '[RecordCounter]'), 'Max Records Presented' (set to '(all)'), and 'More Records Touchtone' (set to '#'). At the bottom left, there is a checked checkbox labeled 'Use All Selected Records in Prompt'.

Alternate view

The following figure shows the alternate view of the Prompt Editor with the Use All Selected Records in Prompt option checked.

This screenshot shows the same 'Prompt Editor' dialog box but in an 'alternate view'. The text input areas on the left now have a table-like structure with a selection column on the left and a text column on the right. In the 'Introduction' section, the first row is selected (highlighted in blue). In the 'Body' section, the 'loopstart' row is selected with '(all)' in the text field, and the 'loopend' row contains '[RecordCounter]'. In the 'More Records' section, the '[new]' row is selected with 'To hear more options, press the pound key.' in the text field, and the 'moreEnd' row is empty. The 'Conclusion' section has its first row selected. The right side of the dialog, including the buttons and the 'Loop Control Options' section, remains the same as in the first screenshot. The 'Use All Selected Records in Prompt' checkbox at the bottom left is still checked.

Introduction prompt components

When the application first begins to execute the Automenu node, the Introduction plays once.

As with any other prompt, the Introduction can consist of any combination of prompt phrases, variables, breaks, and controls.

Body prompt components

After the Introduction plays, the prompt plays the Body components. The Body typically consists of those phrases and variables that you want to repeat for each record found.

As with any other prompt, the Body can consist of any combination of prompt phrases, variables, breaks, and controls.

More Records prompt components

If the Code Generation Options Maximum number of records returned per Database query field is set to retrieve more than the number of records set in the Max Records Presented field, and the application finds more than the number of records set in the Max Records Presented field, the More Records prompt plays after the Body prompt. The More Records prompt is typically a phrase used to tell the caller which touchtone key to press to hear more menu options or records.



Note:

The key for the caller to press is defined in Loop Control Options (for more information, see Setting the Loop Control options).

If the application finds fewer than the number of records set in the Max Records Presented field, the More Records prompt does not play. Also, if the Code Generation Options Maximum number of records returned per Database query field is set to retrieve fewer than the number of records set in the Max Records Presented field, that determines the number of records that are retrieved, and the More Records prompt does not play.

More Records examples

- If you have set the Max Records Presented field to (all), and the Code Generation Options Maximum number of records returned per Database query field is set to 30, the caller is presented 9 records, and then the More Records prompt plays. If the caller presses the key defined in the More Records Touchtone field, the next 9 records are presented and the More Records prompt plays again.
- If you have set the Max Records Presented field to 7, and the application finds only 4 records, only those 4 records are presented to the caller, and the More Records prompt does not play.
- If you have set the Max Records Presented field to 8, but the Code Generation Options Maximum number of records returned per Database query field is set to 5, only 5

records are retrieved and presented to the caller, and the More Records prompt does not play.

Setting the Loop Control options

The following Loop Control Options are used only in conjunction with the More Records prompt components:

- **Record Counter** – Select from the drop-down menu the variable to use as a record counter for presenting the records to the caller. A maximum of nine records can be presented at any one time.
- **Max Records Presented** – Type the number, or select from the drop-down menu, the variable that determines the number of records to make available to the caller. The default is (all), which allows all records found to be presented, 9 at a time.



Note:

If you have the Max Records Presented field set to 5, for example, and the Code Generation Options Maximum number of records returned per Database query field is set to retrieve more than 5 records, the caller has the option to be presented with all records retrieved, 5 at a time.

- **More Records Touchtone** – Type the character key or select from the drop-down menu the variable that contains the character key to be used by the caller to hear more records. The default is the # key.

Conclusion prompt components

After the other prompt components have all finished playing, the Conclusion plays once. This is usually a simple phrase instructing the caller to make a choice and press the corresponding touchtone key.

As with any other prompt, the Conclusion can consist of any combination of prompt phrases, variables, breaks, and controls.

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 two phrase tables with each application:

- The StandardSpeech table, based on the Enhanced Basic Speech language selected for the application
- A custom phrase table, labeled simply PhraseTable

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

The phrases of only one phrase table at a time are displayed in the Phrases Manager.

For practice using phrases, see [Using variables and speech in TAS applications](#) on page 385.

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 [Creating phrases in TAS applications](#) on page 395.

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.

Column heading	Description
Phrase number	Displays the phrase numbers of all phrases defined for the current application.
Recorded	An X in this column indicates that an audio file exists for the corresponding phrase (on the PC).
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.

**Note:**

The phrase text must be entered as single-byte text. Double-byte text cannot be used in TAS applications.

This text is displayed during Simulation.

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.

Related topics:

[Importing phrase tables](#) on page 171

[Creating new phrases](#) on page 172

[Editing existing phrases](#) on page 172

[Right-click menu Options for the Phrases Manager](#) on page 173

[The Phrase Editor](#) on page 175

[The Phrase Table Editor](#) on page 179

Importing phrase tables

IVR Designer lets you import phrase tables for example, from other applications or phrase tables that have been shared by other developers, using the Application Resources tool. This feature is helpful when you want to reuse phrases that have been created elsewhere.

To import a phrase table into your application:

1. Open the Application Resources tool.
2. Click the Import tab at the bottom of the window.
3. (Optional) From the drop-down Drive menu, select the mapped network drive containing available resources.
4. Select Phrase Table from the Application Resources drop-down menu at the top of the window.
IVR Designer displays a list of available phrase tables and their locations.

Enhanced Basic (Standard) Speech is imported automatically, based upon the language selected for the application, and cannot be deleted.

5. Select whether you want to import the resource as Shared or as a Private Copy by clicking the appropriate radio button.
6. Do one of the following:
 - Highlight the line entry for the phrase table you want, then click the Import button.
 - Double-click the line entry for the phrase table you want.
IVR Designer imports the selected phrase table and makes it the active phrase table in the Phrases Manager window.

Creating new phrases

Most of the time, you will 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... from the menu.
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.
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 want to edit.
 - Right-click the row containing the phrase you want to edit, and then select Edit Phrase... from the menu.
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.

Option	Description
New...	Creates a new phrase, automatically assigns a phrase number to it, and opens the Phrase Editor window.
Delete	Deletes the contents of the selected phrase from the table (both the phrase text and, if one exists, the *.wav file, but not the phrase number).
Rename...	Unavailable option in the Phrases Manager.
Find Uses of	Opens the Find window, searches for Uses Of the selected phrase, and presents the results of the search in the Find window results field.
Stay on Top	(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 "Stay on Top", the window actually on top will be the last window you clicked in. All windows toggled to "Stay on Top", however, will stay on top of any other IVR Designer windows that are open.
Edit Phrase...	Opens the Phrase Editor so you can edit the selected phrase.
Display Alphabetically by Tag	Toggles on and off. When checked (toggled on), it lists all phrases in the table alphabetically by phrase tags.
Fill Phrases with Default...	Opens the Create Default Phrases dialog box, which lets you create a user-specified number of new phrases. It also fills each new phrase with default phrase text.
Import Phrases from Text...	Opens the Phrase Text Import dialog box, which lets you select and import predefined text from sources outside IVR Designer.
New Phrase Table...	Opens the Phrase Table Editor, which lets you create a new phrase table and define its basic parameters.
Edit Phrase Table...	Opens the Phrase Table Editor and populates it with information about the currently open phrase table, which you can then edit.
Delete Phrase Table...	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.  Note: The Delete Phrase Table action cannot be undone.
[Phrase Table]	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.

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.

Related topics:

[Phrase Editor window elements](#) on page 175

[Using the Phrase Editor](#) on page 177

[Audio recording tips](#) on page 178

Phrase Editor window elements

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

Element	Description
Phrase Text field	Type or import the text for the phrase you are creating in this field. Text in this field can be edited as with any basic word processor. Text created in this field is displayed 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.
Stop button	Stops a recording or playback in progress and resets the recording to its beginning.
Record button	Digitally records a phrase.
Import button	Imports a prerecorded audio file from another source. Audio files must be in *.wav format to play in IVR Designer. Other audio formats are not currently supported.  Note: The code generation process converts the audio files to either ADPCM32 or G.711 files (depending on the target platform) sampled at 8 kHz. As a result, *.wav files imported into the application that are recorded at higher frequencies might result in a loss of quality after the code generation process. Also, if the imported wav file is in G.711 (CCITT 8 KHz mu-law) format, it will not undergo any conversion during the code generation process.
Edit button	Edit a recorded phrase. Clicking on this button opens the default audio editing application on your machine. This is typically the Microsoft Sound Recorder, although you might have specified another application. This button is active only when there is a recorded phrase present, either as part of the application or in the clipboard buffer.

Erase button	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.
Out of Date checkbox	When checked, this signifies that changes have been made to the written text since a recording was made for the phrase. If no recording was made, this box is checked by default. You can override the "Outdated" status.
Length box	Displays the running time of the recording to the nearest thousandth of a second.
Size box	Displays the size of the recording in numbers of bytes.
Prev button	Selects the previous phrase in the phrase table for editing without having to close and reopen the Phrase Editor.
Next button	Selects the next phrase in the phrase table for editing without having to close and reopen the Phrase Editor.
OK button	Accepts the edits made to the selected phrases and makes the changes in the application.
Cancel button	Cancels any changes made to the phrase and closes the Phrase Editor.

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. 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.

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.

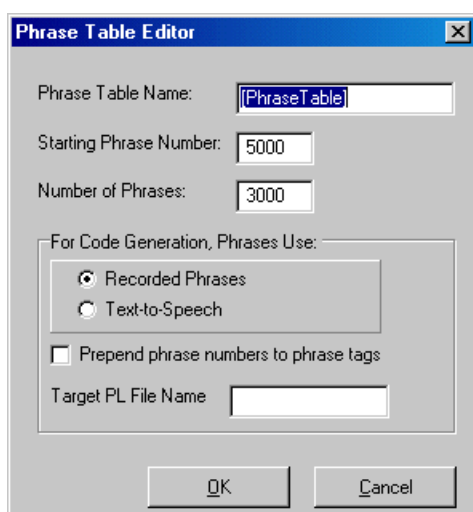
The Phrase Table Editor

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.



Window elements

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

Element	Description
Phrase Table Name field	Type a new phrase table name here or check 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.  Note: Do not put square brackets ([]) around a new name. IVR Designer automatically inserts the brackets when it creates a new table.

Starting Phrase Number field	Enter a starting phrase number for this phrase table. By default, IVR Designer assigns the first unallocated free phrase number.
Number of Phrases field	Enter the number of phrases you want to allocate for this phrase table.
For Code Generation, Phrases Use field	Select whether phrases should be generated for playback on the target system using recorded speech or Text-to-Speech.

Opening the Phrase Table Editor window

To open the Phrase Table Editor window:

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.
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 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.

Related topics:

[Importing text](#) on page 181

[Using the Phrase Text Import dialog box](#) on page 182

[Using the Create Default Phrases dialog box](#) on page 184

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... 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:
 - 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.
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 hosts

Some applications are designed to exchange data with a 3270 IBM-type mainframe host remote to the target voice response system. Before the two machines can exchange data, they must be physically connected, and the interface protocols must be established. To accomplish this, you must define the interface for your target system and the host.

For more information about and practice creating host applications, see [Creating host applications](#) on page 496.

To define a host interface:

1. Identify the host you want to establish a connection and exchange data with. This also requires that you know the screens you want to access on that host, because the connection is established on the target system.
2. Import or create a host definition in IVR Designer.
3. Set up a host flow to define the login, logout, and recovery processes.

Host definitions are imported, created, and edited using the Hosts Manager.

The Hosts Manager

The Hosts Manager displays the names of all hosts defined for the current application. From the Hosts Manager, you can:

- Display host definitions imported using the Application Resources tool
- Create new host definitions
- Rename host definitions
- Edit a host definition
- Define a host flow
- Use the right-click menu in the Hosts Manager to perform other operations with host definitions

The Hosts Manager displays all hosts defined for the current application, listing the hosts in alphabetic order. The Hosts Manager also displays the following information from the Host Definition Editor:

- Host Parameters
- Screen Definitions
- Login/Password Pairs
- Screen Identifications
- Field Definitions

Only headings are displayed in the Hosts Manager. You cannot view or edit the contents of any of these fields directly from the Hosts Manager. Instead, you must use the Host Definition Editor to view and edit host definitions.

Related topics:

[Importing Host Definitions](#) on page 186

[Creating new Host Definitions](#) on page 187

[Renaming a Host Definition](#) on page 188

[Editing a Host Definition](#) on page 189

[Right-click menu options for the Hosts Manager](#) on page 189

[The Host Definition Editor](#) on page 190

[Host Flows](#) on page 201

[Capturing host screens](#) on page 203

Importing Host Definitions

You might want to reuse host definitions created for other applications when they exist. IVR Designer lets you import host definitions that have been created for other applications and that have been shared out by other developers. This can save you the trouble of having to create new host definitions each time you want to use one in your application.

To import a host definition:

1. Open the Application Resources tool.
2. Click the Import tab at the bottom of the window.
3. (Optional) From the drop-down Drive menu at the bottom of the window, select the mapped network drive containing available resources.
4. Select Hosts from the Application Resources Options menu.
IVR Designer displays a list of available host definitions and their locations.
5. Select whether you want to import the resource as Shared or as a Private Copy by clicking the appropriate option.
6. Do one of the following:
 - Highlight the line entry for the host definition you want, and then click the Import button.
 - Double-click the line entry for the host definition you want.
IVR Designer imports the selected host definition and displays it as an entry in the Hosts Manager.

Creating new Host Definitions

To create a new host definition:

1. Open the Hosts Manager.
2. Right-click the Hosts Manager somewhere other than on the name bar.
3. Select New... from the menu.
IVR Designer displays a dialog box and prompts you for a name for the new host.
4. Enter the name for the host in the space provided.
When naming hosts, keep the following in mind:
 - Names must be 11 or fewer characters. IVR Designer truncates host names with more characters to the first 11 characters.
 - Do not use spaces in host names. IVR Designer ignores and deletes spaces when creating the actual host name.
 - Host names are case sensitive. IVR Designer preserves cases as you assign them when naming the branch. 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_Host as the host name, IVR Designer converts it to New_Host when creating the host.



Note:

Although IVR Designer preserves case 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 host called New_Host, you cannot also create a host called NEW_HOST.

- You may use any standard ASCII character as part of a host name, but the first letter must be alphanumeric. IVR Designer does not support the use of extended ASCII characters in host definition names.
5. Click OK.
IVR Designer displays the Host Definition Editor window.
 6. Define the timeouts and the screens to use in exchanging data with the host in the Host Definition Editor.
 7. Click OK.
IVR Designer closes the Host Definition Editor and displays the new host definition name as an entry in the Hosts Manager.



Note:

The Screen Mapping Tool lets you to drag and drop the information into the fields from screens that have been captured from the Host machine.

Renaming a Host Definition

To rename a host definition:

1. Open the Hosts Manager.
2. Right-click the name of the host definition you want to rename.
3. Select Rename... from the menu.
IVR Designer displays a dialog box and prompts you for a new name for the host definition.
4. Enter the new name for the host definition in the space provided.
When naming hosts, keep the following in mind:
 - Names must be 11 or fewer characters. IVR Designer truncates host names with more characters to the first 11 characters.
 - Do not use spaces in host names. IVR Designer ignores and deletes spaces when creating the actual host name.
 - Host names are case sensitive. IVR Designer preserves cases as you assign them when naming the branch. 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_Host as the host name, IVR Designer converts it to New_Host when creating the host.



Note:

Although IVR Designer preserves case 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 host called New_Host, you cannot also create a host called NEW_HOST.

- You may use any standard ASCII character as part of a host name, but the first letter must be alphanumeric. IVR Designer does not support the use of extended ASCII characters in host definition names.

5. Click OK.

IVR Designer changes the name of the host definition in the Hosts Manager window.

Editing a Host Definition

To edit a host definition:

1. Open the Hosts Manager.
2. Right-click the name of the host definition you want to edit.
3. Select Edit Host from the menu.
IVR Designer displays the Host Definition Editor window.
4. Edit the Host Definition as needed.
5. Click OK.

Right-click menu options for the Hosts Manager

The following table describes the right-click menu options for the Hosts Manager.

Menu option	Description
New...	Creates a new host definition.
Delete	Deletes a host definition from the Hosts Manager, if the definition is not in use. If you have used the host definition in your application, you must first remove its use from the application before you can delete it.
Rename...	Renames the selected host definition without altering its contents.
Find Uses of	Opens the Find window, searches for Uses Of the selected host definition, and presents the results of the search in the Find window results field.

Sharing...	Designates a custom host definition as Shared, so that you (or other developers) can use it in other applications. When a host definition is marked as shared, it can be imported into other applications using the Application Resources tool.
Stay on Top	(Available only when the Hosts Manager is undocked) Toggles on and off. When checked (toggled on), it causes the Hosts Manager window to remain on top of other open IVR Designer windows. 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 in. All windows toggled to Stay on Top, however, will stay on top of any other IVR Designer windows that are open.
Copy Host...	Makes a copy of the selected host definition and prompts you for a new name for the copy.
Edit Host	Lets you edit the selected host definition.
Edit Host Flow	Lets you edit the host (call) flow for the selected host definition.

The Host Definition Editor

The Host Definition Editor lets you define all the interface elements needed to enable the target voice-response system to exchange data with the host.

From the Host Definition Editor you can:

- Set host timeout parameters
- Select and define screens to use (on the host)
- Set login and password information
- Specify unique elements to identify each selected screen

- Identify fields for each selected screen to be used for the exchange of data
- Use the right-click menu to perform various operations in the Host Definition Editor

Opening the Host Definition Editor

To open the Host Definition Editor:

1. Open the Hosts Manager.
2. In the Hosts Manager window, right-click the name of the host definition you want to work with.
3. Select Edit Host.
IVR Designer displays the Host Definition Editor.

Related topics:

[The Host Definition tab](#) on page 191

[The Logins tab](#) on page 193

[The Field Definition tab \(optional\)](#) on page 194

[The Helper DIP](#) on page 196

[Right-click menu options for the Host Definition Editor](#) on page 198

[Additional right-click menu options](#) on page 200

The Host Definition tab

Use the Host Definition tab to set timeout parameters and define the screens to be accessed on the host.

Host Parameters area

The Host Parameters area of the Host Definitions tab contains the fields for setting timeout parameters for connecting to the host.

The following table summarizes the options in the Host Parameters area.

Field	Description
Host Timeout	Enter the number of seconds that the system waits for a response from the host after a Host Send. If the host does not respond within this time frame, a Host Timeout condition occurs. The range is 1 through 300 seconds. The default is 45 seconds. If this field is set to 0 (zero), the application waits indefinitely for a response from the host.
Logical Unit Timeout	Enter the number of seconds that the system waits for a Logical Unit (LU) to be assigned to the application. If an LU

	is not available within the time frame specified, a Timeout condition occurs. The range is 1 through 45 seconds. The default is 0 seconds. If this field is set to 0 (zero), the application does not wait for an LU to become available.
Unrecognized Screen Timeout	Enter the number of seconds that the system waits for a specific screen of data from the host. If a screen is received that is not the specified screen, and the time has elapsed, an Unrecognized Screen Timeout condition occurs. The range is 1 through 300 seconds. The default is 45 seconds. If this field is set to 0 (zero), the application waits indefinitely for a response from the host.

Screen Definition area

The Screen Definition area of the Host Definitions tab contains the fields for naming and classifying the screens used in exchanging data with the host.



Note:

Before going to another tab, click the Apply button.

The following table summarizes the options in the Screen Definition area.

Field	Description
Screen Name	Type the name to be assigned to the screen.
Base Type	From the drop-down menu, select the base type for the screen. There are three options: • Login – Typically, the first screen expected from the host • Transaction – The screen you want the system to start with when a call comes in • None – (Default) All other screens.
Blank Screen	From the drop-down menu, select True if this screen is a blank screen. Select False (the default) if it is any other kind of screen.

	You can have only one Blank Screen per host definition.
Rows	Enter the number of rows defined for the screen, normally the default which is 24.
Columns	Enter the number of columns defined for the screen, normally the default which is 80.

The Logins tab

The Logins tab contains the fields for logins and passwords required for logging into the host.



Note:

Before going to another tab, click the Apply button.

The Logins tab has three columns. The following table summarizes the fields on the Logins tab of the Host Definition Editor.

Field	Description
#	Displays the number assigned to each login and password set. A login and password set must be defined for the maximum number of Logical Units (LUs) the host definition will be assigned to. For example, if this host definition will be assigned to 50 LUs, there must be 50 login and password sets defined on this tab. Login and password sets do not have to be unique. If the login and password is the same for each login, Avaya recommends that you leave this tab blank and define the login and password as part of the Host Flow.
Login	Type a login ID that is valid for logging in to the host.
Password	Type a password that is valid for logging in to the host. This must be the password for the Login on the same line. If a password is not required for the Login, leave this field blank.

The Field Definition tab (optional)

Use the Field Definition tab fields to define fields on screens sent to or received from the host.

The Blank Screen: and Base Type: fields reflect the options set in the Screen Definition area of the Host Definition tab.

The following table describes the fields on the Field Definition tab.

Field	Description
Screen Name	From the drop-down menu, select the name of the screen you are targeting on the host. This list is blank until you have defined at least one screen in the Screen Definition area of the Host Definition tab.
Name	Type the name of the field to be defined for this screen.
ID	Automatically assigned, this number indicates the number of occurrences of this field name. Use this number or type another ID number over it. For example, if the field Name "Found_Baggage" appears twice in the Field Definitions tab, the first occurrence is assigned an ID of 1 and the second is assigned an ID of 2.  Tip: This field can be used for repeating lines on a host screen.
Location	From the drop-down menu, select: • Regular if the desired field's location is known • External if the desired field's location is not known
Row	Type the number of the row (1 through 24) within the host screen from which the field originates.

Column	Type the number of the column (1 through 80) within the host screen from which the field originates.
Length	Type the maximum length you want the field to have. When assigning a length to a character field that will be used as part of a host definition, keep in mind that for a single host screen, the total size of all fields defined must be equal to or less than 2000 bytes minus 1 byte of overhead for each host field.
Direction	Select from the drop-down menu the direction that data may travel between the target voice response system and the host. The options are: • To Host – Data may travel only from the target system to the host for the field defined. • From Host – Data may travel only to the target system from the host for the field defined. • Both – Data may travel both directions between the two systems.
Class	Select from the drop-down menu the class of data that may be used for the field defined. The options are: • Character – Includes all standard ASCII alphanumeric characters. • Number – Includes only numbers and the minus symbol. The Number class strips out all dollar signs, decimal points, and commas.

More about the Location field and External fields

For most host definitions, the Location field is set to the default, Regular, setting. This means that you know the location (starting row and column) of the desired field on the host and that it is in a fixed location. When the Location field is set to the default, the Row and Column fields must be defined, so that the application knows where to find the requested field.

In some cases, however, the requested field may not be in a fixed location. In these cases, you cannot know the location of the field, because it may be in a different location each time you need to access it. For these cases, you should set this Location field to External.

When this field is set to External, the Row and Column fields are set to zero (0) and made inactive. Also, the Direction field is set to From Host and made inactive. You must still fill in the Name, ID, Length, and Class fields.

 Note:

In versions 3.0 and earlier of Voice@Work, this field does not exist. If you are upgrading applications created in Voice@Work 3.0 and earlier or in Graphical Designer, this field is, by default, set to Regular for all host definitions.

When the running application encounters an external field, it invokes a "helper DIP", where it expects to find the instructions necessary to locate the field. You, as the application developer, must provide the instructions for this helper DIP.

When working with hosts that use external fields, you may find it helpful to print out the screen containing the external field. This can, for instance, be helpful in debugging an application. To print out the screen in question, you can use the fancy_print subroutine.

The Helper DIP

The helper DIP provides instructions necessary to locate a field defined as an external field in an application. You must modify the existing helper DIP using the C programming language.

The helper DIP file, named extract.c, is located in the /att/ag/hostdip/helper directory on the CONVERSANT or Avaya IR. The helper DIP file already contains the basic structure required to pass field location instructions to an application. What you must provide is the instructions themselves. You can provide instructions to locate as many external fields as necessary.

Editing the Helper DIP

To edit the helper DIP, enter the following at the system prompt:

```
cd /att/ag/hostdip/helper
vi extract.c
```

There are two subroutines in the extract.c file that are called when a screen that uses external fields is received from the host. For every screen that uses external fields, you should modify the new_screen subroutine and the extract subroutine.

The extract.c file provided with IVR Designer is installed whenever the CONVERSANT or Avaya IR system software is reloaded. Reloading the system software overwrites any existing customized extract.c files. If you want to retain a customized extract.c file for later use, first save this file under another name, preferably in a different directory. You can then copy the saved customized file over the generic extract.c file after you have completed the software upgrade procedure.

Using the new_screen subroutine

Using the new_screen subroutine, you may want to check what screen it is and set up some global data structures. You should be aware that the IVR Designer host screen name used at

the TAS level consists of the host name and the screen name with an underscore between them as follows:

host name_screen name

For example, if the host name in IVR Designer is "Hostappl" and the host screen name is "ACCOUNT", the host screen name that should be referenced in the helper DIP is:

hostappl_ACCOUNT

Using the extract subroutine

Use the extract subroutine to check what screen this is, what field is to be extracted, and then to extract the field from the screen and put it into the variable field. You should be aware that the IVR Designer host field name used at the TAS level consists of the host name, screen name, field name, and ID with an underscore between them as follows:

host name_screen name_field name ID

For example, if the host name in IVR Designer is "Hostappl", the host screen name is "ACCOUNT", the host field name is "BALANCE", and the ID is "1", the host field name that should be referenced in the helper DIP is:

hostappl_ACCOUNT_BALANCE1

The extract subroutine is called for every external field defined in every application. If more than one external field is defined within the applications on the system, you should check the parameters of the extract subroutine to see which field should be extracted. The appl_name, screen_name, and fld_name parameters correspond to the names defined within IVR Designer. The field provided by the extract subroutine should not be longer than the field Length size specified within IVR Designer.

Using the fancy_print subroutine

When working with hosts that use external fields, you may find it helpful to print out the screen containing the external field. This can, for instance, be helpful in debugging an application. To print out the screen in question, you can use the fancy_print subroutine, calling it from the new_screen subroutine or the extract subroutine.

When the screen data is printed out, nulls are replaced with blank spaces and attributes with exclamation marks (!) for readability. After 78 characters, db_pr inserts newline characters, so if you try to print more than 78 characters per line, the line is split.

Installing the New Helper DIP

Once you have modified and saved the extract.c file, you must reinstall it on the CONVERSANT or Avaya IR system.

To install the new helper DIP, enter the following at the system prompt:

```
cd /att/ag/hostdip/helper
```

```
stop_vs
```

```
make
```

```
start_vs
```

The installation procedure temporarily halts any other applications running on the system.

Using DIP instance numbers

If you write your own DIP and access it through an external function, you need to know the DIP and the DIP instances reserved by the CONVERSANT or Avaya IR system. The DIP "instance number" must be unique among all instances of this DIP in the system. The instance numbers range from 0 to 34. CONVERSANT or Avaya IR uses 0, 1, 4, and 11 as shown below:

- DIP0, instance 0 – 3270 host DIP
- DIP1, instance 1 – Local database DIP
- DIP4, instance 4 – Recording speech
- DIP11, instance 11 – Helper DIP

Duplicating an instance number within the same DIP number keeps the DIP from operating properly. For example, if you write a DIP and use instance number 0 twice, one instance of your DIP will never get past the call to start up. Whichever instance of the DIP gets started first uses the instance 0 and blocks the other from starting.

Right-click menu options for the Host Definition Editor

The Host Definition Editor has its own right-click menu. The following table describes the menu options.



Note:

The right-click menu options may vary from the list below, according to the active tab.

Menu option	Description
Validate Grid	Checks the data on the selected tab to make sure that all required fields are populated.
Delete Row	Deletes the selected row. This action cannot be undone.
Insert Row	Adds a new row above the selected row.
Add Row	Appends a new row at the end of all screens or fields (rows) defined.
Move Row Up	Moves the selected screen up one row.
Move Row Down	Moves the selected screen down one row.

Copy Screen	This option is available on the Host Definition tab only. It lets you make a copy of the selected screen. When this option is selected, IVR Designer prompts you for a new name for the copy.
Fill Login/Password with default...	This option is available on the Logins tab only. It lets you assign a default login ID and password to any number of sets you want. When this option is selected, IVR Designer displays the Create Default Login window, and prompts you for the default login ID and password and for the number of sets to which you want to assign these defaults. These defaults are assigned to the sets starting with the selected row.
Create Variables	This option is available on the Field Definition tab only. It automatically creates variables for all the fields defined on the Field Definition tab. When selected, this option creates and names a new variable with the ID number appended to it for each field defined, opens the Variables Manager, and places all the newly created variables in the Variables Manager.
Create Variables Interactively	This option is available on the Field Definition tab only. It lets you selectively create variables for fields defined on the Field Definition tab. When selected, IVR Designer prompts you to decide whether you want to create a new variable with the ID number appended to it for each field defined. For each variable you decide to create, IVR Designer opens the Variables Manager and places the newly created variable in the Variables Manager.

IVR Designer does not automatically update variables based on Field Definition entries.

If you create a variable for an entry in the Field Definition tab of the Host Definition Editor and you later decide to change the entry in the Host Definition Editor, you must go back and manually alter the variable as well.

Alternatively, you can choose to recreate the variable and keep both versions of the entry variable or delete the old variable definition.

If you attempt to create a variable from a Field Definition tab entry, and another variable already exists with the same name, no new variable is created, and the original variable definition remains as it was.

Additional right-click menu options

The following table lists and describes the right-click menu options available only from the Host Parameters section of the Host Definition tab.

Menu option	Description
Cut	Cuts the highlighted text from the selected field and saves it to the clipboard.
Copy	Copies the highlighted text without cutting it from the selected field and saves it to the clipboard.
Paste	Pastes whatever is on the clipboard into the selected field. You must first place the cursor in the field where you want to paste. If you do not, the paste function does not work.
Delete	Deletes the highlighted text from the selected field. This action cannot be undone. The highlighted text is not saved to the clipboard.
Select All	Selects all text in the selected field. You must first place the cursor in the field where you want to highlight text.
Insert Variable...	Not functional in the current version.

Host Flows

The primary purpose of the Host Flow is to set up the interactions between the target system and the host. There is one Host Flow for each Host Definition in the application. The only nodes that can be used in the Host Flow are the Host Send node and the Host Send/Receive node.

Host call flows

The following table describes the call flows contained in a Host Flow.

Call flow name	Description
Login	Defines the sequence of screens needed to log in to the host. The goal of this flow is to get from the login screen to the transaction screen, that is, the screen you want the system to be on when a call comes in. The amount of time to wait for a timeout condition to occur is set in the Host Definition Editor. If the system cannot get logged in or get to the transaction screen, the Recoversequence is started.
Logout	Defines the sequence of screens needed to log out from the host. This flow goes from the transaction screen to the login screen. The Logout flow is typically used when the system is coming down for some reason, such as when the voice system is being stopped. The amount of time to wait for a timeout condition to occur is set in the Host Definition Editor.
Recover	Defines the recovery sequence for all screens. The goal of this flow is to recover when an unexpected screen is received, for example, or when a caller hangs up before completing a transaction. In cases like these, the Recover flow attempts to get back either to the transaction screen or the login screen. The amount of time to wait for a timeout condition to occur is set in the Host Definition Editor. If the system cannot recover and get back to either the login or the

	transaction screen, the Fail branch is taken.

Host Flow tips

When working with Host Flows, keep the following in mind:

- The login screen is the screen that your application first uses to establish a working connection with the host.
- The first action step of the Login flow sequence should be to get the host login screen. The last screen of the Login sequence must be a transaction screen.
- The Logout flow sequence should end with the login screen.
- The transaction screen is the screen that the application returns to after a caller transaction. You must specify to the system what screen to start with to get to any other systems. That screen is the transaction screen.
- In many ways, the Recover flow sequence is the most critical of the three host flow sequences. When developing the Recover flow, you must try to include all possible cases, including unrecognized screens and host timeouts. The purpose of the Recover flow is to tell an application how to recover if it detects a screen that it does not expect. Whenever this happens, the system automatically invokes the Recover flow.
- The Recover flow is initiated whenever one of the following occurs:
 - A transaction ends and the current host screen is not the transaction screen.
 - The Login procedure ends and the current host screen is not the transaction screen.
 - The Recover procedure ends and the current host screen is not the transaction screen, unless the current host screen is the login screen, which results in the application going to the Login flow.
- The result of the Recover flow sequence is the transaction screen, unless the current host screen is the login screen, which results in the application going to the Login flow.
- Make sure to include the login screen in the Recover flow. If the login screen is not included, it could be treated as an "unrecognized screen" and subjected to whatever action is specified for the "unrecognized screen" case. If no action is specified for the login screen case, the Login flow is automatically invoked after recovery fails and the login screen is recognized.
- It is possible to define the Login and Recover flows so that the system goes into an infinite loop. If the Recover flow does result in arrival at the transaction screen, it causes the Recover flow to be invoked. Failing to arrive at the transaction screen again, it causes the Recover flow to again be invoked, and so on.
- Make sure you test your flow to ensure that it does indeed terminate at the transaction screen. This can save considerable debugging effort later, when the application is installed on the target system.

Capturing host screens

The Screen Mapping tool lets you capture screens from an IBM host machine and transfer the screens to the IVR Designer PC.

Opening the Screen Mapping tool

To open the Screen Mapping tool, select Screen Mapping from the Tools menu. IVR Designer displays the Screen Mapping window.

Screen Mapping window menus

The following table describes the menus and menu options in the Screen Mapping window.

Menu	Option	Description
File	Open	This option is available only under the mapping tab. Selecting Open from the File menu is equivalent to clicking the Open button. The system displays a window from which you can choose the screen capture file to open.
File	Exit	Exits the Screen Mapping Tool.
Options	Screen Capture	This option is available only under the capture tab. Selecting Screen Capture opens the Screen Capture Options window on which you specify file-transfer options.
Edit	Copy	This option is available only under the mapping tab. Used to copy the selection from the display screen to the host edit window. This is equivalent to dragging and dropping the selection.
Edit	Next	This option is available only under the mapping tab. Selecting Next from

		the Edit menu is equivalent to clicking the right-arrow button on the mapping tab, which displays the next screen capture file. You can also do this by looking in the screen name combobox and choosing the specific screen you want to process.
Edit	Previous	This option is available only under the mapping tab. Selecting Previous from the Edit menu is equivalent to clicking the left-arrow button on the mapping tab, which displays the previous screen capture file. You can also do this by looking in the screen name combobox and choosing the specific screen you want to process.

Refer to the CLEO Screen Capture Utility guide for more information.

Related topics:

[Screen Capture tab](#) on page 204

[Transferring screen-capture files from the target system](#) on page 206

[Mapping tab of the Screen Mapping tool](#) on page 206

[Using the CLEO Screen Capture Utility](#) on page 207

[Example of how to use the Screen Mapping tool with the Host Definition Editor](#) on page 208

Screen Capture tab

The following table describes the options on the Capture tab of the Screen Capture window.

Parameter	Description
Telnet	If you click the Telnet button, the system opens a telnet window and attempts to connect to the remote system specified in the Screen Capture Options window. If the connection is successful, you will

	be prompted to enter a username and password. After logging on, follow the instructions that are shown within the Screen Mapping Window under the Capture tab.  Note: For more information see the CLEO Host Interface Screen Capture Utility User Guide.
Refresh Files	Redisplays the screen capture files that reside on the CONVERSANT or Avaya IR.
Transfer	Copies and transfers the selected files from the CONVERSANT or Avaya IR system to the IVR Designer PC.
Local Files Display	Displays the screens that have been transferred to the PC from the CONVERSANT or Avaya IR. The screens reside in IVR Designer home directory called Screens.
Remote Files Display	Displays the screens that have been captured from the host machine and are ready to transfer to the PC. These files are in the /vs/data/host directory on the CONVERSANT or Avaya IR system.
Screen Mapping window	This window gives instructions on how to perform a screen capture from the Capture tab. Session <#> is a number between 0 and 10. If the session is not specified, then the utility can be used only to perform maintenance on the screen capture file for the specified application name. No new screen captures can be performed. App <name>: (Application name) is the name of the screen capture file. It resides on the CONVERSANT or Avaya IR system (vs/data/host). The directory is created if it does not already exist.

Transferring screen-capture files from the target system

1. Open the Screen Mapping tool.
2. Click the Capture tab.
3. If you have not already done so, select Screen Capture from the Options menu, and specify the options for transferring screens. Transfer options include the name of the CONVERSANT or Avaya IR system on which the screen files reside, and your login name and password.
4. Click the Telnet button. The system connects to the machine specified in the Screen Capture Options window. You can now run the CLEO screen capture utility.

 Note:

For detailed information on commands for capturing screens using the CLEO screen capture utility, refer to the user guide for the CLEO Host Interface Screen Capture utility. This guide is part of the package you get from CLEO when you purchase the screen capture utility.

5. Start the Screen Capture Utility and capture the needed screens from the Host machine.
6. Click the Refresh files button.

 Note:

This window automatically populates both listboxes with the list of files on the CONVERSANT or Avaya IR system and the IVR Designer PC. If the target system is a Avaya IR, the captured screens are stored in /vs/data/host. If the target system is a CONVERSANT, the captured screens are stored in /lt/trans/sc/host_sc. For CONVERSANT or Avaya IR systems, the screen-capture files are transferred to the Screens folder in the IVR Designer home directory on the PC.

7. Highlight the .sc and .nam files you want to transfer to the PC.
8. Click the Transfer button.

 Note:

You can also select the files on the Remote System and drag them to the PC list box.

9. Click the Mapping tab.

Mapping tab of the Screen Mapping tool

The following table describes the elements on the Mapping tab of the Screen Capture window.

Element	Description

Open button	When you click the Open button, the system displays a window from which you can choose the screen capture file to open.
Start Position	Displays the starting row and column numbers of the selected field.
Length	Displays the number of characters selected.
Screen Name	Displays the name of a screen defined during the screen-capture process.
	Press these buttons to move forward or backward one screen.

Using the CLEO Screen Capture Utility

The Screen Capture Utility is included in a CLEO Host Interface installation running on a CONVERSANT or Avaya IR. By accessing your host applications through the Screen Capture Utility, you can capture host application screens by using a hot-key. These screen-capture files can then be viewed, copied, renamed, or deleted as needed.

Opening the Screen Capture Utility

To open the Screen Capture Utility in Capture Mode and take new screen captures, do one of the following:

- Add the desired command line parameter to your invocation command.
 - Type `screen_capture –session session id` on the command line, or
 - Type `screen_capture –session session id -app application name` on the command line, or
 - Type `ctrl + u` to display the menu options, and select Connect under the Session menu.
- Type `esc b` to capture the currently displayed application screen. `Esc b` is a hot-key and can always be used without having to access the menu bar.

To open in the Screen Capture Utility in Maintenance Mode and work on previously created screen capture files:

- Type `screen_capture` on the command line. An initial screen is displayed, where you are prompted for an application name.
- Type `ctrl + u` to display the menus and begin work.

Closing the Screen Capture Utility

You can exit the Screen Capture Utility by clicking the Exit menu item or by using the hot-key (`ctrl + x`) for exiting the screen capture program. You can use the hot-key without having to access the menu bar.

Example of how to use the Screen Mapping tool with the Host Definition Editor

After the screens have been captured from the IBM Host machine, populate the appropriate fields by selecting, dragging, and dropping the information into the Host Definition editor.

1. Have the Mapping tab on the Screen Mapping window open and click the Open button.
2. Locate the appropriate captured files (.sc) and open them.
3. Select Host from the View menu.
4. Right-click the field directly under Host.
5. Select New from the menu and type a name for the Host definition.
The Host Definition Editor opens with the Host definition tab in front.
6. Select cells from the screen in the mapping tool.
7. Drag the cells and place them in the empty field under Screen Name in the Host Definition Editor. The field populates with the screen information.
8. Click the Screen Identification tab.
9. Select the cells from the host screen.
10. Drag and drop the information onto the field under Row. The fields populate with the appropriate information.

Note:

If the screen has not been defined within the application, you must press the Apply button before you can define the mapping of fields.

Working with database tables

Database tables have two primary uses in IVR Designer applications:

- To store input, such as account numbers or order selections
- To provide information stored in a database, such as an account balance or order status

Tables can be imported (shared-in) using the Application Resources tool. They can also be created and populated from within IVR Designer applications.

Tables are grouped together, displayed, and accessed through the Tables Manager. They are created and edited using the Table Schema Editor and the Table Editor.

For practice using database tables in applications, see [Creating database applications](#) on page 408 and [Using stored procedures in TAS applications](#) on page 483.

The Tables Manager

The Tables Manager displays brief descriptions of all tables defined for the current application. Using the Tables Manager, you can:

- Create new tables
- Rename tables
- Edit a table's schema
- Edit a table's data
- Use the Save Table As command to make backup copies of tables
- Use the right-click menu to perform a variety of other operations on database tables

Opening the Tables Manager

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

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

Table Manager elements

The following table describes the elements of the Tables Manager.

Element	Description
Name bar	Contains the name of the resource manager, distinguishable from other fields in a globals window by its magenta color
Name column	Displays the names of all tables defined for the current application

Description field column	Provides in a brief description of the table and its use

Related topics:

[Working with database tables](#) on page 209

[Creating database tables](#) on page 211

[Making backup copies of database tables](#) on page 212

[Renaming database tables](#) on page 213

[Editing a table's schema](#) on page 213

[Editing a table's data](#) on page 213

[Right-click menu options for the Tables Manager](#) on page 214

[The Table Schema Editor](#) on page 215

[The Table Editor](#) on page 223

Working with database tables

Database tables have two primary uses in IVR Designer applications:

- To store input, such as account numbers or order selections
- To provide information stored in a database, such as an account balance or order status

Tables can be imported (shared-in) using the Application Resources tool. They can also be created and populated from within IVR Designer applications.

Tables are grouped together, displayed, and accessed through the Tables Manager. They are created and edited using the Table Schema Editor and the Table Editor.

For practice using database tables in applications, see [Creating database applications](#) on page 408 and [Using stored procedures in TAS applications](#) on page 483.

The Tables Manager

The Tables Manager displays brief descriptions of all tables defined for the current application. Using the Tables Manager, you can:

- Create new tables
- Rename tables
- Edit a table's schema
- Edit a table's data

- Use the Save Table As command to make backup copies of tables
- Use the right-click menu to perform a variety of other operations on database tables

Opening the Tables Manager

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

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

Table Manager elements

The following table describes the elements of the Tables Manager.

Element	Description
Name bar	Contains the name of the resource manager, distinguishable from other fields in a globals window by its magenta color
Name column	Displays the names of all tables defined for the current application
Description field column	Provides in a brief description of the table and its use

Related topics:

[Working with database tables](#) on page 209

[Creating database tables](#) on page 211

[Making backup copies of database tables](#) on page 212

[Renaming database tables](#) on page 213

[Editing a table's schema](#) on page 213

[Editing a table's data](#) on page 213

[Right-click menu options for the Tables Manager](#) on page 214

[The Table Schema Editor](#) on page 215

[The Table Editor](#) on page 223

Creating database tables

To create a new database table:

1. Open the Tables Manager.
2. Right-click the Tables Manager anywhere other than on the name bar.
3. Select New... from the menu.
The system displays a dialog box and prompts you to name the new table.
4. Enter the name for the table in the space provided.

When naming tables, keep the following in mind:

- Names must be 11 characters or fewer in length. IVR Designer truncates names with more characters to the first 11 characters.
- Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the actual table name.
- Table names are not case sensitive. Whatever case you use, IVR Designer uses upper-case letters when creating the actual table name.
- Table names must begin with alphabetic characters only. After the first character, you can use other standard ASCII alphanumeric characters.

5. Click OK.

The Table Schema Editor window is displayed.

6. Define the fields for the table in the Table Schema Editor window.



Note:

For Oracle databases, the number fields should specify the length as 10 to exactly match the corresponding field length defined in IVR Designer. Alternatively, for Avaya IR 1.2 and Avaya IR 1.2.1, you can specify an Oracle number field without a length; however, any number used in the application must not be greater than length of 10 digits.

7. Click OK.

IVR Designer closes the Table Schema Editor window and displays the new table name as an entry in the Tables Manager. The table is now available for use within your application.

8. (Optional, but recommended) Type a brief description of the table in the Description field.

Making backup copies of database tables

You might want to keep a copy of your original table, so you can restore it if needed. The best way to do this is to save the original table with a different name, using the Save Table As command.

To save a copy of your original table under a different name:

1. Open the Tables Manager.
2. Right-click the name of the table you want to save.
3. Select Save Table As from the menu.
IVR Designer displays the Save Table As dialog box.
4. Type a new name for the table. When naming tables, keep the following in mind:

- Names must be 11 characters or fewer in length. IVR Designer truncates names with more characters to the first 11 characters.
- Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the actual table name.
- Table names are not case sensitive. Whatever case you use, IVR Designer uses upper-case letters when creating the actual table name.
- Table names must begin with alphabetic characters only. After the first character, you can use other standard ASCII alphanumeric characters.

5. Click OK.

Renaming database tables

To rename a database table without altering its contents:

1. Open the Tables Manager.
2. Right-click the name of the table you want to rename.
3. Select Rename... from the menu.
IVR Designer displays a dialog box and prompts you for a new name for the table.
4. Enter the new name for the table in the space provided.
5. Click OK.
IVR Designer changes the name of the table in the Tables Manager window, and changes each reference to the table name in the application.

Editing a table's schema

To edit a table's schema:

1. Open the Tables Manager.
2. Right-click the name of the table whose schema you want to edit.
3. Select Edit Table Schema... from the menu.
IVR Designer displays the Table Schema Editor window.
4. Edit the Table Schema fields as needed.
5. Click OK.

Editing a table's data

To edit a table's data:

1. Open the Tables Manager.
2. Right-click the name of the table whose data you want to edit.
3. Select Edit Table Data... from the menu.
IVR Designer displays the Table Editor window.
4. Edit the table data as needed.
5. Click OK.

Right-click menu options for the Tables Manager

The following table describes the right-click menu options for the Tables Manager.

Menu option	Description
New...	Lets you create a new database table.
Delete	Deletes the database table from the Tables Manager, if not in use. If you have used the table in your application, you must remove its use from the application before you can delete it.
Rename...	Lets you rename the selected database table without altering its contents, and automatically changes each reference to it in the application.
Find Uses of	Opens the Find window, searches for Uses Of the selected table, and presents the results of the search in the Find window results field.
Sharing...	Makes a copy of a table (keeping a copy of the original) and places it in the Sharing directory. When a database table is marked as Shared, it can be imported into other applications using the Application Resources tool.
Stay on Top	(Available only when the Tables Manager is undocked) Toggles on and off. When checked (toggled on), it causes the Tables Manager window to remain on top of other open IVR Designer windows

	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 in. All windows toggled to "Stay on Top", however, will stay on top of any other IVR Designer windows that are open.
Save Table As...	Makes a copy of a table (keeping a copy of the original), assigns it a new name, and places it in the Tables Manager.
Edit Table Schema...	Lets you edit the schema of a table in the Tables Manager.
Edit Table Data...	Lets you edit data in a table in the Tables Manager.

The Table Schema Editor

Use the Table Schema Editor to define or edit a table's schema. This editor appears whenever you create a new database table or select Edit Table Schema... from the Tables Manager right-click menu.

Opening the Table Schema Editor

To open the Table Schema editor window, do one of the following:

- Create a new database table.
- Right-click the Tables Manager window on the name of the table you want to edit, and select Edit Table Schema...

IVR Designer displays the Tables Schema Editor window.

Table Schema Editor field descriptions

The following table describes the fields in the Table Schema Editor and how to use them to define or edit a table's schema.

Field	Description
Key	To designate a field as a key field, select Yes from the drop-down menu. Designating a table field as a key field makes it a unique index. IVR Designer creates only unique indexes. To create non-unique indexes, you must use Oracle or another database program. IVR Designer does let you designate more than one field per record as a key

	field. In such cases, IVR Designer uses all the key field entries to make the unique index.
Name	Enter the name of the field here. When naming table fields, keep the following in mind: • Table field names are limited to 25 characters. If you attempt to use more than 25 characters, the name reverts to the previous name or to UNTITLED, if there was no previous name. • Do not use spaces in table field names. IVR Designer eliminates them when you exit the field. • Table field names are not case sensitive. Whatever case you use, IVR Designer uses upper-case letters when creating a database table name. • IVR Designer does not create variables for database field names. If you want the field names to be treated as variables, you must define them as such, using the Variables Manager.
Class	From the drop-down menu, choose from among the following class types: • Character • Currency • Date • Number • Real • Time
Format	The options in this column depend on what class is selected.

Length	The values possible in this column depend on what class is selected.

Table field class types

The following table describes how IVR Designer treats each Class type in a table. The date and currency formats are all for US English only. Formats for other languages will vary.

Class type	Description
Character	Treats the input as ASCII characters or as numerical digits. Allows only standard ASCII characters as input and does not display extended ASCII characters.
Currency	Treats the input as monetary amounts.
Date	Treats the input as a date. Data must be entered using one of the following formats: • yymmdd • yyyyymmdd • mm-dd-yy • mm/dd/yy • mm dd yy IVR Designer rejects other formats or values. All dates are stored and used in the CONVERSANT or Avaya IR system using the yyyyymmdd format.
Number	Treats the input as a number value or as a phrase number used to call a phrase.
Real	Treats the input as a real number with up to nine decimal places.
Time	Treats the input as a time. Data must be entered as two or three sets of two digits separated by colons, for example, xx:xx:xx or xx:xx. IVR Designer rejects other formats or values. All times are stored and used in the CONVERSANT or Avaya IR system using the hhmmss format.

Table class formats

The following table describes the options available and the defaults for each Class type. The options in the Format column depend on what Class type is selected. The first option for each

class is the default for that class. The date and currency formats are all for US English only. Formats for other languages vary.

Class	Format	Description/Examples
Character	Char	Treats the input as ASCII characters.
	Digits	Treats the input as numerical digits.
Currency	\$0.##	Displays the amount as dollars and cents. Example: \$543.62
	\$0	Rounds and displays the amount to the nearest dollar. Example: \$544
Date	m/d/yy	Displays the month and day with the fewest possible digits, and displays the year with two digits. The month, day, and year are separated with slashes. Example: 5/23/98
	mm/dd/yy	Displays the month, day, and year, all as two-digit numbers separated with slashes. Example: 05/23/98
	m-d-y	Displays the month and day with the fewest possible digits and a two-digit year, separated with hyphens. Example: 5-23-98
	mm-dd-yy	Displays the month, day, and year, all as two-digit numbers separated with hyphens. Example: 05-23-98
	M d, y yyymmdd yyymmdd	Displays the month by name, the day with the fewest possible digits, and a four-digit year. Example: May 23, 1998

Note: All dates are stored and used in the CONVERSANT or Avaya IR system using the yyyyymmdd format. The class format selects only how the date will be displayed and printed.		
Number	0	Treats the numeral input as a number that can be mathematically manipulated.
	Phrase	Treats the numeral input as a phrase number.
Real	0	Rounds and displays the number to the nearest integer value. Example: 5.77777777 becomes 6
	0.0	Rounds and displays the number to the nearest tenth. Example: 5.77777777 becomes 5.8
	0.00	Rounds and displays the number to the nearest hundredth. Example: 5.77777777 becomes 5.78
	0.000	Rounds and displays the number to the nearest thousandth. Example: 5.77777777 becomes 5.778
	0.0000	Rounds and displays the number to the nearest ten-thousandth. Example: 5.77777777 becomes 5.7778
	0.00000	Rounds and displays the number to the nearest hundred-thousandth. Example: 5.77777777 becomes 5.77778
	0.000000	Rounds and displays the number to the nearest millionth. Example: 5.77777777 becomes 5.777778

Time	hh:min	Displays the time in two-digit hours and two-digit minutes. Example: 8:25 in the evening (entered as 20:25) is displayed as 20:25
	hh:min:sec	Displays the time in hours, minutes, and seconds. If no second is entered, defaults to 0 seconds. Example: 8:25 in the evening (entered as 20:25) is displayed as 20:25:00
	hh24:min	Same as hh:min
	hh24:min:sec	Same as hh:min:sec
	hh:min AMPM	Displays the time in 12-hour format, appended with an AM or PM. Example: 8:25 in the evening (entered as 20:25) is displayed as 8:25 PM
Note: All times are stored and used in CONVERSANT and Avaya IR systems using the hhmmss format. The class format selects only how the time is displayed and printed.		

Table field lengths

The following table describes the limits and default values available for each class. The values possible in the Length column depend on what Class is selected.

Class type	Field length limits
Character	Maximum of 255 characters Default: 24  Note: IVR Designer lets you put more than 255 characters in this field. If you do, however, the value reverts to the default (24) without warning when you exit the Table Schema Editor.
Currency	No limit, but values with more than 15 digits are displayed in scientific notation Default: 15

Date	Cannot be changed by the user Default: 8
Number	Maximum number of digits possible: 10 Maximum and minimum values: 2,147,483,647 and -2,147,483,648 Default: 10  Note: For Oracle databases, the number fields should specify the length as 10 to exactly match the corresponding field length defined in IVR Designer. Alternatively, for Avaya IR 1.2 and Avaya IR 1.2.1, you can specify an Oracle number field without a length; however, any number used in the application must not be greater than length of 10 digits.
Real	No limit, but values with more than 19 digits are displayed in scientific notation Default: 10
Time	Cannot be changed by the user Default: 6

Related topics:

[Defining fields using the Table Schema Editor](#) on page 221

Defining fields using the Table Schema Editor

Adding a field to a table

To add a field to a table's schema:

1. Right-click the Fields area of the Table Schema window.
2. Select Add Field from the menu.
IVR Designer adds the new field after the last field already defined in the table.

Inserting a field in a table

To insert a field into a table's schema:

1. Right-click the Fields area of the Table Schema window.
2. Select Insert Field from the menu.
IVR Designer adds the new field before the currently selected field.

Deleting a field from a table

To delete a field from a table's schema:

1. Right-click the Fields area of the Table Schema window.
2. Select Delete Field from the menu.
IVR Designer displays the Confirm dialog box.
3. Click OK.
IVR Designer deletes the selected field and moves the remaining fields up in the schema.

Moving a field within a table

To move a field within a table's schema:

1. Right-click the Fields area of the Table Schema window.
2. Do one of the following:
 - To move the field up (left) in the table, select Move Field Up.
 - To move the field down (right) in the table, select Move Field Down.

Naming or renaming a field

To name or rename a field in a table's schema:

1. Click the Name column on the line of the field you want to name.
2. Type the name you want to assign to the field.

Defining a field as a key field

To define a field as a "key" field in a table's schema:

1. Click the Key column on the line of the field you want to define.
2. Click the drop-down menu pointer.
3. Select Yes from the drop-down menu.

Selecting a class type for a field

To select a class type for a table field:

1. Click the Class column on the line of the field whose class you want to define.
2. Click the drop-down menu pointer.
3. Select the class type you want from the drop-down menu.

Defining a format for a field class

To define a format for a table field's class:

1. Click the Format column on the line of the field whose class format you want to define.
2. Click the drop-down menu pointer.
3. Select the class format you want from the drop-down menu.

Setting the field length

To set the maximum length a table field can have:

1. Click the Length column on the line of the field whose length you want to set.
2. Type the length you want for the field.

When assigning a length to a field, keep the following in mind:

- Make sure the field length as defined in IVR Designer is equal to (strongly recommended) or greater than the corresponding field length in the database on the target system.

**Note:**

The field length as defined in IVR Designer should never be less than the corresponding field length in the database on the target system. Doing so can cause the application to fail.

- For Oracle databases, the number fields should specify the length as 10 to exactly match the corresponding field length defined in IVR Designer. Alternatively, for Avaya IR 1.2 and Avaya IR 1.2.1, you can specify an Oracle number field without a length; however, any number used in the application must not be greater than length of 10 digits.
- If it will be used as part of a host definition, the total size of all fields defined for a single host screen must be equal to or less than 2000 bytes minus 1 byte of overhead for each host field.
If you determine that the total size of all fields defined could exceed this 2000-byte buffer during a database table query, you should explore the possibility of creating a view of the database in Oracle. See your Oracle documentation or your database administrator if you need help with this.

The Table Editor

The Table Editor lets you manipulate the data in a database table defined for the current application on the PC.

**Note:**

Data manipulation in IVR Designer takes place only on your PC. If you want to preserve your data and transfer it to your target system, you must perform the appropriate Code Generation, Application Transfer, and SCI (Service Creation Installation) actions.

From the Table Editor, you can:

- View data records contained in the table, using a variety of navigational means.
- Edit existing data records.
- Add new data records to the table.
- Insert new data records into the table.
- Delete records from the table.
- Perform various operations in the Table Editor, using the right-click menu.

Opening the Table Editor

To open the Table Editor:

1. Open the Tables Manager.
2. Right-click the name of the table that contains the data you want to edit, and select Edit Table Data...
IVR Designer displays the Table Editor window.

Related topics:

[Using the Table Editor](#) on page 224

[Right-click menu options for the Table Editor](#) on page 225

Using the Table Editor

Viewing records

To view data records:

1. Open the Table Editor.
2. Use the Tab, Enter, and directional arrow keys to navigate through the table.

Editing existing records

To edit existing data records:

1. Open the Table Editor.
2. Select the field or fields you want to edit.
3. Make the desired changes.
4. Click OK.

Adding new records

To add new data records to a database table:

1. Open the Table Editor.
2. Choose one of the following actions:
 - Right-click the Table Editor records area, and select Add Record from the menu.
 - Press Control + A.
 - Press Tab from the last record field of a line or the down arrow from the last record line.
3. Fill in the record data fields as desired.

Inserting new records

To insert new data records in a database table:

1. Open the Table Editor.
2. Choose from among the following actions:

- Right-click the Table Editor records area, and select Insert Record from the menu.
- Press Control + I.

3. Fill in the record data fields as desired.

Deleting records

To delete data records from a database table:

1. Open the Table Editor.
2. Choose one of the following actions:
 - Right-click the Table Editor records area, and then select Delete Record from the menu.
 - Press Control + D.

Note: This action cannot be undone. Records are deleted with no warning and no chance to cancel.

Navigating

When working in the Table Editor, keep the following in mind:

- The Insert key inserts a new record line before the currently selected record.
- The Tab key moves from one field to the next. When you reach the end of a line, pressing Tab again adds a new record line at the end of the table.
- The directional arrows can be used to move around already defined fields. If you are on the last line, the down arrow adds a new record line.
- Other navigational options are included in the right-click menu. This menu is available from anywhere in the Table Editor data records area.

Right-click menu options for the Table Editor

The following table describes the right-click menu options for the Table Editor.

Menu item	Description
Delete Record	Deletes the selected record from the table.  Note: This action cannot be undone. Records are deleted with no warning and no chance to cancel the action.
Insert Record	Inserts a new record before the currently selected record.
Add Record	Adds a new record after the last record in the table.

Go to First Record	Moves the cursor to the first record in the table.
Go to Last Record	Moves the cursor to the last record in the table.
Go to Next Record	Moves the cursor to the next record in the table.
Go to Previous Record	Moves the cursor to the previous record in the table.

Working with application resources

The Application Resources tool makes it easy for you to:

- See what resources are in use in the current application.
- Import additional resources into your application, either as private copies or as shared resources.
- See what resources in your application are being shared-out and shared-in.

For more information about and practice sharing resources, see [Sharing TAS resources](#) on page 342.

Opening the Application Resources tool

To open the Application Resources tool, select Application Resources from the Tools menu. The Application Resources window opens and displays the types of resources in use by the application.

Application Resources window elements

The following table describes the elements of the Application Resources window.

Element	Description
	Signifies that there are no resources of that type currently in use. Otherwise, the pathname shows where the specific resource file is located.
	Signifies that there are resources of that type currently in use but that the directory for viewing them is closed.
	Signifies that there are resources of that type currently in use and that the directory for viewing them is open.

Usage tab	Displays the resources used in the current application.
Import tab	Lets you import additional resources into the application.
Sharing tab	Displays the resources that are shared-out and shared-in by the current application.

Viewing additional resource information

To open a closed directory and view its contents, double-click a line containing the   icon.

The directory expands to display its contents.

The  icon before a resource pathname indicates that the resource is in use. The pathname shows where the specific resource file is located.

Related topics:

[About sharing application resources](#) on page 227

[Guidelines for sharing resources](#) on page 228

[Importing additional resources](#) on page 229

[Viewing shared resources](#) on page 232

[Sharing-out an application resource](#) on page 233

[Updating shared-out resources](#) on page 233

[Creating a voice response application](#) on page 234

[Refreshing shared-in resources](#) on page 238

[Right-click menu options for the Application Resources tool](#) on page 238

About sharing application resources

Application resources can be shared between applications on the same PC or between different PCs using mapped network drives.

Application resources that can be shared in IVR Designer include:

- Host definition files
- Database tables
- Custom external functions
- Custom phrase table lists (not speech files)
- FlexWord lists

Defining sharing terminology

The developer who creates an application resource is the resource's owner.

The application resource can be shared only when the owner designates it as a Shared Resource. This is known as sharing-out the application resource (see [Sharing-out an application resource](#) on page 233).

When a shared-out resource is imported into and used by another application, it is said to be shared-in.

Setting up for sharing resources

When IVR Designer is first installed, it creates a directory named Shared, with subdirectories for each type of resource that can be shared. When a resource is designated as a shared-out resource, IVR Designer places it in the appropriate subdirectory.

Global resource managers for resources that can be shared-out have Sharing... menu options on their right-click menus.

The owner has the responsibility for deciding what read and write permissions to set on the shared resource directories and subdirectories. Only the owner of a shared resource may update it, using the Update Shared Resources tool (see [Updating shared-out resources](#) on page 233).

Any developer who wants to use a shared-out application resource on a different PC must first map the shared resource's subdirectory as a network drive. It is only at this point that the shared resource appears on the developer's PC.

Once the directory is mapped, the developer can share-in the application using the Application Resources tool (see [Working with application resources](#) on page 226). To use the shared resource, the developer must import the resource either as shared or as a private copy.

Guidelines for sharing resources

When sharing application resources, keep the following guidelines in mind:

- Whether the developer imports a resource as shared or as a private copy, IVR Designer makes a local copy of the resource. This lets the developer continue working on the application, even if the network is not operating. The Shared option means that the developer cannot alter the original resource, but may only access it. The Refresh From Shared Resources tool lets the developer update a shared-in resource to the current state of the shared-out resource at any time (see [Refreshing shared-in resources](#) on page 238).



Note:

Before you can use the Refresh from Shared Resources tool, you must have already imported one or more resources as a shared resource.

- Shared-in resources are not automatically updated as part of the Verify Design and Code Generation tools' operations. It is the developer's responsibility to update shared-in resources before using either of these tools.
- If the owner of a shared-out resource deletes the resource (using the Delete option from the right-click menu of that resource), IVR Designer warns the owner that the resource is a shared resource and might be in use by other developers. If the owner

chooses to continue, IVR Designer deletes the resource from the Shared directory and corresponding subdirectory.

Any developers who may be using the resource as a shared-in resource are notified of that fact the next time they try to refresh the shared resource (using the Update From Shared Resources tool). At that point, the developer has the option to retain the local copy and become the new owner of the shared resource or delete the resource.

If more than one developer is using the shared resource, each can become an owner of it independently of the others. This creates two resources of the same name on different PCs. If they later try to install the application on the same target system, they must then decide which one to use, whether to overwrite one with the other, or to rename one of them.

Importing additional resources

IVR Designer lets you import six types of additional resources into your application.

These resources can be:

- Resources on your PC that you have shared-out from other applications
- Resources that another developer has made available by sharing-out the resource and sharing the network drive it resides on



Note:

You must first map the other developer's drive as a network drive on your PC before you can import the resource.

IVR Designer also offers two additional types of resources, which cannot be imported. Both of these are included as part of the standard IVR Designer package. They are available for use by all applications and do not need to be imported to be used.

The following table lists and describes the types of resources you can import.

Resource type	Can be imported?	Description
External Functions	Yes	Use External Functions when you need IVR Designer to do something that cannot be done with the currently existing nodes.
FlexWord lists	Yes	FlexWord speech recognition recognizes words by matching phonemes (parts of words).

		 Note: FlexWord speech recognition is not supported on Avaya IR.
Host Definitions	Yes	Host Definitions are the specifications for the interface between the target system and a remote 3270 host machine.
Phrase Tables	Yes	Phrases are bits of spoken or written speech that instruct customers or prompt them for information. They are contained in Phrase Tables.
Tables (Database)	Yes	Tables can be imported into your application from outside or created and populated from within IVR Designer.
Dial Pulse Recognition Grammars	No	Dial Pulse Recognition (DPR) recognizes dial pulse signals received from phones where touchtone service is not available. Obtain descriptions of these grammars by printing a report of the Dial Pulse Grammars from the File menu Print... option.  Note: Dial pulse recognition is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on

		CONVERSANT R9 and earlier releases.

To import additional resources into your application:

1. Open the Application Resources tool.
IVR Designer displays the types of resources in use by the application.
2. Click the Import tab.
The Application Resources window displays the Import view.
3. Select which type of resource you want to import from the drop-down menu at the top of the window.
4. Select whether you want to import the resource as a private copy or as a shared resource.
5. Use the Drive drop-down menu to select the drive you want to import the resource from. To use the default drive (that is, the drive where IVR Designer is installed), leave this field blank.
6. Select the resource you want to import.
7. Click the Import button.

Import tab elements

The following table describes the elements on the Import tab of the Application Resources tool.

Element	Description
Resource type drop-down menu	Use the drop-down menu at the top of the window to select which type of resource you want to import.
Resource column	Select the resource you want from this list, which displays the resources available on a particular drive for the selected resource type.
Location column	Shows the drive and full pathname of resources available for importing.
Create Private Copy and Share radio buttons	Choose how you want to import the resource into your application. To import: • A private copy that you can modify and save with a different filename, select Create Private Copy. • A resource that is being shared by another developer, select Share.

Import button	Click this button to import the resource using the parameters you have selected.
Drive select drop-down menu	Use this drop-down menu to select the drive you want to search for available resources. Assuming you have mapped the appropriate network drives, you can see resources shared by other developers.

Viewing shared resources

You can use the Application Resources tool to see at a glance what resources in your application are shared-out for use by others and what resources you have shared-in from outside your application.

To view shared resources:

1. Open the Application Resources tool.
IVR Designer displays the types of resources in use by the application.
2. Click the Sharing tab.
The Application Resources window displays the Sharing view.

Sharing tab elements

The following table describes the elements found on the Sharing tab of the Application Resources tool.

Icon	Location	Description
	In front of a resource type	Indicates only the types of resources that might be shared-in or shared-out.
	In front of a Shared-out or Shared-in directory	Indicates that there are no shared resources of that type.
	In front of a Shared-out or Shared-in directory	Indicates that there are resources of that type shared-out or shared-in but that the directory for viewing them is closed.
	In front of a Shared-out or Shared-in directory	Indicates that there are resources of that type shared-out or shared-in and that the directory for viewing them is open.

	In front of a Shared-in resource (under a directory)	Shows the drive and pathname for the resource.

Sharing-out an application resource

To designate an application resource you own as shared-out, making it available to other developers and applications:

1. Open the global resource manager for the type of resource you want to share.
2. Right-click the name of the resource you want to share.
3. Select Sharing... from the menu.
The Application Sharing dialog box appears.

Updating shared-out resources

When you designate an application resource as shared, IVR Designer makes it available to any other application on the same PC, or any other developer who has the ability to map a network drive to your PC.

For this reason, you should update your shared resources from time to time, to make the most recent version of the shared resource available to any developer who might be using it. To update these shared resources, use the Update Shared Resources tool.



Note:

Before you can use this tool, you must have already designated one or more resources as a shared resource.

To update your shared-out resources:

1. From the Sharing menu, select Update Shared Resources.
The following message is displayed:

Application will be saved. Do you want to continue?
2. Click Yes.
IVR Designer updates all application resources you have designated as shared, and displays the following message:

Successfully updated all shared resources.
3. Click OK.

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.

Related topics:

[Working with data events \(Events String Editor\)](#) on page 234

Working with data events (Events String Editor)

The Events String Editor lets you determine which variables will be treated as call data events on the target voice response system.

The Events String Editor displays a list of all the variables defined for the application and a list of the variables selected to be treated as call data events.

Opening the Events String Editor

To open the Events String Editor, select Events String Editor from the Tools menu.

Related topics:

[Selecting variables](#) on page 234

[Removing variables](#) on page 235

[Moving variables on the Selected Variables list](#) on page 236

[Events String Editor menus](#) on page 236

[Data Collection options](#) on page 237

Selecting variables

There are three methods for adding a variable to the Selected Variables list:

- Appending the variable to the end of the Selected Variables list.
- Inserting the variable before a particular variable on the Selected Variables list.
- Inserting the variable after a particular variable on the Selected Variables list.

**Note:**

When you add a variable to the Selected Variables list, the Event String Length value at the bottom of the window increases by the length defined for that variable.

Appending a variable

To append a variable to the end of the Selected Variables list, do one of the following:

- Double-click the name of the variable you want to add from the Available Variables list.
- Highlight the variable you want to add from the Available Variables list, and then select Append Variable from the Edit menu.
- Highlight the variable you want to add from the Available Variables list, and then click the Append Variable button.

Inserting a variable before a selected variable

To insert a variable before a variable on the Selected Variables list:

1. Highlight the variable you want to insert from the Available Variables list.
2. Highlight the variable on the Selected Variable list you want it to go before.
3. Do one of the following:
 - Click the Insert Variable Before button.
 - From the Edit menu, select Insert Variable.

Inserting a variable after a selected variable

To insert a variable after a variable on the Selected Variables list:

1. Highlight the variable you want to insert from the Available Variables list.
2. Highlight the variable on the Selected Variable list you want it to go after.
3. Click the Insert Variable After button.

Removing variables

There are two methods for removing variables from the Selected Variables list:

- Removing a single variable.
- Removing all variables.



Note:

When you remove variables from the Selected Variables list, the Event String Length value at the bottom of the window decreases by the length defined for those variables.

Removing a single variable

To remove a single variable from the Selected Variables list, do one of the following:

- Double-click the name of the variable you want to remove from the Selected Variables list.
- Click to select the variable you want to remove, and then select Delete Variable from the Edit menu.
- Click to select the variable you want to remove, and then click the Delete Variable button.

Removing all variables

To remove all variables from the Selected Variables list, do one of the following:

- Click the Delete All Variables button.
- Click the word Clear on the menu bar.

Moving variables on the Selected Variables list

To move a variable up or down on the Selected Variables list:

1. Click to select the variable you want to move.
2. To move the variable:
 - Up, click the Move Up button or select Raise Variable from the Edit menu.
 - Down, click the Move Down button or select Lower Variable from the Edit menu.

Events String Editor menus

The following table summarizes the menu options for the Events String Editor.

Menu options	Description
Reset	Resets the Events String Editor to the selections and status it had when it was first opened.
Clear	Removes all variables from the Selected Variables list.

Edit	Contains the following options: • Add Variable – Inactive in the current version of IVR Designer. • Append Variable – Appends a variable selected in the Available Variables list to the end of the Selected Variables list. • Insert Variable – Inserts a variable from the Available Variables list before a selected variable in the Selected Variable list. • Delete Variable – Removes the selected variable from the Selected Variables list. • Lower Variable – Moves a variable downward on the Selected Variables list. • Raise Variable – Moves a variable upward on the Selected Variables list.

Data Collection options

The following table summarizes the Data Collection options for the Events String Editor.

Option	Description
Prompt and Collect Nodes	When checked, tracking is turned on for every Prompt and Collect node in the application. The information is collected in Call Data Handling reports. These reports can be viewed from the target system.
Menu Nodes	When checked, tracking is turned on for every Menu node in the application. The information is collected in Call Data Handling reports. This report can be viewed from the target system.
Auto Menu Nodes	When checked, tracking is turned on for every Automenu node in the

	application. The information is collected in Call Data Handling reports. This report can be viewed from the target system.

The total number of bytes used by the tracking event variables should not exceed 400. To collect more variables, change the Save Response In variable field size in each tracked node. In other words, if you make the variable size smaller in the Save Response In field, more variables can be collected but with less information in each variable. When tracking is turned on, IVR Designer tracks the input or data entered as a response in a Menu, Automenu, or Prompt and Collect node.

Refreshing shared-in resources

When you share-in (import as shared) an application resource, IVR Designer makes a local copy so you can continue to work with it even if the network is not operating.

You should refresh your shared-in resources from time to time, to ensure that you are working with the most recent version of the shared resource. To refresh shared-in resources, use the Refresh from Shared Resources tool.



Note:

Before you can use this tool, you must have already imported one or more resources as a shared resource.

To refresh your shared-in resources:

1. From the Sharing menu, select Refresh from Shared Resources.
The following warning message is displayed:

Refreshing from shared resources. This operation cannot be undone. Are you sure you want to refresh from shared resources?
2. Click OK.
The system displays the Resource Refresh Results window, and shows a summary of the results of the Refresh from Shared Resources action.
3. Click OK.

Right-click menu options for the Application Resources tool

The following table describes the Application Resources tool right-click menu options.

**Note:**

The right-click menu is accessible only from the Usage or Sharing tabs. To access this menu, select any resource subdirectory, and then right-click.

Menu item	Description
Delete	(Usage tab only) Deletes a resource that has been imported. This option does not let you delete: • A system resource (such as Standard Speech) • A resource that is in use by the current application
Change to Private Copy	(Usage tab only) Creates a private copy of a resource that you have previously designated as shared-in.
Expand	Provides an expanded view of the selected directory.
Collapse	Provides a collapsed view of the selected directory.
Expand All	Expands all directories.
Collapse All	Collapses all directories.

Chapter 7: 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 TAS application](#) on page 330.

Related topics:

[Checking an application for errors](#) on page 241

[Simulating an application](#) on page 243

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 an application, see [Verifying a TAS application](#) on page 331.

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.

Element	Description
Results display pane	Displays the results of the verification.
	Indicates that possible errors at this node are hidden. To view the possible errors in this node, double-click the icon to expand it.

	Indicates that possible errors at this node are displayed below the node name.
	Indicates a possible error or omission. The node where the possible error or omission is located is listed just above this description.
Call flow tabs	Used 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.
	Lets you scroll to different call flows to view possible errors.

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 the search results for another call flow, click the tab for that call flow at the bottom of the window.

Related topics:

[Right-click menu options for the Verification Results window](#) on page 242

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.

Menu option	Description
Edit	Locates the node in the design pad.
Verify Node	Verifies only the selected node for possible errors or omissions.
Verify Call Flow	Verifies only the current call flow.
Verify Design	Verifies the entire application design.
Expand	Expands a collapsed node entry, indicated by the * 📁 icons.
Collapse	Collapses an expanded node entry, indicated by the = 📁 icons.
Expand All	Expands all node entries in the current call flow.
Collapse All	Collapses all node entries in the current call flow.

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 TAS application performance](#) on page 332.

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.

Element	Description
Stop button	Stops the simulation.
Pause button	Causes the simulation to pause.
Play button	Resumes running a simulation that has been paused or stopped, or runs a simulation that has ended.
Reset Variables button	Resets all variables to their default values.
Step-Through button	Steps through a simulation, one node at a time.
Simulation script field	Displays a script of all simulated events while the simulation is running.
Time Remaining field	Displays a "countdown" for each node that uses a timed delay for responses.
Collected Digits field	Contains any digits that have been collected (using touchtone input or data from a table) and not used.
Touchtone keypad	Simulates caller input from a touchtone telephone.  Note: 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.
Clear button	Clears any digits in the Collected Digits field.
ASR button	Simulates a spoken response from the caller.
Asynchronous Event button	Simulates an asynchronous event. 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.  Note: If you have defined a routine to handle asynchronous events, pressing this button calls that routine.
Force Timeout button	Enables you to skip the countdown, or delay, that is programmed into many nodes, and go on to the next node.
DPR button	Simulates a Dial Pulse Recognition (DPR) response from the caller.

Related topics:

[Running an application simulation](#) on page 245

[Examining the simulation script](#) on page 246

[Right-click menu options for the Simulation tool](#) on page 247

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.



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.

Option	Description
Undo	Inactive in the current version.
Cut	Removes the highlighted text and copies it to the clipboard.
Copy	Copies the highlighted text to the clipboard, leaving the original where it is.
Paste	Pastes a selection previously copied to the clipboard to the current cursor location.
Delete	Removes the highlighted text. The deleted text is not copied to the clipboard.
Select All	Highlights all text in the script window so you can cut, copy, or delete it.

Chapter 8: Generating and transferring an application

Generating and transferring an application describes how to use the Code Generation tool to generate the files necessary to install and compile an 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 Transfertool 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 code and transferring and installing a TAS application](#) on page 333.

Related topics:

[Generating TAS instruction code](#) on page 249

[Transferring the application](#) on page 261

[Transferring files using a modem](#) on page 269

Generating TAS instruction code

The Code Generation tool generates the code files necessary to install and compile the application on the target voice response system.

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. For example, if no database tables are defined in the application, the

Database files option is inactive. If no hosts are defined, the Host definition file option is inactive, and so on.

Related topics:

[Setting Code Generation options](#) on page 250

[Generating the source code files](#) on page 252

[Code generation file types](#) on page 253

[The code generation process](#) on page 255

[Menus in the Code Generation window](#) on page 256

[Adjusting the volume of generated audio files](#) on page 258

[Database Table Provisioning window](#) on page 259

Setting Code Generation options

The Code Generation Options dialog box lets you set options for the Code Generation process.

To set the Code Generation options:

1. Open the Code Generation/Application Transfer tool window.
2. Click the Code Generation tab.
3. From the Options menu, select Code Generation.
IVR Designer displays the Code Generation Options dialog box.
4. Check the appropriate boxes to select the options you want.
5. Click OK.

Code Generation options

The following table describes the options in the Code Generation Options dialog box.

Option	Description
SCCS strings	Causes Source Code Control System (SCCS) identification strings (for example, %W%) to be written to all target files that support a comment convention. The default is disabled (unchecked).
Debug statements	Adds comment lines that trace function calls in the Code Generation source code to the *.t and *.h files. Useful for debugging the generated application code on the target system. The default is disabled (unchecked).
Step execution	Causes the Code Generation process to pause after the code is generated for each call flow node. Useful only for

	debugging the Code Generation software. The default is disabled (unchecked).
Show warnings	Generates warnings for possible application errors or missing files, such as unrecorded speech files. The default is enabled (checked).
Use SP style TTS speech rate codes	Causes the Code Generation process to use escape codes for the SP circuit card, rather than the default SSP circuit card escape codes. The default is disabled (unchecked).
Generate separate .pl files	Generates a separate *.pl file for each phrase table defined for the application. The default is disabled (unchecked). When checked, this option: • Assigns the filename EBS_language_name .pl to the phrase table containing the Enhanced Basic Speech (EBS) for the standard speech phrase table (called Standard Speech in IVR Designer). For example, if the EBS language is US English - Female, IVR Designer generates a phrase table file called US_English.pl. • Uses application_name.pl for the default custom phrase table (called Phrase Table in IVR Designer), unless you have renamed the default custom phrase table. For example, if you have an application named new_app, and you generate the phrase table file without renaming the file, IVR Designer generates a file called new_app.pl. • Gives any other phrase tables the name table.pl, where table represents the phrase table name you assigned to it when you created it. For example, if you create a phrase table and name it my_table, IVR Designer generates a phrase table file called my_table.pl. When this option is unchecked, IVR Designer generates a single *.pl file

	called application_name.pl. IVR Designer lets you create a maximum of 8 phrase table (*.pl) files.
Maximum number of records returned per database query	Lets you define how many records can be returned for each database query operation. The default value is 30.  Note: This number represents the maximum limit of records for all tables, so you must select a value for this field carefully. Set the number high enough to allow for the greatest number of records you want to return for any one table. At the same time, consider that the higher this number, the greater the amount of memory you are committing.
F_TEMP workspace size	Lets you set the size of the temporary workspace required by an application. The default value is 288. The range is 288 through 102400.  Caution: Because the workspace is allocated on a per-instance basis, exercise caution when increasing the workspace size. Making the workspace too large can cause memory conflicts.

Generating the source code 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 Code Generation Options you want.
4. (Optional) Select the tables you want to include in your build.
5. Select the files you want to generate.
To generate specific types of files, check the appropriate boxes on the list in the Target Files to Generate box.
6. Select the target platform for which code will be generated.

- To generate source code for CONVERSANT R9 or for CONVERSANT V8, select R9 or earlier Conversant Platforms.
- To generate source code for Avaya IR Releases 1.0 through 1.2, select Avaya Interactive Response R1.0 - R1.2 Platform.
- To generate source code for Avaya IR Release 1.2.1 or later, select Avaya Interactive Response R1.2.1 or later Platform.

7. Do one of the following:

- Click the Apply button.
- Select Build All from the Build menu.
If Build All was selected from the Build menu and 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.

Code generation file types

The following table describes the types of files generated during the Code Generation process that are available for transfer during the Application Transfer process.

If no resource exists in the application for a particular file type (such as Host Definitions or Tables), that option is inactive in the Code Generation tab window, and no files are generated for that file type.

File types	Names and extensions	Description
Script files	*.t	Contains the actual TAS code (script) for the application.
Script files	*.h	Contains the header information for the *.t file.
Script files	*.D	Contains Call Data Events information.
Database files	*.sql	Contains the table definitions and data for tables being exported.  Note: Use the Database Table Provisioning window to select the tables to include in the build. By default, all

		tables defined for the application are included.
Host Definition files	*.hs	Contains information about host definitions. One file is generated for each host defined for the application.
Phrase List files	*.pl	Contains a list of all phrases defined for the application. The list is generated even if the application does not use recorded speech.
Phrase sound files	*.vis	Contains *.wav audio files converted into a CONVERSANT-specific or Avaya IR-specific format.
Automatic file (always generated)	*.ccv	Provides details on how much of the code was Simulation tested, the number of call flows in the application, and the number of different types of nodes used.
Automatic file	*.fea	Contains a list of features used by the application.
Automatic file	*.toc	Provides table of contents information for all the source files and target files in the application.
Automatic file	*.err	Contains information about errors encountered during the Code Generation process.
Automatic file	*.flx	Converts FlexWord lists into a format the target system can use.

Automatic file	Resource	Lists all the database tables defined for the application. If the table is defined as a remote host table, there is an additional field to identify the remote host machine name.
Automatic file	LangName	Contains the name of the Enhanced Basic Speech (EBS) language used by the application.
Automatic file	symbols	Contains information about application symbols – not used by the target system.

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.

Beginning with Release 5.1, the code generation process checks the length of variables and database fields to which a variable is assigned. IVR Designer will display an error message if the length of the variable or database field to which a variable is assigned is not large enough to accommodate the variable. For example if a Set and Test node is used to assign a variable with a length of 20 to a variable with a length of 10, IVR Designer will display an error message similar to the following:

Source length [20] is larger than the destination field variable_name .

Variable length checking applies only to character variables and character strings. IVR Designer displays variable length error and warning messages only if the Show warnings option in the Code Generation Options dialog box is enabled (checked). As a result of variable length checking, you may see error messages during code generation when updating an application to Release 5.1, even if no changes were made to the application.

Menus in the Code Generation window

The following table describes the menus in the Code Generation window and their options.

Menu	Option	Description
File	Open	Not available from the Code Generation tab view.
File	Exit	Closes the Code Generation/Application Transfer dialog box immediately.
Options	Code Generation	Opens the Code Generation Options window.
Options	Application Transfer	Not available from the Code Generation tab view.
Tools	External Function Source Editor	Opens the External Function Source Editor window, where you can create or edit custom external function TAS files. Inactive if no custom external functions have been defined for the application.
Tools	Database Table Provisioning	Opens the Database Table Provisioning window, where you can select the tables to include in the build.
Tools	Refresh From Shared Resources	Refreshes any shared resources used in the application. Inactive if no application resources are shared.

Target	[Generated files list]	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.
Build	Build All	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 Yes. To cancel the operation, click No.
Build	Remove	Removes all previously generated files from the application. If none exist, this menu item is inactive.  Note: 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 Yes. To cancel the operation, click No.
Build	Errors	Displays the error log, if one exists. If no error log

		exists, this menu item is inactive.

Adjusting the volume of generated audio files

Note:

The following procedure for adjusting the volume of generated audio files applies only to CONVERSANT V8 and R9 systems.

During the code generation process, recorded speech files are converted from .wav (PCM) format to .vis (ADPCM32) format. If the volume of the converted files is too high or too low when played back on the CONVERSANT system, you can use the following procedure to change the volume setting for the audio conversion process. After you change the volume setting, you have to run the code generation process, transfer the application, and check the volume level again.

1. Use Notepad or Wordpad to open the audio_conv.ini file in the following folder:
C:\Program Files\Avaya\Avaya IVR Designer\bin

The audio_conv.ini file contains the following text:

```
wav2vis -s 1 <input_file> <output_file>
```

where -s is the volume scaling option, and 1 is the volume scaling factor. To increase the volume, set the volume scaling factor to a value greater than 1. To decrease the volume, set the volume scaling factor to a value less than 1.

Note:

Do not change any text in the audio_conv.ini file other than the volume scaling factor.

2. Save and close the audio_conv.ini file.

Examples

To increase the volume of recorded speech files by 6 dB, change the volume scaling factor to the following:

```
wav2vis -s 2 <input_file> <output_file>
```

To decrease the volume of recorded speech files by 6 dB, change the volume scaling factor to the following:

```
wav2vis -s 0.5 <input_file> <output_file>
```

Database Table Provisioning window

Use the Database Table Provisioning window to select the tables you want to include for Code Generation and export during Application Transfer.

To open the Database Table Provisioning window, select Database Table Provisioning from the Tools menu of the Code Generation window.

IVR Designer displays the Database Table Provisioning window.

Database Table Provisioning window elements

The following table describes the fields on the Database Table Provisioning window.

Field	Description
Table Name	Displays the names of all tables defined for the current application.
Export Schema	Use to select which table schemas will be exported during the Code Generation and Application Transfer processes. • For table schemas you want to include for export, select True in the Export Schema field. • For table schemas you do not want to include for export, select False in the Export Schema field.
Export Data	Use to select which table data will be exported during the Code Generation and Application Transfer processes. • For tables that contain data you want to include for export, select True in the Export Data field. • For tables that contain data you do not want to include for export, select False in the Export Data field.

Database ID	Displays the names to be assigned to the tables when they are transferred to the target voice response system. You must assign a Database ID to each table defined for your application, according to the following guidelines. For CONVERSANT systems, if the table is to reside on: • The CONVERSANT system, select a DB#_LOCAL format. • A remote host (relative to the target system), select a DB#_REMOTE format. For Avaya IR systems, select the DBDIP# that corresponds to the database that you want to access. The association between a DBDIP# and a database is administered on the target Avaya IR system.  Note: For CONVERSANT systems, if the table is to reside on a remote host, you must also enter a name in the Host Name column. For Avaya IR systems, do not enter a name in the Host Name column for any host.
Host Name	Displays the name of the remote host where the remote database table resides.  Note: Host names can have a maximum of 11 characters and must start with an alphabetic character.

Selecting tables and data for export

By default, all table schemas and data are selected for export. If there are table schemas or data you do not want to export, you must select the value False for them.

For example, if you plan to populate the table with data from another source, export the table schema but not the data. But if the schema already exists on the target machine, and you want to populate it with new data, export the table data but not the schema.

To select tables and data for export:

1. Open the Database Table Provisioning window.

Each table name appears in the Table Name column.

2. Select table schemas you want to include for export.
3. Select table data you want to include for export.
4. Assign a Database ID from the pull-down menu for each table in your build.
5. If the table is to reside on a remote host, enter the name of the remote host in the Host Name field.
6. Click OK.

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.

Related topics:

[Setting application transfer options](#) on page 261

[Transferring an application to the target system](#) on page 265

[Transferring files from floppy disks to the target system](#) on page 267

[Transferring application source files](#) on page 267

[Application Transfer window menus](#) on page 268

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.

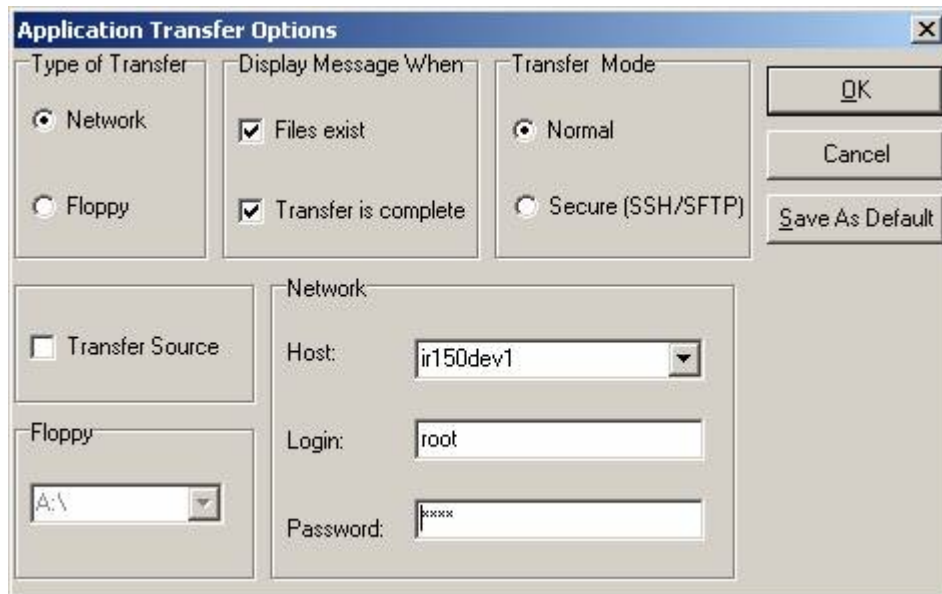


Note:

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 269.

- 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.



- 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.

Related topics:

[Configuring SSH settings to use the SFTP service for transferring applications](#) on page 264

Configuring SSH settings to use the SFTP service for transferring applications

To use the SFTP service, certain SSH-related configuration changes need to be made to the `ssh_config` and `sshd_config` files on the IR system.



Note:

SSH is enabled by default in Solaris 10 Update 3. No configuration change is required if login/user is other than root. To enable root login via SSH, uncomment the attribute `PermitRootLogin` and set it to `yes` in the file `/etc/sshd_config`. Then refresh the SSH service by using the command `svcadm refresh ssh`. IVR-D will be able to transfer application via SFTP once SSH login is enabled for root.

The `ssh_config` file resides in the `/etc/ssh/ssh_config` directory and the `sshd_config` file resides in the `/etc/ssh/sshd_config` directory. These files contain the SSH configuration information.

-
- 1. In the `sshd_config` file, uncomment the following lines by deleting the # character. Also, change the settings from `no` to `yes` or vice-versa.


```
Protocol 2,1
PermitRootLogin yes
RhostsRSAAuthentication no
```

**Note:**

If the SFTP service will use root login, `PermitRootLogin` should be set to `yes`, otherwise it should be set to `no`.

2. In the `ssh_config` file, uncomment the following lines by deleting the `#` character. Also, add the lines if any of these lines are not present.

```
ForwardX11 yes
PasswordAuthentication yes
HostbasedAuthentication no
StrictHostKeyChecking ask
ConnectTimeout 0
AddressFamily any
```

Next steps

Stop and restart the SSH service. Make the above changes in the system to ensure that the SFTP-server (`/usr/lib/ssh/sftp-server`) process starts up after reboot. The SFTP-server process needs to be running in order to allow SFTP. For this to occur, changes in the `sshd` and `prngd` scripts are required.

Transferring an application to the target system

Before you can transfer your application to the target system, you must first:

- Generate the application code files using the Code Generation tool.
- Set the Application Transfer Options.

When transferring an application with a name matching an application that already exists on the target voice response system, Avaya recommends that you first remove the old application files. To do this, use the `scrm` command on the target system. See [Removing an application](#) on page 294. For example, suppose you had an application `abc_auto` on the target system that you later replaced with a new, totally different `abc_auto` application that does not use the same FlexWord lists and database tables. When you transfer the new `abc_auto` application to the target system, only the files with matching filenames are replaced. The old FlexWord lists and database tables remain, however, and can cause conflicts when you use the new version.

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 (such as application files or database files). Use the drop-down menu below the Local Files pane to select the file types.

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 discontinuous 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 select 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 select 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 select 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.

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.

Menu	Option	Description
File	Open	Temporarily opens another application for application transfer purposes only. You can: • Select the application of your choice from a standard Open dialog box. • View files for the selected application in the Local Files pane. • Work with the newly selected

		application as if it were the application currently open on the IVR Designer design pad.  Note: This option remains active only as long as you are actually in the Application Transfer tab. When you leave the Application Transfer tab, IVR Designer reverts to the currently open application for Application Transfer.
File	Exit	Closes the Code Generation/Application Transfer dialog box.
Options	Code Generation	Is not available from the Application Transfer tab view.
Options	Application Transfer	Opens the Application Transfer Options window.
Edit	Select All	Selects all available Local Files for transfer to the target destination.

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 271. 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](#) on page 265.

Note:

The modem 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, 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)

Related topics:

[Windows 2000 PPP configuration](#) on page 271

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 .
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).
16. Click OK (in the Internet Protocol (TCP/IP) Properties 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.
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.

Chapter 9: Converting Script Builder applications

Converting Script Builder applications provides information about how to convert Script Builder applications to IVR Designer applications.

You can convert existing Script Builder applications to IVR Designer automatically, without having to recreate them in IVR Designer.

The Script Builder Conversion tool is activated by an option on the File menu. After you select the application on the target CONVERSANT system, the Script Builder Conversion tool copies the application to the IVR Designer PC, and converts it to IVR Designer format. After the application is converted, you can modify it as desired before reinstalling it on the Avaya IR or CONVERSANT system.

Depending upon the complexity and features used in the original Script Builder application, the Script Builder Conversion Tool may not be able to convert the application completely. You may have to do some initial preparation and follow-up reconstruction. But in many cases, the conversion tool will convert most, if not all, of the original application. This can result in substantial savings in development time when moving to IVR Designer.

At the end of the conversion process, IVR Designer displays a list of any conversion issues discovered during the conversion process. This list includes the functions from the original Script Builder application that were not converted and need to be recreated manually.

The conversion process does not have any impact on the original Script Builder application. It still resides on the CONVERSANT system until you remove it.

Related topics:

[Prepare application for conversion](#) on page 275

[Convert an application](#) on page 278

[Resolving conversion problems](#) on page 279

[Breaking a long application into sections](#) on page 288

Prepare application for conversion

Before converting a Script Builder application to IVR Designer, complete the following steps.

1. Ensure that the correct IVR Designer and SC software are installed.

For the client side (PC), install Voice@Work, version 4.0 or later, or IVR Designer. For the server side (target Avaya IR or CONVERSANT system), install Intuity Service Creation Integration Package (SC), version 8.0 or later.

2. Ensure that a LAN connection is established between the IVR Designer PC and the Avaya IR or CONVERSANT system with the Script Builder application. You cannot use floppy disks to transport Script Builder applications onto the IVR Designer PC.
3. Ensure that the application is completely built with Script Builder and that it is verified and working.
Before a Script Builder application can be converted, it must first be verified and installed on an Avaya IR or CONVERSANT system. The IVR Designer conversion routine compares a Script Builder application's date and time to its .t code (TAS code). They must match. Otherwise a verification error is generated.

If the .t code file has been modified, the date and time for the modified file will not match the date and time of the original Script Builder application, and the application will not be eligible for conversion. If the Script Builder application is verified and installed again, the modified .t code will be overwritten. Capabilities that were added will be lost.

If the .t code has been changed, examine the original Script Builder application and the modified .t code to determine what changes were made. Make careful notes. Then, verify and install the original Script Builder application. This overwrites the modified .t code. Use IVR Designer to convert the Script Builder application. Then, incorporate the additional processes in the converted IVR Designer application.

4. Ensure that the Script Builder application to be converted resides on the target Avaya IR or CONVERSANT system in /att/trans/sb.
IVR Designer does not look in other directories for the applications to be converted. If it is not already there, move the Script Builder application.
5. Ensure that any custom external functions are located in appropriate directories. Before conversion, move custom external functions to one of these directories:
 - /att/trans/sb/application name
 - /vs/bin/ag/lib
 - /vs/bin/ag/eascripts
6. Examine the Script Builder application for the following occurrences:
 - a. Variables have the same name but are in a different case (e.g., my_field and My_Field).
In Script Builder, the variable My_Field is different from my_field because Script Builder distinguishes between upper and lower case. However, IVR Designer does not. If an application with duplicate variables is selected for conversion, IVR Designer halts the conversion and provides an issue message.

Create new variables so that no two variables have the same name. Modify the code in the Script Builder application to take into account these new names.

b. More than one standard language is used.

Check the Script Builder application for use of more than one standard language. An IVR Designer application can support only one standard language. That's because Text-to-Speech can support only one language at a time. However, TAS code can support more than one standard language.

If more than one standard language is used in the Script Builder application, break the application into a menu that calls other Script Builder applications, each using one language. For example, the menu would allow callers to press 1 for English, 2 for Spanish, and so forth. The menu would then call separate applications. Convert each of these applications separately into IVR Designer applications.

c. Database tables appear with lower-case characters.

Database table names are changed to upper case during conversion to IVR Designer. When the converted IVR Designer application and its tables are re-installed on the Avaya IR or CONVERSANT system, there will be two database tables: the original one with lower-case characters in its name and the newer one all in upper case.

However, if the original database table name was in all upper case, then that database will be used by the converted IVR Designer application, and no duplicate table will be created.

To avoid having duplicate tables, use SQL commands after installation of the converted application to export the data from the original table into the new table. If the original Script Builder application will no longer be used, delete the original table after the new application is installed. This conserves space.

d. The following external functions are used: Concat, Length, Parse, and Substring.

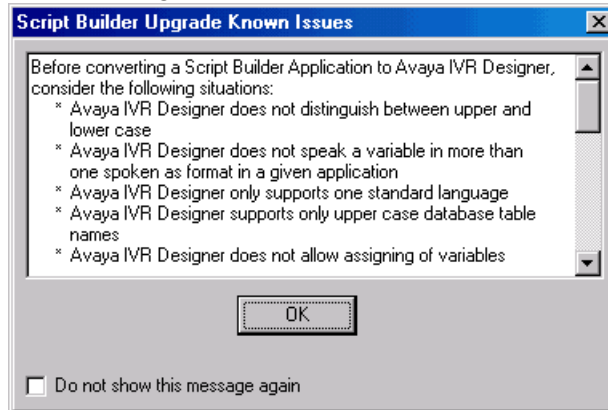
These external functions are converted to IVR Designer as custom external functions, not as standard functions. The converted IVR Designer application will be more difficult to maintain and will run less efficiently than if these functions were coded in IVR Designer.

In the Script Builder application, carefully document where and how these external functions are used. Delete them. After conversion, create new IVR Designer code using the Concat, Length, Parse, and Substring expressions in Set and Test nodes.

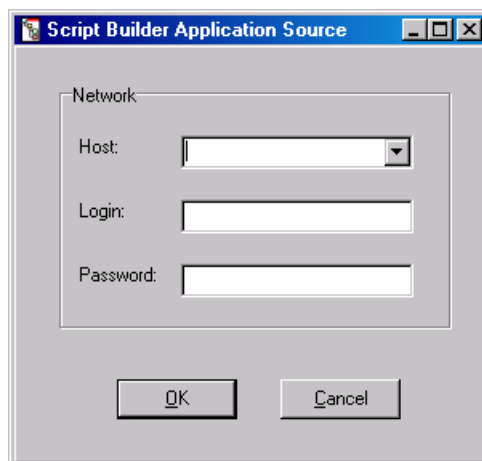
Convert an application

To convert a Script Builder application to IVR Designer, complete the following steps.

1. In IVR Designer, select File > Open Script Builder Application.

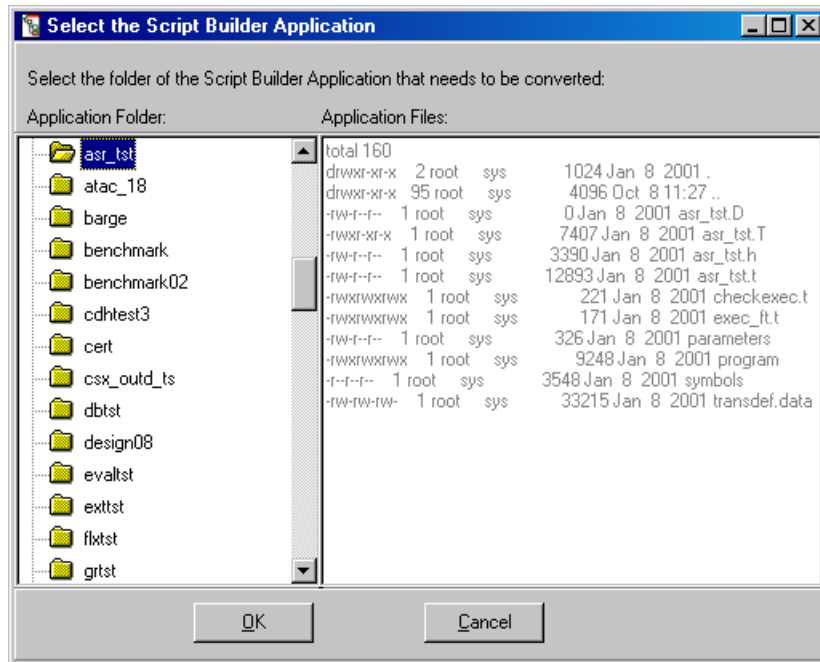


2. Click OK in the Known Issues dialog box.
3. Overwrite the default application name with the name of your application. Use a name different from the original Script Builder application. Do not overwrite any existing applications.
Most of the new application files overwrite the original application files upon installation on the target Avaya IR or CONVERSANT system. However, some of the original files are not overwritten. This may cause maintenance problems later. It is good practice to use a different name for the converted IVR Designer application.



4. In the Script Builder Application Source dialog box, select the CONVERSANT system that has the Script Builder application you wish to convert.
If the target system is not on the list, type its name or IP address in the Host field.
5. Log onto the target CONVERSANT system as root and enter the password.

The IVR Designer PC must be connected over a network to the target system. Script Builder applications cannot be copied to diskette from the CONVERSANT system and run on the PC.



6. In the left pane, select the application folder containing the Script Builder application to be converted. Then, click OK to convert it.
7. Handle the verification error, if necessary.
If the system displays the following message, the Script Builder application must be verified and installed again. The .t files (TAS code) will be overwritten when the application is verified and installed. Any modifications to the .t files made after the last verification and installation will be lost.

The Script Builder application that you have selected needs to be verified and installed without any errors before it can be upgraded to Avaya IVR Designer.

Resolving conversion problems

During or after conversion of the Script Builder application, you may encounter error messages, a converted application that doesn't function properly, or new or unfamiliar variables, database table names, or other items. Resolve conversion problems describes these conversion problems and how to correct them.

Related topics:

[Simulation halted](#) on page 280

[Remote database tables do not appear](#) on page 282

[SCI variables not found](#) on page 283

[Variables are not spoken correctly](#) on page 284

[Discard Previous Touchtones always set to True](#) on page 284

[Seasonal greetings do not work](#) on page 285

[New database tables are not the most current](#) on page 285

[Time Branch nodes show redundant date branches](#) on page 285

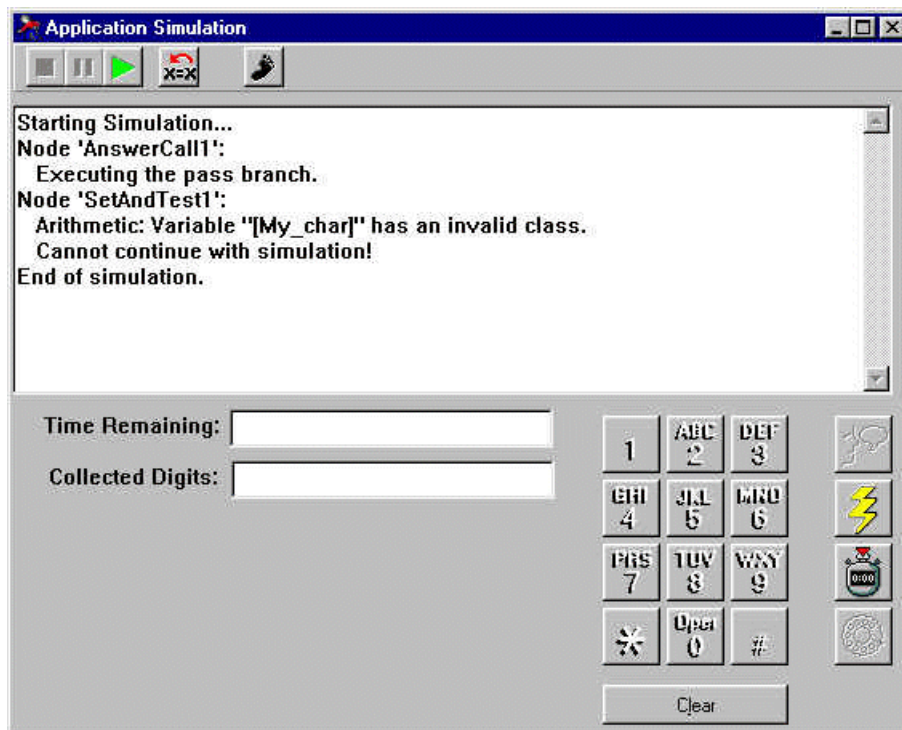
[Custom phrase table names](#) on page 286

[Host definition issues](#) on page 286

Simulation halted

Problem

Simulation of converted application halts due to errors with mixed data types.



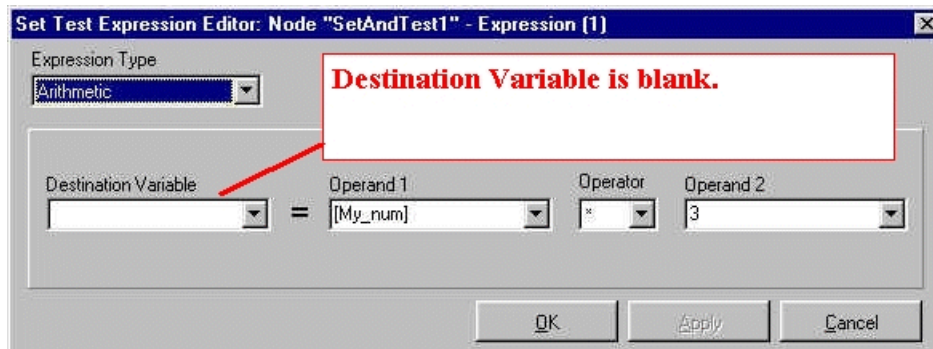
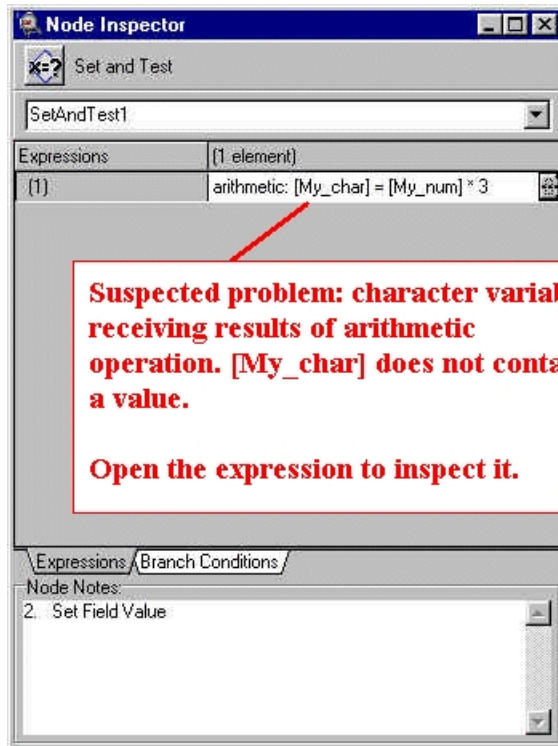
Reason

These errors generally occur because Script Builder allows variables to receive values of a different type during arithmetic operations (e.g., assigning the results of a numerical operation to a character variable). IVR Designer does not allow a variable to receive a value of a different type as a result of an arithmetic operation. The conversion process does not catch these assignments. That is, the converted application can be verified and installed without error. Errors will appear, however, during simulation or in actual operation of the application on the target Avaya IR or CONVERSANT system. It is possible that mixed data type errors can occur

during operations other than arithmetic operations. However, problems with arithmetic operations are the most obvious.

How to correct

In the converted IVR Designer application, check any Set and Test nodes where the code does not operate correctly.



If any fields in the Set Test Expression Editor are blank, IVR Designer was unable to create the variable. Create new variables with the appropriate data types, then rebuild the expressions.

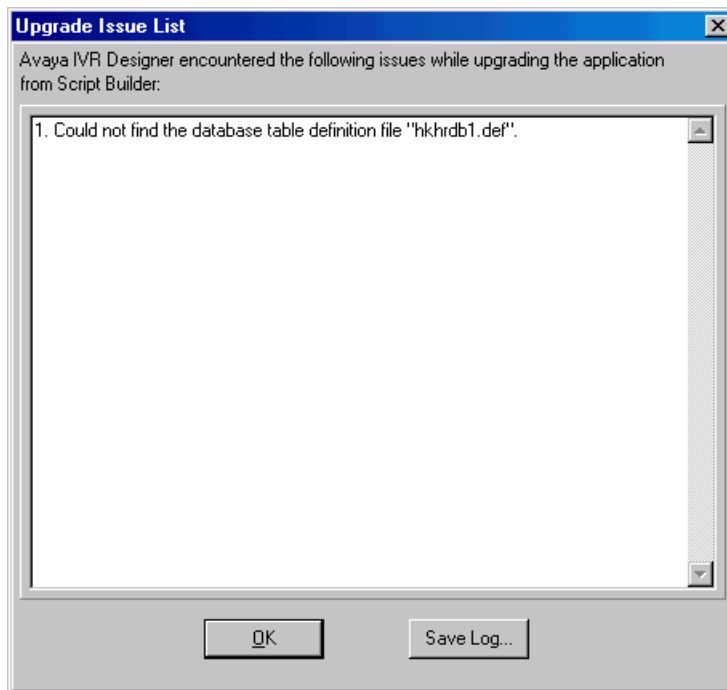
Remote database tables do not appear

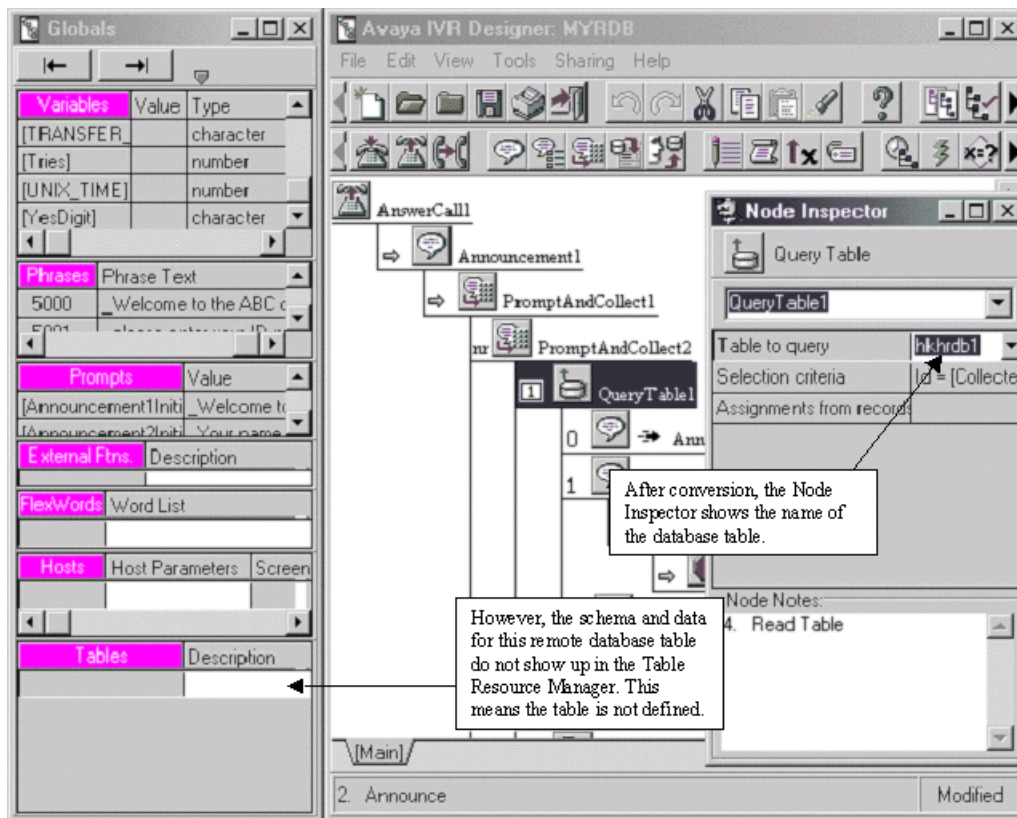
Problem

Remote database tables do not appear in converted IVR Designer Application Resources.

Reason

IVR Designer does not support remote database tables. The Script Builder code that uses such tables is converted. However, an issue message is displayed. For example, Query Table nodes show the name of the remote database table although it doesn't appear in the Tables Resource Manager.





How to correct

There are two ways to solve this problem. If the remote table is small or there are few tables, recreate each table in IVR Designer after the conversion. Otherwise, change the Script Builder code before conversion so it can refer to remote tables as local tables. Then, copy the tables to the local machine.

SCI variables not found

Problem

During Service Creation Installation (SCI), error messages indicate that variables are not found.

Reason

A possible cause for this error is defining variables in the Script Builder application and also using them in customized external functions.

Script Builder internally stores a variable name by attaching F_ to the beginning of it (e.g., F_abc). Programmers who write custom functions may hard code the variables in the same format. The result is that the same variable name is used in both the Script Builder application and a customized external function. However, IVR Designer does not attach the F_ to variables

it stores internally. Thus, during SCI, IVR Designer cannot locate the relevant variables in custom external functions.

How to correct

Remove the F_ attached to variables in custom external functions to resolve this issue.

Variables are not spoken correctly

Problem

Variables are not spoken correctly.

Reason

Script Builder supports multiple Spoken As formats for a variable. IVR Designer supports only one Spoken As format at a time for any given variable within an application.

How to correct

In the new IVR Designer application, create a new variable for each of the Spoken As formats used by the original variable. Use Set and Test nodes to assign values to these new variables where appropriate in the call flow.

Discard Previous Touchtones always set to True

Problem

In the converted IVR Designer application, Discard Previous Touchtones for Prompt and Collect and Announcement nodes is always set to True, even when it should have been set to False.

Reason

In IVR Designer, Discard Previous Touchtones, used in the Announcement and the Prompt and Collect node, discards any touchtones entered by the caller before the node is executed. This function is equivalent to the Script Builder Type Ahead external function, which can be used to turn on and off the ability of the application to discard previous touchtones. The Type Ahead external function is not supported in IVR Designer, and it is ignored. So, in IVR Designer Announcement and Prompt and Collect nodes, Discard Previous Touchtones is always set to the default True after conversion.

How to correct

After conversion, use the Node Inspector to manually set Discard Previous Touchtones to False where appropriate.

Seasonal greetings do not work

Problem

Seasonal greetings do not work.

Reason

Seasonal greetings are not used in IVR Designer and are ignored in conversion.

How to correct

Note where seasonal greetings are used in the Script Builder application. After the application is converted, write IVR Designer code to handle these processes.

New database tables are not the most current

Problem

New database tables are not the most current.

Reason

Database tables on the Avaya IR or CONVERSANT system were modified between the time the Script Builder application was converted and the new IVR Designer application was installed.

How to correct

After installing converted database tables onto the target system, use the SQL Export function to update the new IVR Designer tables from the original tables.

Time Branch nodes show redundant date branches

Problem

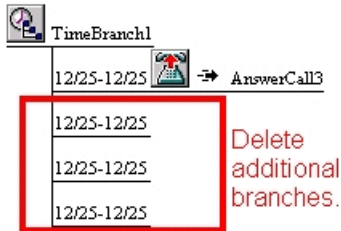
Time Branch nodes show redundant date branches in the same year. Only the first branch is converted.

Reason

Script Builder supports multiple years for a date. For example, branching decisions may be based on December 25, 2000, December 25, 2001, December 25, 2002, and December 25, 2003. IVR Designer, however, does not support multiple years for a date. In the converted application, the first branch is converted. Additional branches for the same date (but different years) are shown without the year, and no branches are converted.

How to correct

Delete the identical branches.



Custom phrase table names

Problem

The names of custom phrase tables change to names like VW_cust1 or VW_cust2. The names of standard speech tables change to names like VW_std1 or VW_std2.

Reason

During conversion, a custom speech table is assigned a new name with the format VW_custn, where n is an incrementally assigned number to provide a unique name. For example, the speech pool myspeech becomes VW_cust1 after conversion.

Standard speech pools are compared to the Enhanced Basic Speech Package during conversion. The standard speech pools convert to the name [StandardSpeech] in IVR Designer. However, any phrases added to the standard speech pool are converted to a speech table with a name in the format VW_stdn, where n is an incrementally assigned number to provide a unique name.

How to correct

Do not change the names of the new tables or the associated files will not be found on the Avaya IR or CONVERSANT system. The .pl files are created and placed on the target system during conversion.

If you must change the name of a converted phrase table or its name in the IVR Designer application, use the Avaya IR or CONVERSANT system copy phrase command to copy over the associated files.

Host definition issues

Issue

New variables appear in the converted IVR Designer application. These new variables resemble host field names in the original Script Builder application.

Reason

In Script Builder, host field names can be used just like variables in operations. In IVR Designer, separate variables must be used. During conversion, new variables are created, using the original host field names with unique ID numbers appended.

When host definitions are converted to IVR Designer, these field names are brought across. However, IVR Designer creates a new variable corresponding to each host field name and

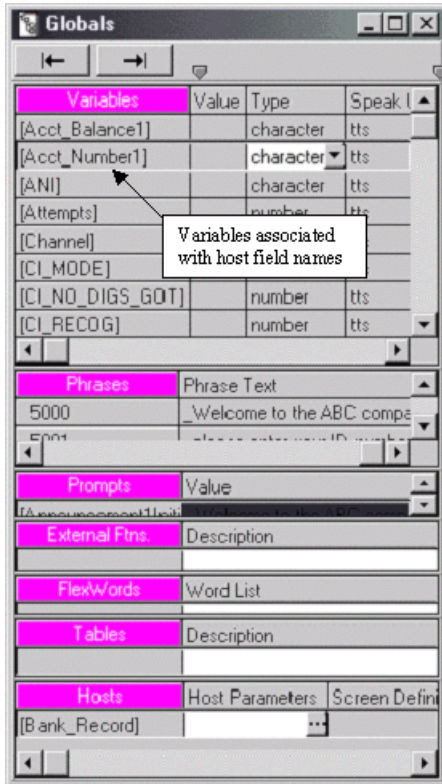
appends a unique ID number to each variable. For example, the Script Builder host field name `Acct_Balance` is created as a variable named `Acct_Balance1`. The converted code is automatically updated to handle these new variables in operations.

The dialog box titled "Host Definition Editor - '[Bank_Record]'" has four tabs: "Host Definition", "Logins", "Screen Identification", and "Field Definition". The "Field Definition" tab is active. It contains a "Screen Selection" section with a "Screen Name" dropdown set to "Accounts", a "Blank Screen" checkbox set to "False", and a "Base Type" dropdown set to "Transaction". Below this is a table with columns: Name, ID, Location, Row, Column, Length, Direction, Class, and Format. The table contains two rows: "Acct_Number" and "Acct_Balance". A callout box labeled "Host field name" points to the "Acct_Balance" row.

Name	ID	Location	Row	Column	Length	Direction	Class	Format
Acct_Number	1	Regular	10	1	5	From Host	Number	NZ
Acct_Balance	1	Regular	10	10	5	From Host	Number	NZ

Host field name

OK Cancel Apply



Issue

Shared Hosts have become local hosts.

Reason

Shared hosts in Script Builder applications are converted to local hosts in the IVR Designer application.

Breaking a long application into sections

When a Script Builder application is converted to a IVR Designer application, the application is created as a single call flow, even if the application is very large and complex. However, IVR Designer gives you the ability to break up long applications into sections, called embedded call flows. Breaking a long application into sections makes it easier to read. For more information about and practice using embedded call flows, see [Using embedded call flows in TAS applications](#) on page 400.

Chapter 10: Installing an application on the target system

After you have created, generated, and transferred your IVR Designer application to the target system, you must install the application and assign it to channels or telephone numbers before you can use it. For more information about and practice installing an IVR Designer application, see [Installing TAS applications](#) on page 338.

The CONVERSANT and Avaya IR systems include a set of Service Creation Installation (SCI) tools to help you accomplish this task. SCI refers to the collection of utilities you use to install your application and the resources it needs on the target system after the files have been generated and transferred.



Note:

You must log in as root to use the SCI tools on the target voice response system.

There are four utilities included in the SCI tools:

- [Using the sci command](#) on page 292 – Installs the application on the target system.
- [Removing an application](#) on page 294 – Removes an application, speech files, or database tables from the target system.
- [Importing or removing a professionally recorded speech file](#) on page 295 – Imports or removes professionally recorded speech files to or from the target system.
- [Transferring files from floppy disks to the target system](#) on page 267 – Transfers an application from floppy disks to the target system.

You can also assign the application to channels using the Web Administration tool on the Avaya IR system or the `cvis_menu` on a CONVERSANT system. For information about installing an application and assigning it to channels or telephone numbers on the target system, refer to the product documentation for the target voice response system.

Related topics:

[Preparing to install an application](#) on page 290

[Installing the application](#) on page 290

[Assigning the installed application using the Application Install tab](#) on page 293

[Removing an application](#) on page 294

[Importing or removing a professionally recorded speech file](#) on page 295

[Transferring an application from floppy disks](#) on page 296

Preparing to install an application

Before you can install an application on the target system:

1. Create your application using IVR Designer.
2. Generate the source code using the Code Generation tool.
3. Transfer the application to the target system using the Application Transfer tool.



Note:

When you transfer the application, IVR Designer places it in the following directory on the target system: `/lt/trans/sc/application_name/`, where `application_name` is the name of the application.

Installing the application

IVR Designer offers the following options for installing applications on the target system:

- Installing the application from within IVR Designer using the Application Install tab in the Code Generation/Application Transfer window
You can also use the Application Install tab to assign the installed application to one or more channels on the target system.



Note:

The Application Install tab is available only if the Network transfer type is selected from the Options menu of the Application Transfer tool.

- Installing the application using the `sci` command at the target system or using telnet

Related topics:

[Using the Application Install tab](#) on page 290

[Using the sci command](#) on page 292

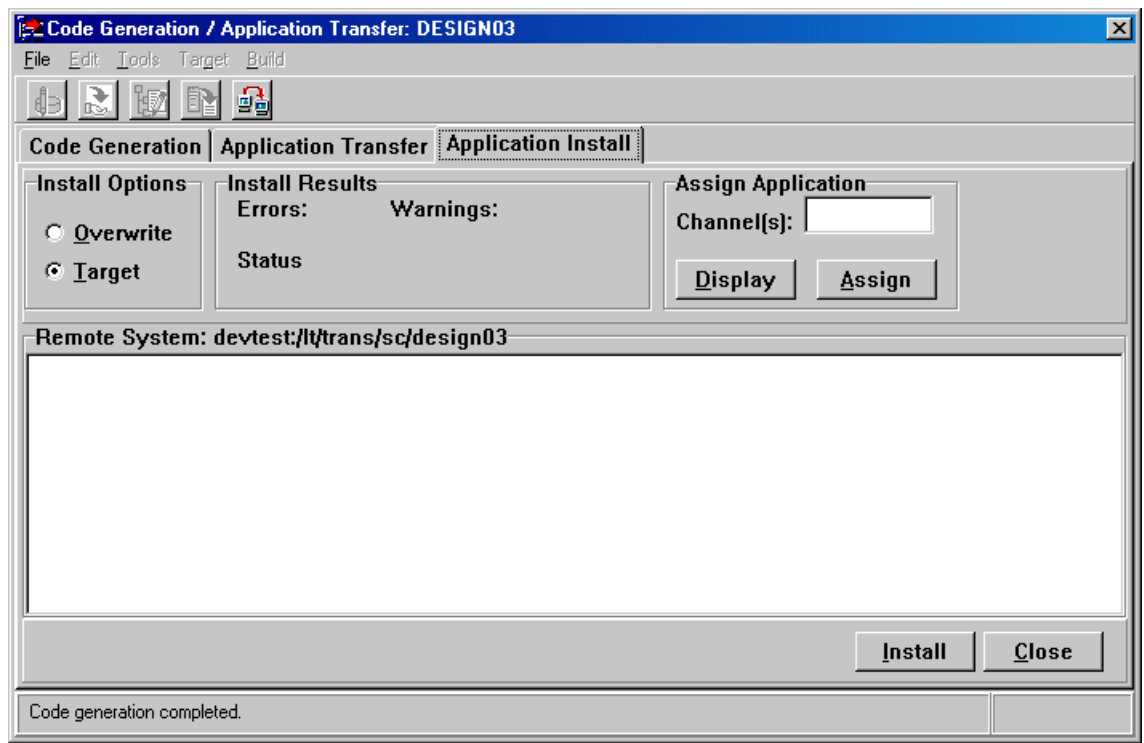
Using the Application Install tab

To install an application on the target system using the Application Install tab in the Code Generation/Application Transfer window:

1. Click the Application Install tab in the Code Generation/Application Transfer window. IVR Designer displays the Application Install pane.

Note that the Remote System: field displays the name of target system on which the application is to be installed and the name and installation pathname for the

application to be installed. For example, in the following figure, the application is to be installed on the target system named devtest in the directory path /lt/trans/sc/design03.



2. From the Install Options box, select the appropriate installation option:
 - If you want to overwrite existing resources with matching names, such as database tables, host definitions, speech pools, and speech recognition grammars, click Overwrite.
This is the same as using the -O option in the sci command.
 - If you do not want to overwrite existing resources with matching names but would rather use existing resources, click Target.
This is the same as using the -T option in the sci command.
 - If you want to be prompted when there are file name conflicts, before either overwriting resources or using existing resources, use the -I option of the sci command. See [Using the sci command](#) on page 292 for information about the -I option.
3. Click Install.
During the installation, the Application Install pane displays the following information:
 - The Install Results box displays the current status of the procedure, as well as the number of errors and warnings encountered during installation.
 - The Remote System: field displays general information about the installation, including information about errors and warnings. You can scroll through the displayed text to get more information about errors or warnings.



Note:

The success or failure of the installation is based upon whether the TAS compiler can successfully compile the application on the target system. The Errors and Warnings count messages displayed in the Install Results box are based on messages from the entire sci process, including database table, host definition, and speech interface components. Thus, an application can install successfully yet still generate errors and warnings. For example, during application installation, a database table error might occur, which would cause the Errors count to be 1. The application as a whole, however, could still install successfully, in which case the status message would read Install successful. This means that the application is installed and usable, but actions using the affected database table would not work correctly.

Using the sci command

To install an application on the target system using the sci command:

1. Log in to the target system as root.
2. At the system prompt, enter some form of the following command:
`sci [-O | -T | -I] [-t] application_name`

Description

- sci (Service Creation Install) is the command to install an application and application resources on the target system.
- [-O | -T | -I] are options. You may use only one of these three options at a time.
 - O overwrites existing resources on the target system with matching names without warning you. Examples of resources affected by this option include database tables, host definitions, speech pools, and speech recognition grammars.
 - T uses existing resources on the target system with matching names without warning you.
 - I when there are file name conflicts, the system prompts the user before either overwriting resources or using existing resources. This is the default if no option is entered as part of the command.
- [-t] is an option. This option turns off tracing for Trace Variables nodes.
- application_name is the name of the application.

Examples

To install an application named acct_bal using the resources that exist on the target system, and to turn off tracing for Trace Variables nodes, enter:

```
sci -T -t acct_bal
```

To install an application named acct_bal , overwrite the resources on the target system, and to leave tracing on for Trace Variables nodes, enter:

```
sci -O acct_bal
```

The install.log file

During the sci installation process, a file named install.log is produced. The purpose of the install.log file is to provide developers and support personnel with information about the installation and compilation of an IVR Designer application. Although the install.log file displays error messages from Oracle, TAS, and other CONVERSANT or Avaya IR software, it does not provide any explanation about the cause of or solution for the error. For information about causes of and solutions for errors, see the error message documenting for your system.

Restarting the database DIP

If you are installing database tables as part of the application installation and using the interactive mode (-I option), the sci tool asks if you want to restart the database data interface process (DIP).

If all of the tables are completely new tables, or if you have not changed the table schema for any of the tables, select No.

If you have changed the table schema on any existing table since the application using it has been run, and tables have been accessed in that application, select Yes.



Caution:

If you select Yes to restart the database DIP, any calls that attempt to access a database for that application while the database DIP is being restarted will fail. For this reason, you should plan to restart the database DIP at a time when call volume is low, if possible.

Assigning the installed application using the Application Install tab

After you have installed the application on the target system, you can also assign the application to one or more channels on the target system, using the Application Install tab of the Code Generation/Application Transfer window.

To assign the application using the Application Install tab:

1. (Optional, but recommended) Check to see if the channel or channels to which you want to assign the application are available for assignment on the target system.

To check channels:

- Enter the number or numbers of the channels you want to check in the Channel(s): field.
Use a hyphen to separate the first and last numbers of a series (for example, 4-6). Use commas to separate multiple entries (for example, 1,4-6).

To check all channels, leave this field blank.

- Click Display.

The Remote System: field displays information about the selected channel or channels.

2. In the Channel(s): field, enter the number or numbers of any channels to which you want to assign the application.
3. Click Assign.
The system assigns the application to the designated channels and the Remote System: field displays a message confirming the assignment.

Removing an application

To remove an application and application resources from the target system using the SCI tools.

1. Log in to the target system as root.
2. At the system prompt, enter some form of the following command:
`scrm -[d | i | s | t | a] application_name`

Description

In the scrm command above:

- scrm (Service Creation ReMove) is the command to remove an application and application resources from the target system.
- -[d | i | s | t | a] are options. You may use any combination of these options.
 - d removes local database tables
 - i removes installed files
 - s removes speech files
 - t removes transaction files
 - a removes all files associated with the application
- application_name is the name of the application.

Examples

To remove all database and speech files from an application named acct_bal, enter:

```
scrm -ds acct_bal
```

To remove all files associated with an application named acct_bal, enter:

```
scrm -a acct_bal
```

Importing or removing a professionally recorded speech file

Avaya recommends that you use professionally recorded speech in your IVR Designer applications. Professionally recorded speech can be obtained from Avaya or from other sources.

Before you begin

Before you can import a professionally recorded speech file to the target system, you must:

1. Create the application using IVR Designer. IVR Designer creates a list of all the required speech phrases during the Code Generation process.
2. Obtain professionally recorded speech files on floppy disk.
3. Generate the source code using the Code Generation tool.
4. Transfer the application to the target system using the Application Transfer tool.

Procedure

To import or remove a recorded speech file to or from the target system using the SCI tools:

1. Log in to the target system as root.
2. If you are importing a recorded speech file, insert the floppy disk containing the file into the floppy disk drive.
3. At the system prompt, enter some form of the following command:
`scsp [-o] [-r] application_name phrase_list_filename`

Note: The speech file is copied to the following directory: `/lt/trans/sc/application_name /pro_speech/phrase_list_filename`

Description

In the `scsp` command above:

- `scsp` is the basic command used to import or remove the recorded speech file to or from the target system.
- `[-o]` is an option. This option overwrites files with matching file name without warning the user.
 When transferring an application with a name that matches an application that already exists on the target system, Avaya recommends that you first remove the old application files using the `scrm` command.

For example, suppose you had an application named `abc_auto` on the target system, and you later developed another, totally different application to replace it. In the earlier `abc_auto` application, you used certain FlexWord word lists and database tables. In the newer version, you do not to use the same FlexWord word lists and database tables.

When you transfer the new version of `abc_auto` to the target system, only the files with matching file names are replaced. The old FlexWord lists and database tables are still there. This can cause conflicts when using the newer version.

To avoid this problem, first remove the old application (or the appropriate application files) from the target voice-response system, using the `scrm` command. Then you can transfer and install the new application.

- `[-r]` is an option. This option removes the recorded speech file from the application.
- `application_name` is the name of the application.
- `phrase_list_filename` is the file name of the recorded speech phrase list file. This is required because an application may have multiple phrase list files.



Note:

You must define a phrase list file for your recorded speech and generate it using the Code Generation tool. This assigns it a `.pl` extension. The floppy disk with the professionally recorded speech must also contain a file called `Backup.pl` that contains the phrase list for the speech. This phrase list should be the same as the one you generated in your application. The actual recorded speech must be in files formatted with the phrase number and a `.phr` extension (for example, `1000.phr`). A file called `Sequence` must contain the number of floppies in the set, with the last floppy containing the word "LAST". After all the phrases are copied from the floppy disk, if there is a `.pl` file in `/speech/talk` that matches what was copied in, the speech will be loaded into the talkfile.

Examples

To import a recorded speech file named `greeting` into an application named `acct_bal`, without being prompted if the target system already has a speech file with that name, enter:

```
scsp -o acct_bal greeting
```

To remove a recorded speech file named `greeting` from an application named `acct_bal`, enter:

```
scsp -r acct_bal greeting
```

Transferring an application from floppy disks

IVR Designer lets you transfer your finished applications to the target voice response system using floppy disks. See [Transferring files from floppy disks to the target system](#) on page 267 for the procedure.

Chapter 11: TAS application development tours overview

TAS application development tours provide detailed examples to help you learn how to use Avaya IVR Designer features in your applications.

- Basic TAS 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 TAS 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 [Basic TAS application development tours](#) on page 297.

Related topics:

[Basic TAS application development tours](#) on page 297

[Advanced TAS application development tours](#) on page 436

Basic TAS application development tours

Basic TAS application development tours walk you through some of the key tasks that you perform in creating Avaya IVR Designer applications. It is recommended that you complete the tours in sequence, if possible.

In some of these tours, you build on applications from other tours or on tour applications you loaded with the IVR Designer sample applications. The tour applications are contained in the Program Files/Avaya/Avaya IVR Designer/Tour Applications/TAS folder.

You can view and print the entire Basic TAS application development tours section in PDF format.

To continue, see [TAS IVR Designer interface](#) on page 298.

Related topics:

[TAS IVR Designer interface](#) on page 298

[Configuring the TAS IVR Designer desktop](#) on page 307

[Creating a basic TAS application](#) on page 312

[Testing a TAS application](#) on page 330

[Generating code and transferring and installing a TAS application](#) on page 333

[Sharing TAS resources](#) on page 342

[Branching based on time, day, or date in TAS applications](#) on page 348

[Copying TAS nodes and applications](#) on page 355

[Presenting menus to callers in TAS applications](#) on page 359

[Collecting information in TAS applications](#) on page 368

[Using calculations in TAS applications](#) on page 379

[Using variables and speech in TAS applications](#) on page 385

[Using embedded call flows in TAS applications](#) on page 400

[Creating database applications](#) on page 408

TAS IVR Designer interface

The TAS IVR Designer interface tour introduces you to the IVR Designer interface. It provides an overview of the key elements of the interface used to develop applications. Throughout the tour, you may click links to go to the Avaya IVR Designer Help, which provides detailed information.

To continue, see [TAS Menus](#) on page 299.

Related topics:

[TAS Menus](#) on page 299

[TAS Toolbar](#) on page 302

[TAS Node Palette](#) on page 302

[TAS Globals](#) on page 303

TAS Menus

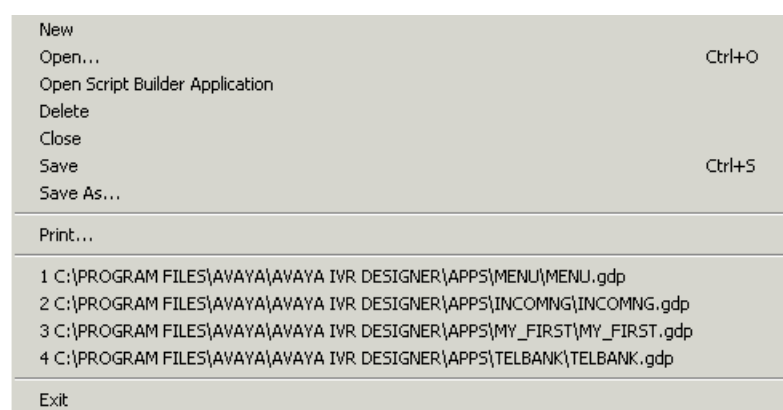
The IVR Designer menu bar, shown in the following figure, provides access to the menus that you use to develop applications.



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.



Note the following when using the File menu:

- Only one IVR Designer application may be open at a time.
- When you open Script Builder applications, they are automatically converted to IVR Designer format. Depending upon the complexity and features used in the original Script Builder application, the Script Builder Conversion tool may not be able to convert the application completely. You may have to do some initial preparation and follow-up reconstruction. See Script Builder applications for detailed information.

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.

Undo	Ctrl+Z
Redo	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Configure Toolbar...	
Configure Node Palette...	
Configure External Function Palette...	
Find...	

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, Node Palette, and External Function 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
External Function Palette
Globals
Variables
Phrases
Prompts
FlexWords
Hosts
External Functions
Tables

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 the Globals Manager.
- Clicking a resource manager (for example, Variables or Prompts) displays the resource manager, if it is undocked. Managers must be undocked before you can display them individually.

Window Menu

Use the Window menu, shown in the following figure, to set up how call flow windows are displayed.

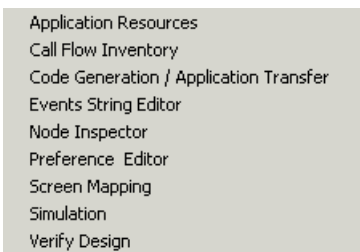


Note the following about using the Window menu:

- The menu appears only if you select 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.



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.

Sharing Menu

Use the Sharing menu, shown in the following figure, to update or refresh shared resources.

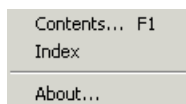


Note the following about the Sharing menu:

- You may update resources that you have shared out to be used by other developers.
- You may refresh resources that you shared in from other developers.

Help Menu

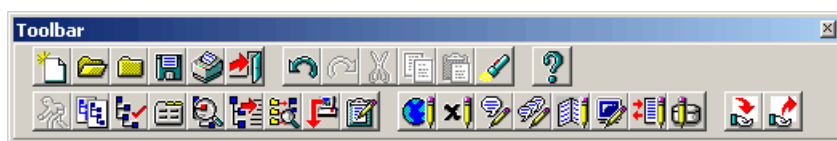
The Help menu, shown in the following figure, provides access to the Avaya IVR Designer Help topics.



To continue, see [TAS Toolbar](#) on page 302.

TAS 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 find most Toolbar functions using the IVR Designer Menu bar. See [Toolbar icons](#) on page 538 for explanations of each Toolbar icon.



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 TAS IVR Designer desktop](#) on page 307 provides practice in configuring the Toolbar.

To continue, see [TAS Node Palette](#) on page 302.

TAS 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 Answer Call, Disconnect Call, and Announcement.



To add a node object, drag it onto the design pad. Node objects save you time because the programming for the function represented by the icon is done for you.

Node Palette configuration

You can:

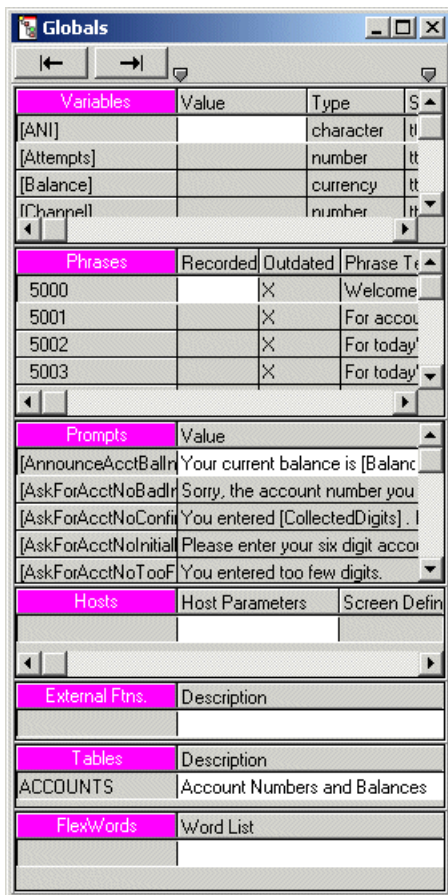
- Undock the Node Palette from its default location under the Menu bar, move it to another location, size it, and hide it. (The Node Palette shown above is undocked.)
- Dock the Node Palette to its default location.
- Select the icons you want to include on the Node Palette and select the order in which they are arranged.

[Configuring the TAS IVR Designer desktop](#) on page 307 provides practice in configuring the Node Palette.

To continue, see [TAS Globals](#) on page 303.

TAS 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 the Globals Manager.

[Configuring the TAS IVR Designer desktop](#) on page 307 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.

Variables	Value	Type	Speak Using	Spoken As	Phrase Table	Input As	Length	Default	System
[ANI]		character	tts	A		char	10		true
[Attempts]		number	tts	AN		#	10	0	true
[Channel]		number	tts	AN		#	10	1	true
[MNDFl]		number	tts	AN		#	10	0	true

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.

Prompts	Value
[NewYearGreetingIn]	Happy New Year. You have reached the Big Bank information line. We are now closed. Your call
[TransferMessage]	Please wait while your call is being transferred.
[WeekendGreetingIn]	You have reached the Big Bank information line. We are now closed. Your call is being forward...

With the Prompts Manager, you can:

- Create new prompts.
- Rename and revise existing prompts.

Phrases Manager

Use the Phrases Manager, shown in the following figure, to work with the phrase tables that store standard and custom phrases for use in IVR Designer applications. The Phrases Manager displays one phrase table at a time.

Phrases	Recor	Outda	Phrase Text
5000	X		Welcome to Big Bank.
5001	X		We are now closed.
5002	X		Your call is being forwarded to our twenty-four hour message center.
5003	X		Please wait while your call is being transferred.
5004	X		Thank you for calling.

With the Phrases Manager, you can:

- Create new phrases and edit existing phrases.
- Share phrase tables with other application developers.
- Record phrases or import previously recorded phrases.

FlexWords Manager

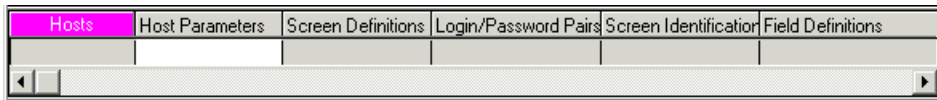
FlexWord speech recognition is supported on CONVERSANT V8 and R9 systems.

FlexWord speech recognition recognizes words by matching phonemes (parts of words).

Avaya IVR Designer FlexWord options are available only if the FlexWord speech recognition package is purchased and installed on the CONVERSANT system.

You use the FlexWords Manager, shown in the following figure, to work with FlexWord lists.

target system. As part of defining the interface, you use the Hosts Manager, shown in the following figure, to manage host definitions in IVR Designer.



With the Hosts Manager, you can:

- Create, edit, and rename host definitions.
- Share host definitions with other application developers.
- View host definitions that you have imported using the Application Resources tool.

External Functions Manager

External functions provide capabilities beyond those available from the basic node objects in IVR Designer. For example, the Answer Call node is used in applications to take incoming calls. The Make_Call external function may be used to place an outgoing call.

The actual external functions reside on the target Avaya IR or CONVERSANT system, not in IVR Designer. The External Functions Manager, shown in the following figure, displays the names of and brief descriptions for all external functions defined for the current application.



With the External Functions Manager, you can:

- Define the interface to new external functions, which you create on the Avaya IR or CONVERSANT systems.
- Edit, rename, and customize existing external function interfaces.
- View external functions that you have imported using the Application Resources tool.

Tables Manager

Database tables have two primary uses in IVR Designer applications:

- To store input, such as account numbers or order selections
- To provide information stored in a database, such as an account balance or order status

The Tables Manager, shown in the following figure, lists and describes the tables included in the current application.



With the Tables Manager, you can:

- Create, modify, and rename tables.
- Share tables with other application developers.
- Edit the schema of a table by making changes to fields.
- Edit data in a table by making changes to data records.

To continue, see [Configuring the TAS IVR Designer desktop](#) on page 307.

Configuring the TAS 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:

- Your individual preferences
- Application and project requirements

Configuring the TAS IVR Designer desktop 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 TAS interface elements](#) on page 307.

Related topics:

[Configuring TAS interface elements](#) on page 307

[Setting TAS preferences](#) on page 310

Configuring TAS 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

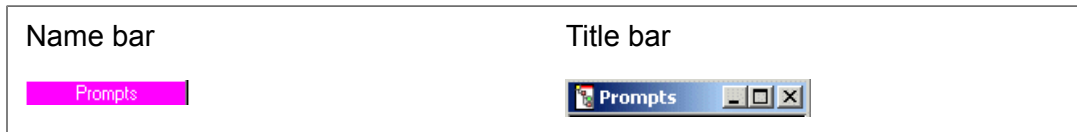
As with any Windows interface, you may minimize, maximize, and close elements by clicking the appropriate buttons. You size and shape windows by clicking and dragging their borders.

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 from 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.



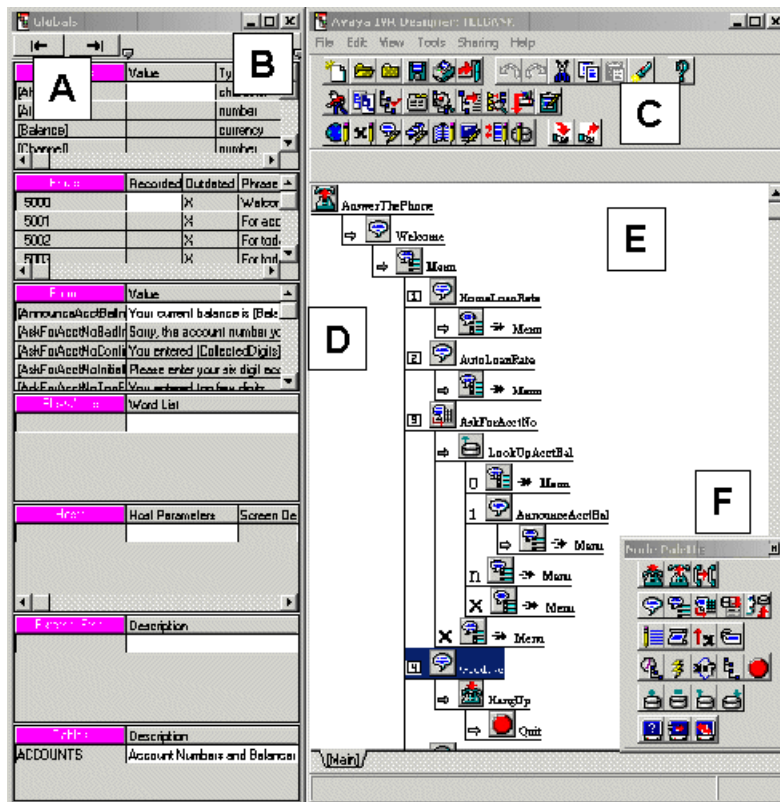
To undock the Toolbar, click a blank area or an inactive icon and drag the Toolbar away. To dock the Toolbar, click it as in undocking, drag the Toolbar over the menu bar, and release it. If you do not place the Toolbar over the menu bar, it does not dock. You dock and undock the Node Palette in the same way.

Displaying and hiding

Except for the menu bar, you may hide elements on the IVR Designer interface. To hide elements, you 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. This figure shows how elements of the workspace can be adjusted.



A. You can click  to enlarge the Globals Manager and click  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 use the Multiple Document Interface mode, you can move individual call flow windows within the design pad. The design pad above shows a 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 > TAS 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 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.
 8. 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.

[Setting TAS preferences](#) on page 310 provides practice in setting up the Preference Editor.

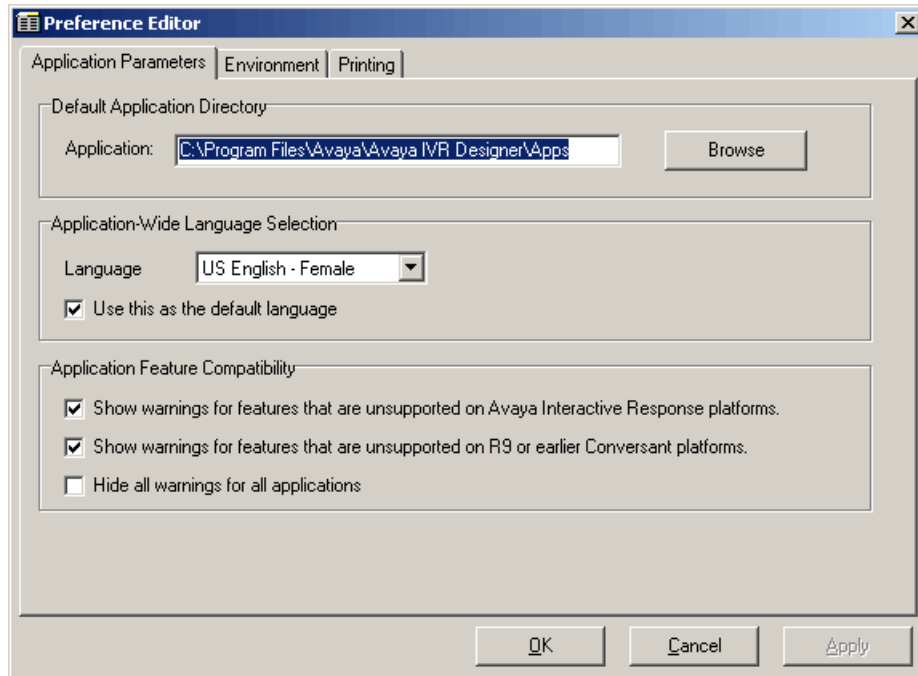
To continue, see [Setting TAS preferences](#) on page 310.

Setting TAS preferences

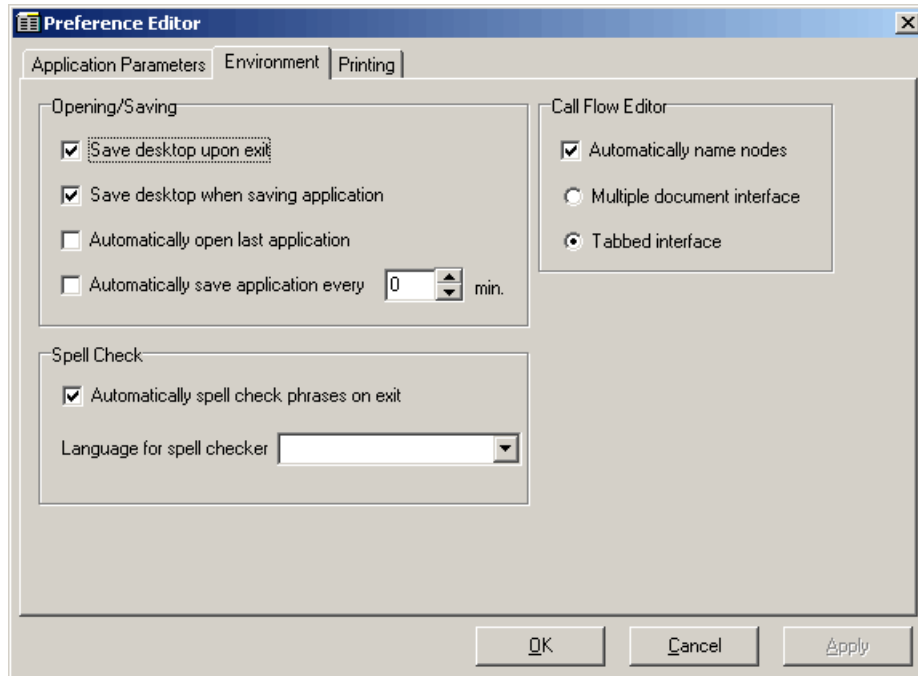
The Setting TAS preferences topic introduces you to the Preference Editor and the options it provides.

1. Click the Preference Editor  icon on the Toolbar.

IVR Designer displays the Preference Editor, shown in the following figure.



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.
 - Call Flow Editor options allow you 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 TAS application](#) on page 312.

Creating a basic TAS application

The topics in Creating a basic TAS application provide a step-by-step example of how to design and build a simple application using IVR Designer.

To continue, see [TAS Application design overview](#) on page 313.

Related topics:

[TAS Application design overview](#) on page 313

[Creating a TAS application](#) on page 318

[Specifying parameters for TAS applications](#) on page 326

TAS Application design overview

IVR Designer can be used to develop voice applications that can be run on either the Avaya IR or CONVERSANT system. The recommended workflow for producing voice 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 and simulate the application.
5. Generate the source code file to transfer to the target Avaya IR or CONVERSANT 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 the needs of your client.

To continue, see [Scripting an application](#) on page 313.

Related topics:

[Scripting an application](#) on page 313

[Developing a step analysis](#) on page 315

[Creating a flowchart for an application](#) on page 317

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, 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 present three options:

- Provide information on auto loan interest rates.
- Transfer the caller to someone for assistance in obtaining an auto loan.
- Provide account information.

Translate these requirements into a typical call:

- Agent: Hello, Big Bank. How may I help you?
- Caller: What's your current interest rate on auto loans?
- Agent: (Refers to chart of interest rates) That's 7.9%. May I help you with anything else?
- Caller: Yes, I'd like to check my checking account balance.
- Agent: What's your account number?
- Caller: It's 12345678.
- Agent: And the last four digits of your Social Security Number?
- Caller: 5555.
- Agent: One moment. (Looks up the balance). Your balance is \$2010.27. May I help you with anything else?
- Caller: Well, I'd like to speak to somebody about an auto loan.
- Agent: One moment and I'll transfer you. (Transfers call then hangs up.)

Now look at our sample banking call again. This time, 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? **Answer telephone, announce greeting, prompt for information.**
- Caller: What's your current interest rate on auto loans? **State request for information.**
- Agent: (Refers to chart of interest rates) That's 7.9%. May I help you with anything else? **Look up requested information, announce requested information, prompt for information.**
- Caller: Yes, I'd like to check my checking account balance. **State request for information.**

- Agent: What's your account number? Based on request, prompt for further information.
- Caller: It's 12345678. Provide information requested.
- Agent: And the last four digits of your Social Security Number? Based on request, prompt for further information.
- Caller: 5555. Provide information requested.
- Agent: One moment. (Looks up the balance). Your balance is \$2010.27. May I help you with anything else? Use provided information to look up requested information, announce requested information, prompt for information.
- Caller: Well, I'd like to speak to somebody about 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.

Note the basic types of interaction in this call:

1. Agent answers the telephone.
2. Agent makes announcement.
3. Agent prompts the caller for information.
4. Caller requests information.
5. Agent looks up the requested information.
6. Agent reports the information to the caller.
7. Agent repeats steps 2 through 6 until the caller's request requires call transfer.
8. Agent announces transfer.
9. Agent transfers the call.
10. 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 (for example, account number or Social Security number), and specific actions he or she must take (searching for loan rates and account balances). Later in this course you practice working with more detailed requirements statements, and developing more complex applications.

To continue, see [Developing a step analysis](#) on page 315.

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:

STEP	Agent	Caller
Answer telephone.	x	
Announce greeting.	x	
Prompt for information.	x	
State request for information.		x
Look up requested information.	x	
Announce requested information.	x	
Prompt for information.	x	
State request for information.		x
Based on request, prompt for further information.	x	
Provide information requested.		x
Based on request, prompt for further information.	x	
Provide information requested.		x
Look up requested information.	x	
Announce requested information.	x	
Prompt for information	x	
State request for information.		x
Announce transfer.	x	
Transfer call appropriately.	x	

Disconnect.	x	

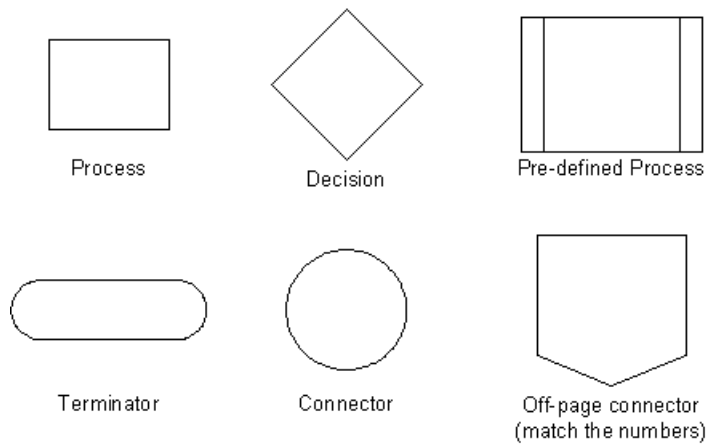
This logical step analysis describes the flow of the call transaction. When you break a call down by speaker, it is easy to include all of the interactions between the caller and the agent.

To continue, see [Creating a flowchart for an application](#) on page 317.

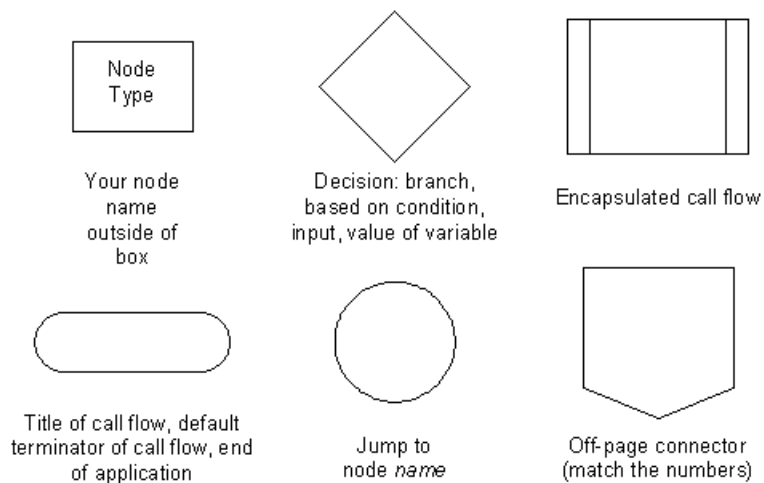
Creating a flowchart for an application

Because an IVR Designer 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.

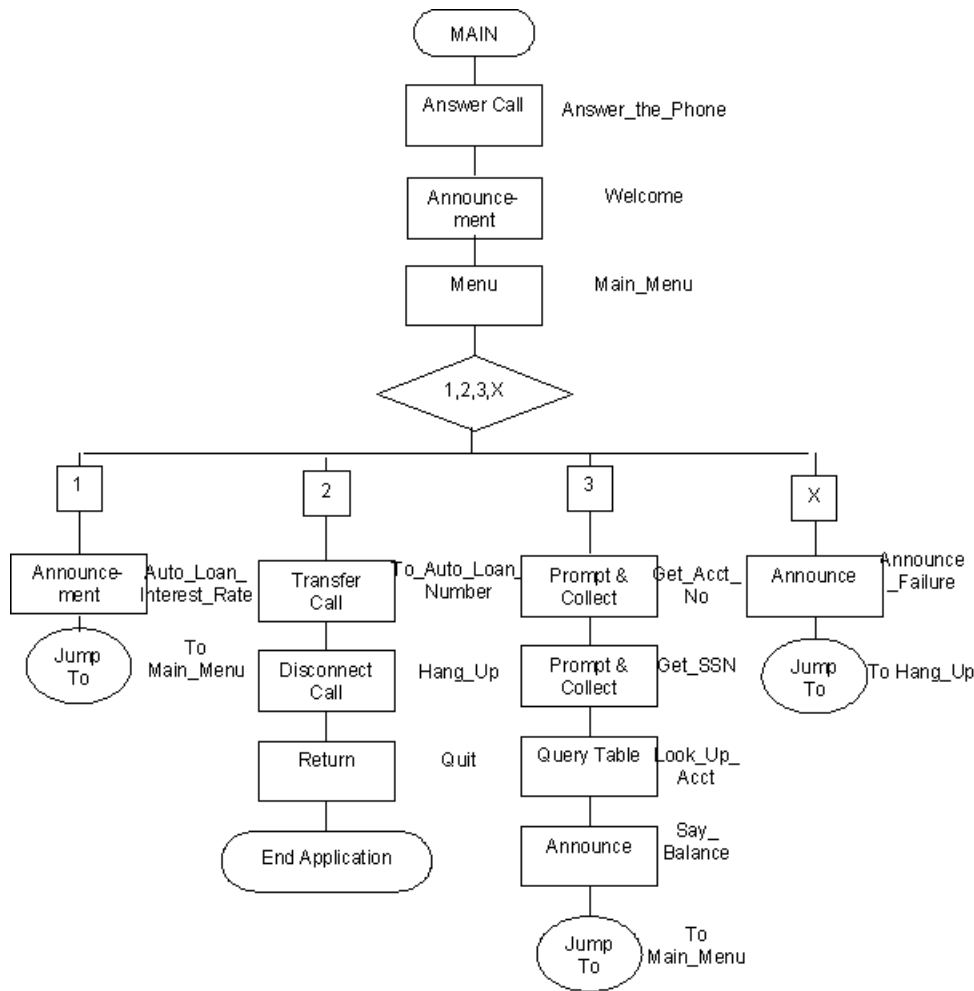
You may be familiar with the standard flowcharting symbols. They are shown in the following figure:



Modified versions that can be used to specify IVR Designer call flows are shown in the following figure:



With these modified symbols, the flowchart for the step analysis would look like this:



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 TAS application](#) on page 318.

Creating a TAS application

The application you develop in Creating a TAS application does the following:

- Answers the telephone.
- Plays a greeting announcement to the caller.
- Transfers the call to another number and announces the transfer to the caller.

- Notifies the caller if the call cannot be transferred successfully, before disconnecting the call.
- Disconnects and terminates the call.

You begin your application development by adding nodes.

To continue, see [Creating a new TAS application](#) on page 319.

Related topics:

[Creating a new TAS application](#) on page 319

[Adding an Answer Call node](#) on page 320

[Adding an Announcement node](#) on page 320

[Adding a Transfer Call Node](#) on page 321

[Adding a Disconnect Call node](#) on page 322

[Adding a Return node](#) on page 323

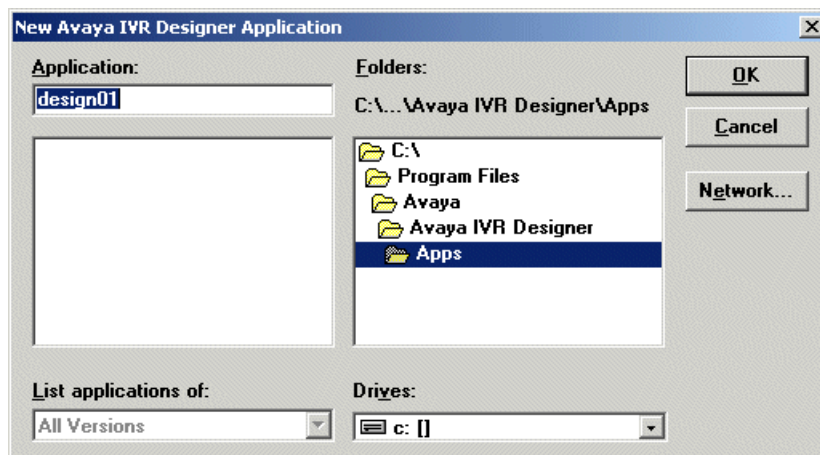
[Adding a Transfer Fail Announcement node](#) on page 323

[Using the Jump to Node function](#) on page 324

Creating a new TAS application

Before you can begin building your first application, you must create it. To create a new TAS 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. In the Application: field, replace the default name with my_first.
3. Click OK. IVR Designer creates a new TAS application called my_first.
4. 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.

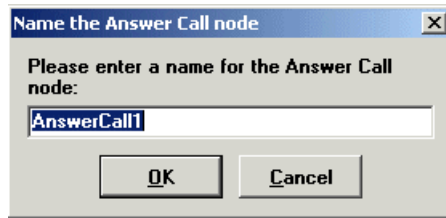
You are now ready to begin developing your application. The most basic task in developing an application is building one or more call flows by dragging the appropriate nodes onto the design pad.

To continue, see [Adding an Answer Call node](#) on page 320.

Adding an Answer Call node

To add an Answer Call node to the call flow:

1. Locate the Answer Call  icon on the Node Palette and drag it onto the design pad.
IVR Designer displays the node naming dialog box. If the Automatically name nodes check box is selected in the Preference Editor, IVR Designer suggests a name for the node, shown in the following figure.



2. Type answer call 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, you must first double-click it to highlight it, and then replace it by typing in the new name.
3. Click OK.
IVR Designer adds the new node to the design pad display, shown in the following figure, and opens the Node Inspector, if it is not already open.



Notice that IVR Designer uses upper case for the initial character of each word in the node name 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 will define the nodes after placing all nodes in the call flow.

To continue, see [Adding an Announcement node](#) on page 320.

Adding an Announcement node

To add an Announcement node:

1. Locate the Announcement  icon on the Node Palette and drag it onto the empty branch of the AnswerCall node.
You may need to move the Node Inspector window to get to the Announcement node or other nodes.
2. Type say hello 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 Announcement node onto the empty branch of the AnswerCall node, the Announcement 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.

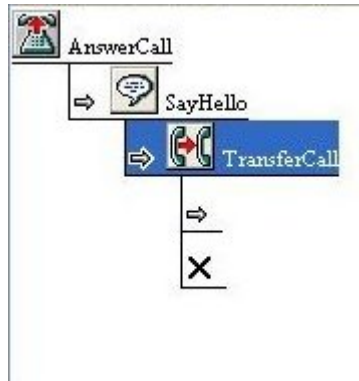
If the nodes in a call flow are not connected, the application works only as far as the disconnected node. Any nodes after the disconnected node do not execute. This is true of the application both in Simulation mode in IVR Designer and in operation on the target voice response system.

To continue, see [Adding a Transfer Call Node](#) on page 321.

Adding a Transfer Call Node

To add a Transfer Call node:

1. Locate the Transfer Call  icon on the Node Palette and drag it onto the empty branch of the SayHello node.
IVR Designer displays the node naming dialog box.
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.



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.

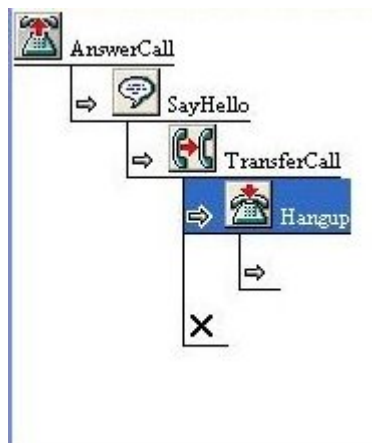
The second branch, with the X symbol, is the fail branch. This is the branch that is taken if the call cannot be successfully transferred.

To continue, see [Adding a Disconnect Call node](#) on page 322.

Adding a Disconnect Call node

To add a Disconnect Call node:

1. Locate the Disconnect Call  icon on the Node Palette and drag it onto the empty pass branch of the TransferCall 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.

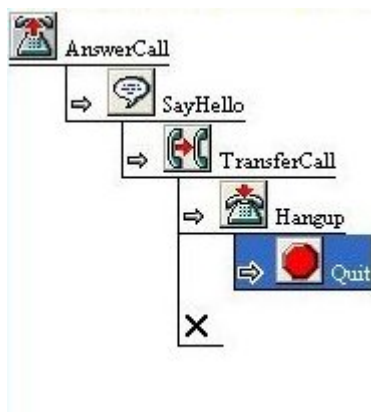


To continue, see [Adding a Return node](#) on page 323.

Adding a Return node

To add a Return node:

1. Locate the Return  icon on the Node Palette and drag it onto the empty branch of the Hangup 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.



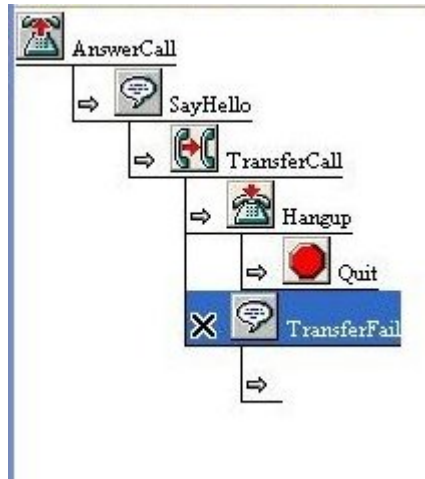
The purpose of the Return node is not as obvious as the purpose of other nodes you have used. In the [Main] call flow, the Return node terminates the application script and allows the target system to reset itself for another caller. Notice that the Return node has no branches beneath it.

To continue, see [Adding a Transfer Fail Announcement node](#) on page 323.

Adding a Transfer Fail Announcement node

To add another Announcement node:

1. Drag the Announcement icon to the empty fail branch of the TransferCall 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.



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.

To continue, see [Using the Jump to Node function](#) on page 324.

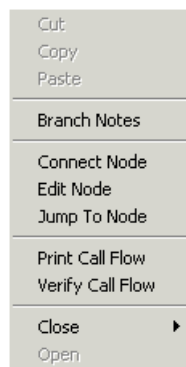
Using the Jump to Node function

The application does not know how to finish the call if the transfer fails. You must direct the application to disconnect the call and end the script, much as you 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.



One of the powerful features of IVR Designer 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.

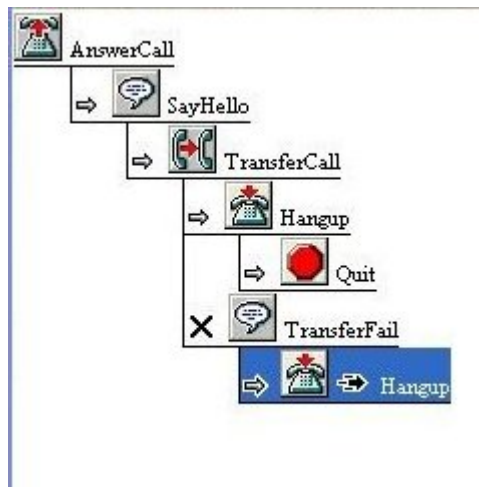


This dialog box displays all the nodes used in the current call flow.

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 Hangup node.



The ➞ symbol indicates that the node branch jumps to another node.

To continue, see [Specifying parameters for TAS applications](#) on page 326.

Specifying parameters for TAS applications

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 TAS application](#) on page 318, the following node parameters must be defined:

- 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.

To continue, see [Defining an Announcement node prompt](#) on page 326.

Related topics:

[Defining an Announcement node prompt](#) on page 326

[Setting Transfer Call node options](#) on page 328

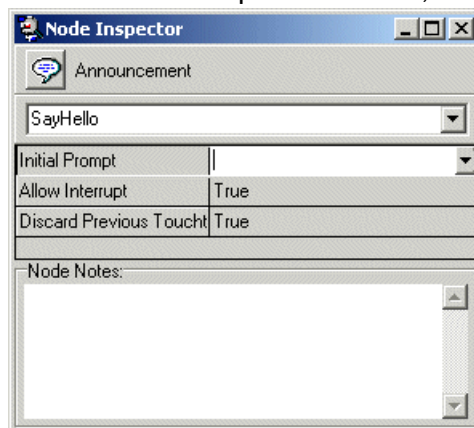
[Defining a Transfer Call node failure announcement](#) on page 329

Defining an Announcement node prompt

As it stands now, the application is set to play an announcement to the caller after answering the telephone. 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. Click the Node Inspector window, shown in the following figure, to bring it to the front.

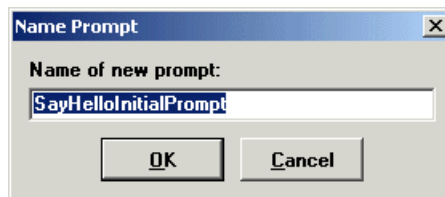


If the Node Inspector is totally hidden by the main window, you must bring it to the front. To avoid this extra step, position the Node Inspector so that its window is

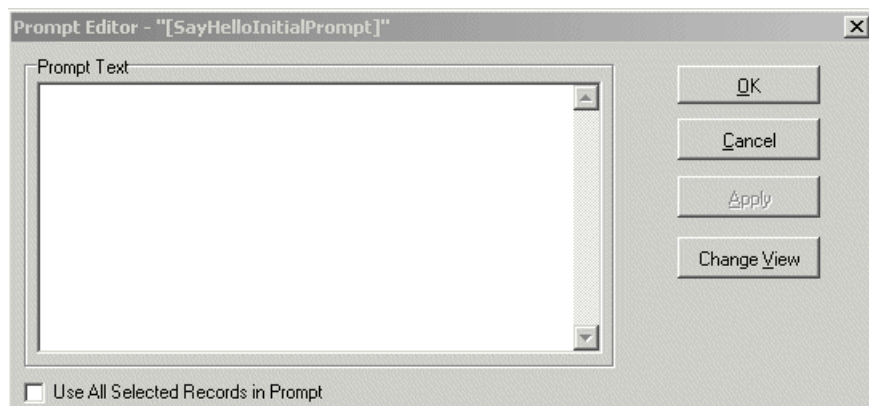
visible outside the main window. Then you simply click the Node Inspector window to bring it to the front. Resize the window if needed to view all the fields.

If you need to bring the Node Inspector to the front, 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.
IVR Designer displays the Prompt Editor window, shown in the following figure.

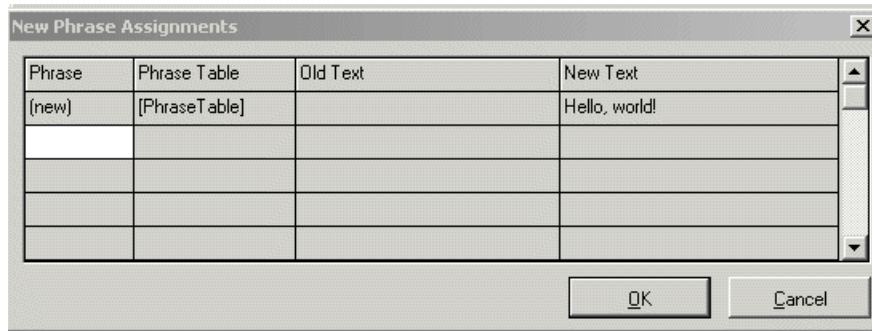


5. In the Prompt Text field, type the following text:
Hello, world!

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.

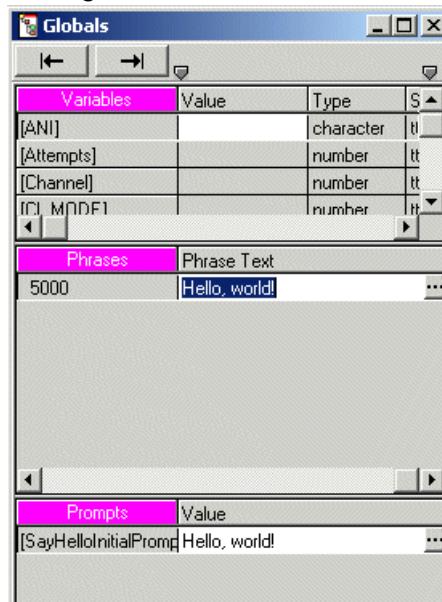
6. Click OK.
IVR Designer displays the New Phrase Assignments window, shown in the following figure.



7. Click OK to accept the defaults.

IVR Designer performs the following actions:

- 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.

To continue, see [Setting Transfer Call node options](#) on page 328.

Setting Transfer Call node options

The application is set to transfer the call to another number after the announcement. You must define how to handle the transfer by setting the Transfer Call node options.

To set the Transfer Call node options:

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.

The screenshot shows the 'Node Inspector' window for a 'Transfer Call' node. The window has a title bar with standard window controls. Below the title bar is a tab labeled 'Transfer Call'. Underneath is a dropdown menu showing 'TransferCall'. The main area is divided into three sections: 'Transfer Options', 'Answer Options', and 'Caller Options'. The 'Transfer Options' section includes fields for 'Phone Number' (set to 1234), 'Transfer Type' (set to Blind), 'Transfer Status', and 'Maximum Rings' (set to 7). The 'Answer Options' section includes 'Answer Message', 'Allow Answer Message If' (set to False), and 'Answer Connection' (set to Complete). The 'Caller Options' section includes 'Message To Caller'. At the bottom, there is a 'Node Notes' text area and two tabs: 'Transfer Options' (selected) and 'Failure Treatment'.

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.

To continue, see [Defining a Transfer Call node failure announcement](#) on page 329.

Defining a Transfer Call node failure announcement

The application is not set to answer the call, play a simple announcement, and transfer the call to another number before hanging up. 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.

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, shown in the following figure.
5. In the Prompt Text field, type the following text:
I'm sorry, your call could not be transferred. Please try again later.

If you press Return at the end of 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 follow this example through the testing and debugging phase, see [Testing a TAS application](#) on page 330.

Testing a TAS application

Before generating code and transferring the application to the target system, you should always test and debug the application using the Verify Design and Simulation tools included with IVR Designer.

The topics in Testing a TAS application provide examples of how to verify and simulate applications.

 Note:

This tour makes use of a TAS 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 [Verifying a TAS application](#) on page 331.

Related topics:

[Verifying a TAS application](#) on page 331

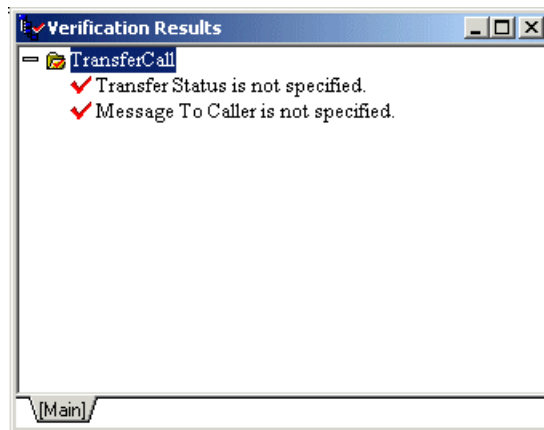
[Simulating TAS application performance](#) on page 332

Verifying a TAS application

You continue to use the application that you created (my_first). If you did not complete this application, use the sfirst TAS tour application provided with IVR Designer.

The Verify Design tool checks the application for possible errors or omissions in design.

1. Open the my_first or sfirst 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.



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 generation for the application will not complete successfully unless the problem is corrected. In this case, the Transfer Status was deliberately left unspecified, so you do not need to be concerned about it.

You do, however, 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 second 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.

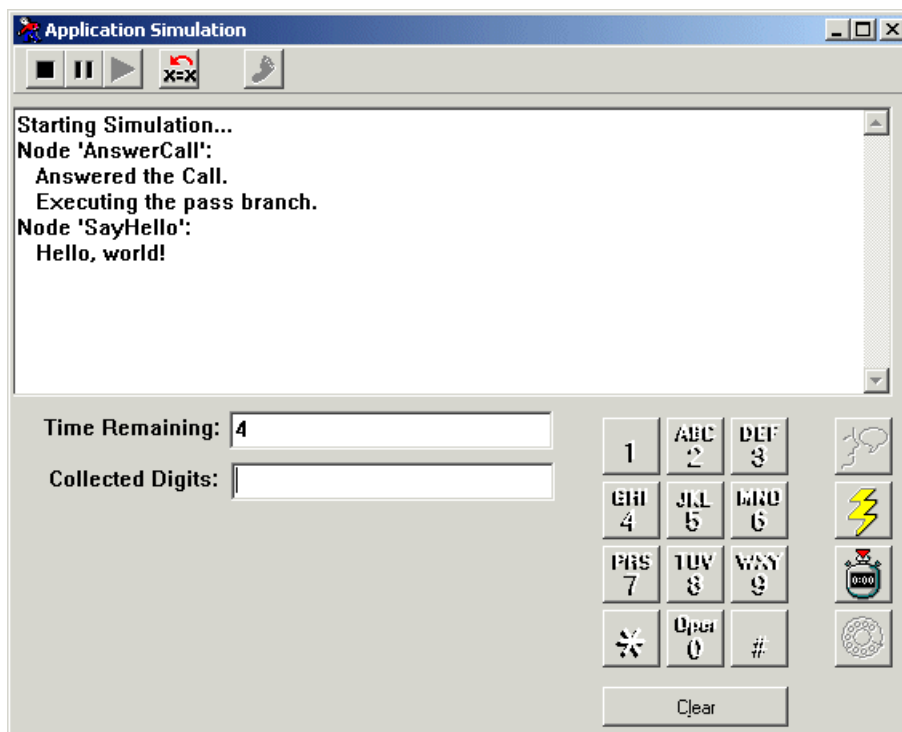
To continue, see [Simulating TAS application performance](#) on page 332.

Simulating TAS application performance

You use the Simulation tool to 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 my_first application using the Simulation tool:

1. Click the AnswerPhone 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 sets the simulation to start from the beginning of the call flow.
2.  Click the Simulation icon on the Toolbar.
The Simulation tool, shown in the following figure, starts running automatically. You can monitor the progress of the simulation in the Application Simulation window.



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 follow this example through generating the code, and transferring and installing the application, see [Generating code and transferring and installing a TAS application](#) on page 333.

Generating code and transferring and installing a TAS application

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 install the application.

The topics in Generating code and transferring and installing a TAS application show you how to complete these tasks.

 Note:

This tour makes use of a TAS 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 [Generating TAS code](#) on page 334.

Related topics:

[Generating TAS code](#) on page 334

[Transferring TAS applications](#) on page 335

[Installing TAS applications](#) on page 338

Generating TAS code

Now that you have designed and tested the my_first application, you can generate the code to be transferred to the target system.

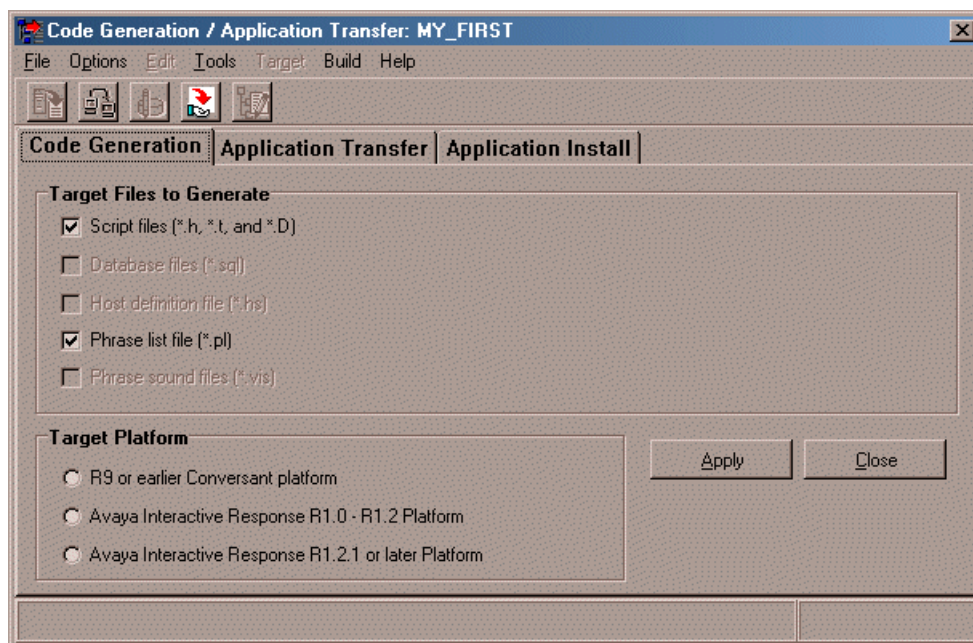
To generate code:

1. Open the my_first application or the sfirst1 TAS tour application provided with IVR Designer.

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.

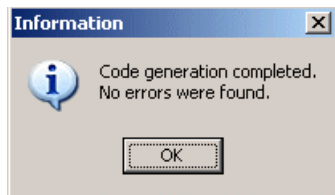
IVR Designer displays the Code Generation tab, shown in the following figure, of the Code Generation / Application Transfer window.



This window displays the kinds of files you can generate from the current application. Notice that because you have no database tables, host definitions, or phrase sound files defined for this application, these options are inactive.

3. Select the appropriate Target Platform and click Apply.
If the application will run on CONVERSANT platforms, select that option. If you will use an application on both types of platforms, you need to generate code separately for each platform.

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 below.



4. Click OK.

To continue, see [Transferring TAS applications](#) on page 335.

Transferring TAS applications

Once the TAS code is generated, transfer the application to the target system where it will run.

Caution:

If you do not have access to a separate development system, you may want to read [Transferring TAS 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:

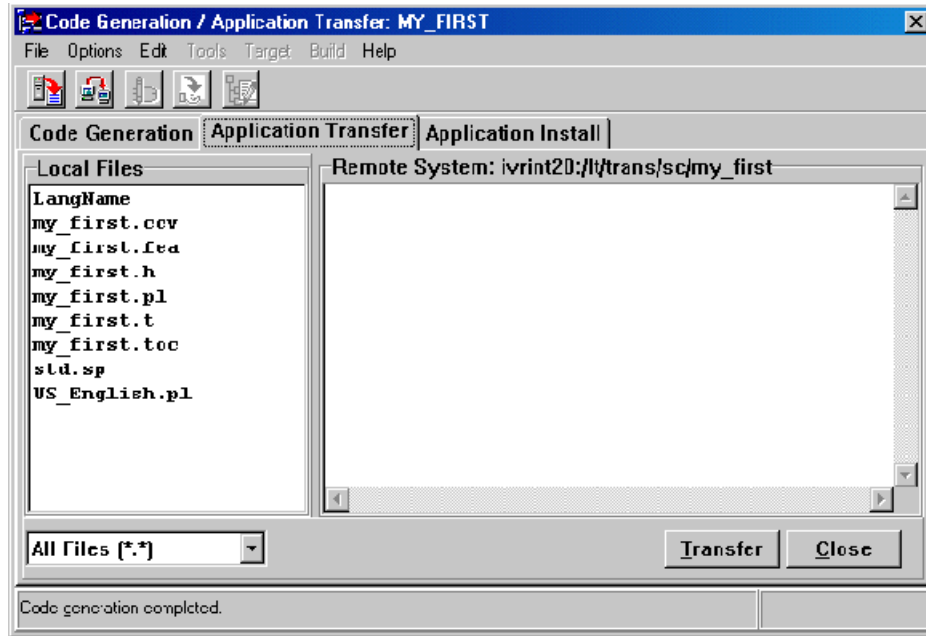
- On diskettes
- Over a network

Only network transfer is available on Avaya IR systems.

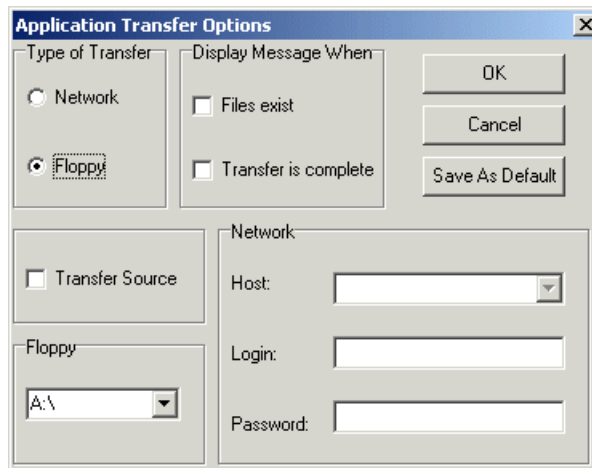
Setting the application transfer options

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, you 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. 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.

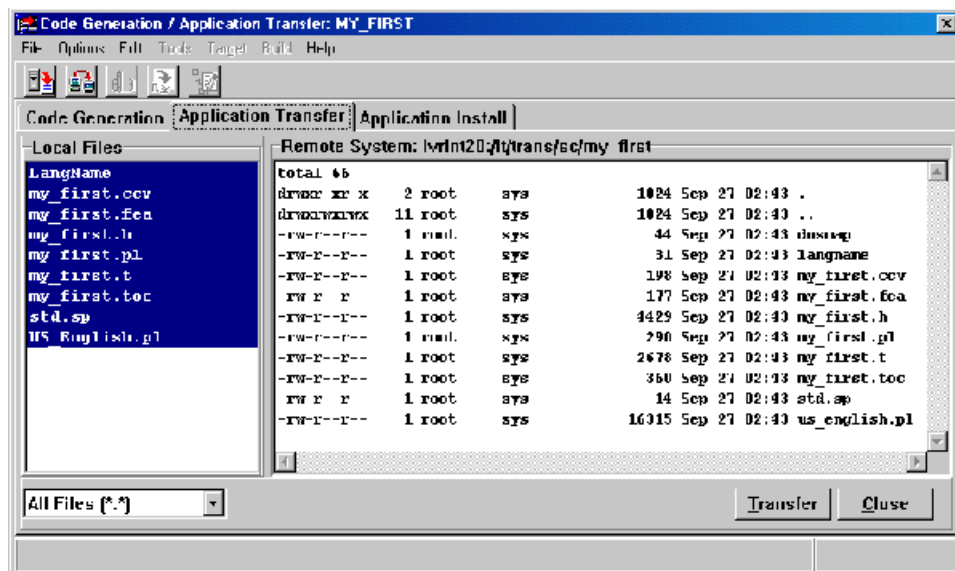
You might want to transfer the source files if you want to archive them on your Avaya IR or CONVERSANT system.

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 the 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 transfer results, as shown in the following figure.



If you are transferring to diskettes, you must transfer the files from the diskettes to the target system. For more information, see [Transferring files from floppy disks to the target system](#) on page 267.

To continue, see [Installing TAS applications](#) on page 338.

Installing TAS applications

Once the TAS code has been transferred to the target system, it must be installed and compiled there before you can run the application.

 Note:

Caution If you do not have access to a separate development system, you may want to read Installing TAS applications without installing the application on your system. Installing training applications on your production system could affect performance.

IVR Designer offers the following options for installing applications on the target system:

- From within IVR Designer, using the Application Install tab in the Code Generation / Application Transfer window

 Note:

The Application Install tab is not visible unless you have selected Network for Type of Transfer.

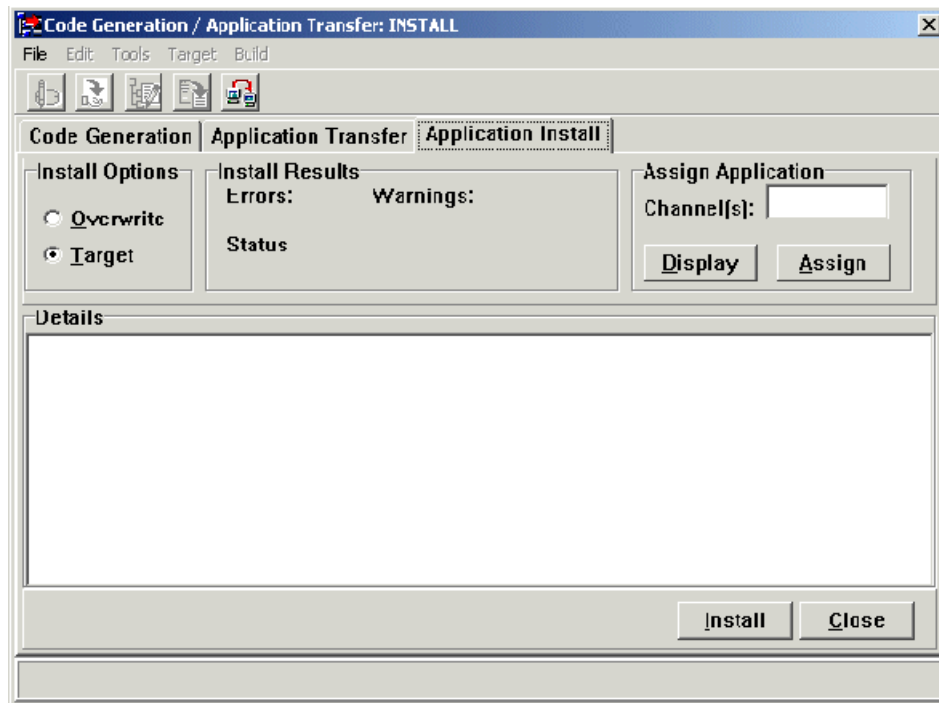
- Using the sci command at the target system or over the network

In Installing TAS applications, use the Application Install tab to install the application and assign it to one or more channels on the target system.

Installing the application

To install an application on the target system using the Application Install tab in the Code Generation / Application Transfer window:

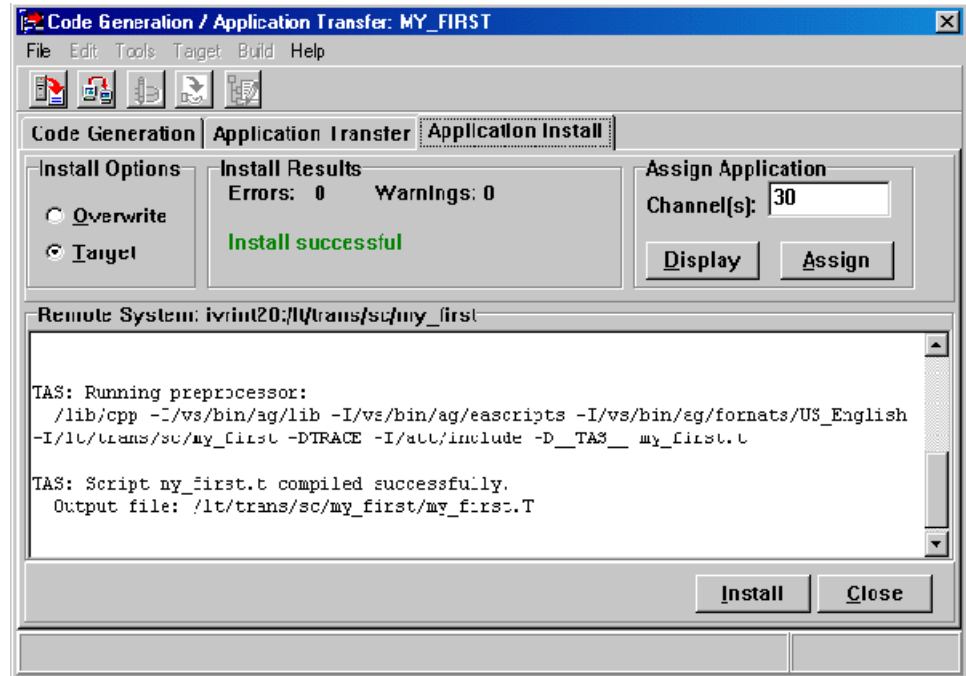
1. In the Code Generation / Application Transfer window, select the Application Install tab, shown in the following figure.



2. In the Install Options area, select the appropriate installation option.
 - To overwrite existing resources with matching names, such as database tables, host definitions, speech pools, and speech recognition grammars, select Overwrite.
 - To use existing resources, select Target.
3. Click Install.

During the installation, the Application Install tab, shown in the following figure, displays the following information:

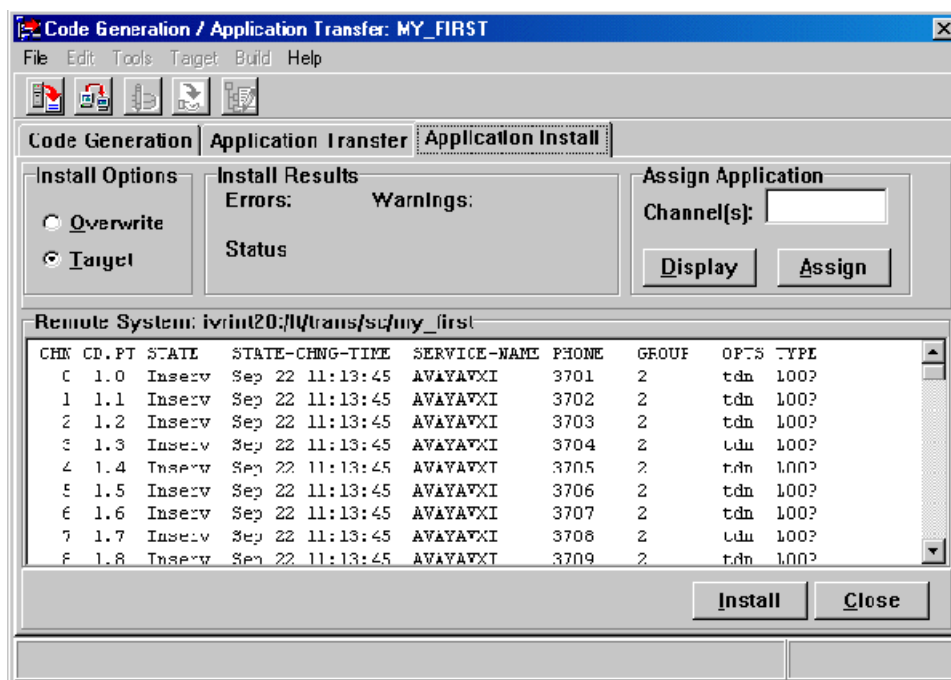
 - Current status of the procedure, as well as the number of errors and warnings encountered during installation
 - General information about the installation, including information about errors and warnings



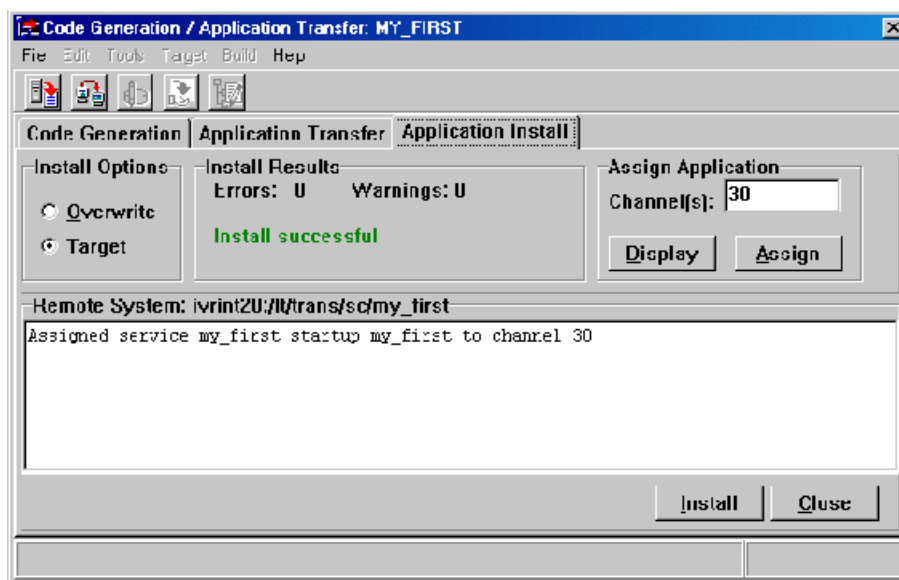
Assigning the application to channels

To assign the application using the Application Install tab:

1. Check to see if the channels to which you want to assign the application are available for assignment on the target system. To check all channels, click Display. The Remote System: field, shown in the following figure, displays information about the selected channel or channels.



2. In the Channel(s): field, enter the number or numbers of any channels to which you want to assign the application.
3. Click Assign.
The system assigns the application to the designated channels and the Remote System: field, shown in the following figure, displays a message confirming the assignment.



You can also assign the application to channels using the Web Administration tool on the Avaya IR system or `cvis_menu` on a CONVERSANT system.

To continue, see [Sharing TAS resources](#) on page 342.

Sharing TAS resources

The topics in the Sharing TAS resources tour provide a step-by-step example of how to share resources between applications.

 Note:

This tour makes use of a TAS 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 [Sharing out TAS resources](#) on page 342.

Related topics:

[Sharing out TAS resources](#) on page 342

[Sharing in TAS resources](#) on page 346

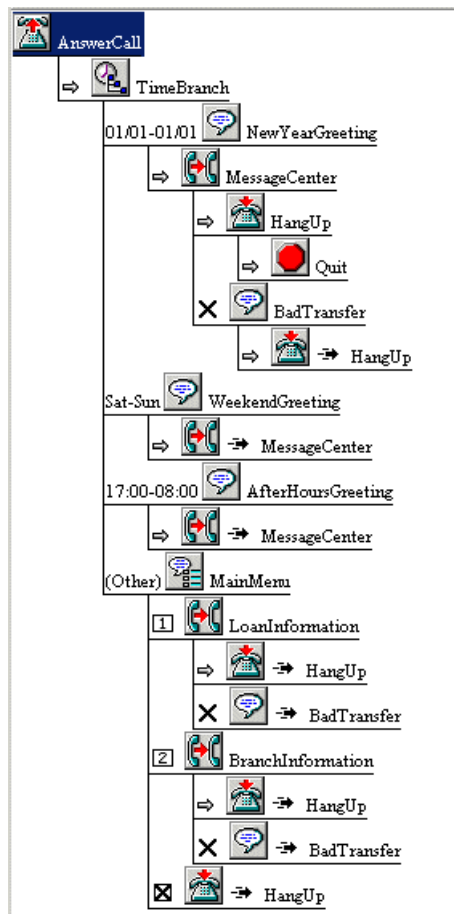
Sharing out TAS resources

This topic guides you in sharing out a resource from an application.

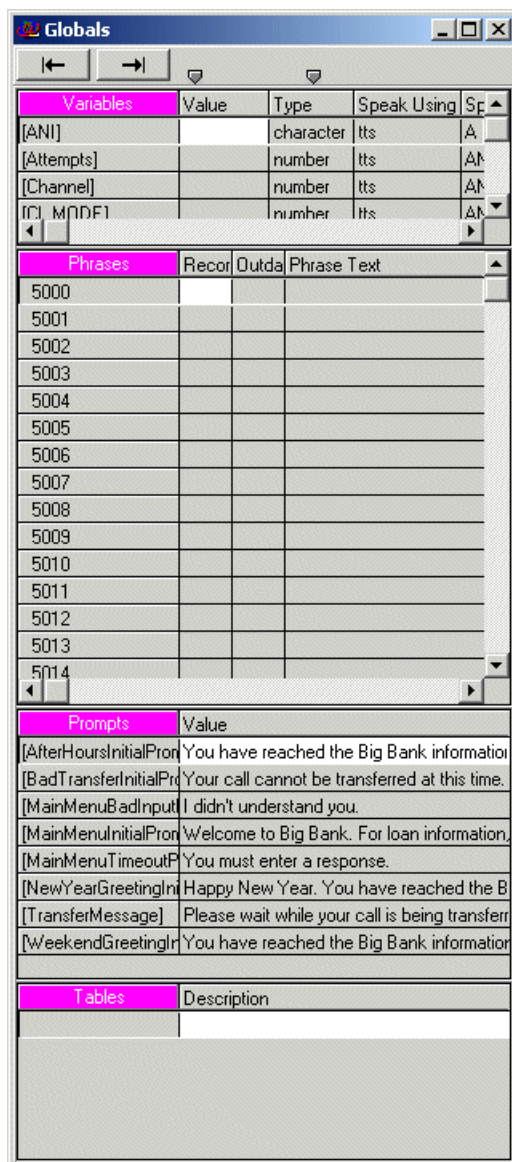
When you create a resource, such as a phrase table, you can share it with other developers. In this topic, share out the phrase table from the scallsin TAS tour application provided with IVR Designer.

To share out the phrase table:

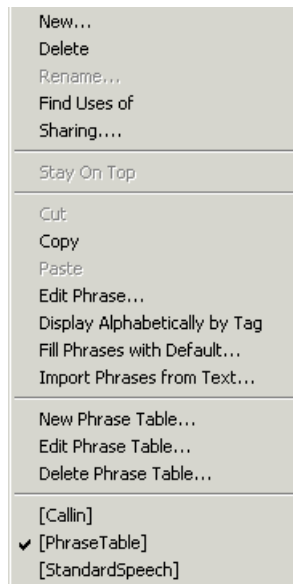
1. Open the scallsin TAS tour application.
The application is displayed in the design pad, shown in the following figure.



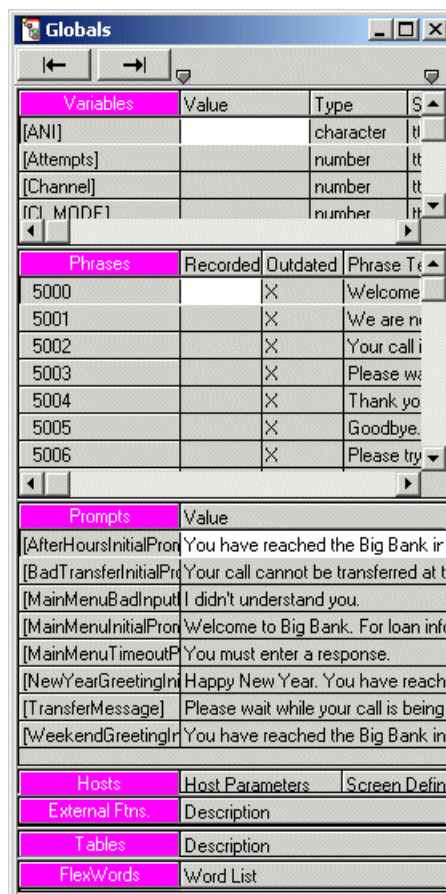
2. Open the Globals Manager by clicking the  icon. The Globals Manager, shown in the following figure, is displayed. Position the windows, as needed, for best viewing.



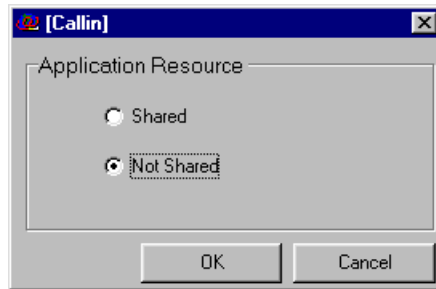
3. Right-click the Phrases resource manager.
This is the resource you want to share out. Your options for the resource, including the [Callin] phrase table, are displayed, as shown in the following figure.



4. Click the [Callin] phrase table.
The Phrases Manager, shown in the following figure, displays the phrases for this phrase table.



5. Right-click the Phrases resource manager again, and click Sharing.
The Application Resource window, shown in the following figure, is displayed.



6. Click Shared.

7. Click OK.

The [Callin] phrase table is now shared out. Other developers can get access to it across a LAN. You can also use it in other applications.

When you modify a shared resource, you can update it to make the changes available to other users. See [Updating shared-out resources](#) on page 233 for more information on updating shared resources.

To continue, see [Sharing in TAS resources](#) on page 346.

Sharing in TAS resources

Once a resource is shared out, you can use the Application Resources icon to import it or share it in to an application. Decide how to bring in a shared resource based on how you will use it, as follows:

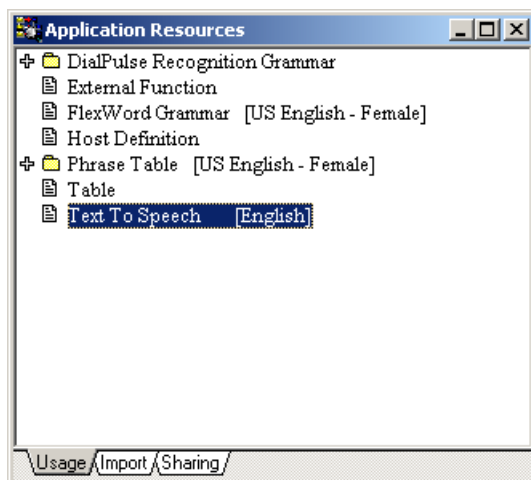
- When you import a shared resource, you bring a private copy into the application. You may modify the resource, as needed.
- When you share in a resource, you use the resource as it is. If the owner modifies the resource, you can refresh it to obtain the latest changes. With sharing in, work groups can use common resources.

Whether you share in or import a resource, the system makes a full copy of the resource on your hard drive. Bring in only those resources that you plan to use in an application. Otherwise, you waste hard drive space on files you do not need.

To complete this topic, you need the shared phrase table [Callin] from [Sharing out TAS resources](#) on page 342. If you have not completed that topic, do that now.

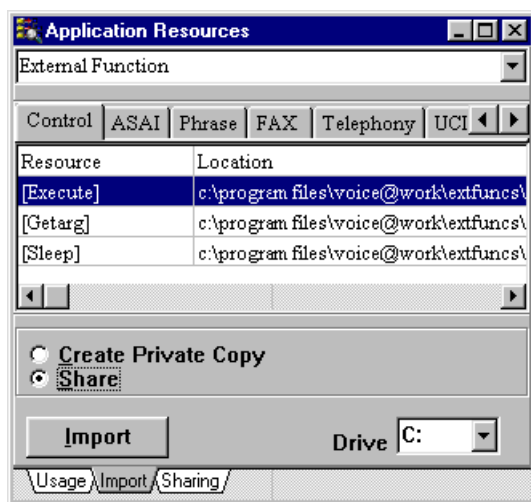
To share in or import the [Callin] phrase table resource:

1. Create a new TAS application, naming it sharein.
2. Click the Application Resources  icon on the Toolbar.
The Application Resources window, shown in the following figure, is displayed with the Usage tab in front. This tab shows the resources currently associated with the application.

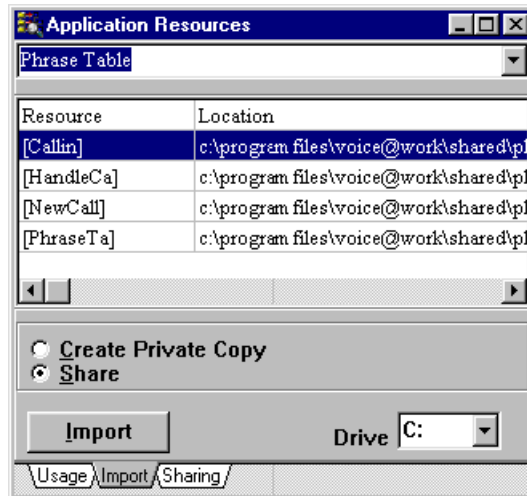


3. Click the Import tab.

The Import tab, shown in the following figure, defaults to the external functions that can be imported into applications.



4. Select Phrase Table from the drop-down menu.
The shared out phrase tables, shown in the following figure, are displayed.



5. Select the [Callin] phrase table.
6. Select Create Private Copy to import the resource.
To share in a common resource, you would leave the default Share selected.
7. Click Import to complete the process.
The resource is imported.

You imported the phrase table, creating a private copy, and you can make changes to it. If you share in resources, remember to refresh them if the owner makes changes to them. See [Refreshing shared-in resources](#) on page 238 for more information on refreshing shared in resources.

To continue, see [Branching based on time, day, or date in TAS applications](#) on page 348.

Branching based on time, day, or date in TAS applications

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 the Branching based on time, day, or date in TAS applications tour, you create an application, named incomng. You add a Time Branch node and specify branches for weekend, holiday, after hours, and business hour calls.

The incomng application:

- Answers the telephone.
- 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 TAS 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 a TAS phrase table](#) on page 349.

Related topics:

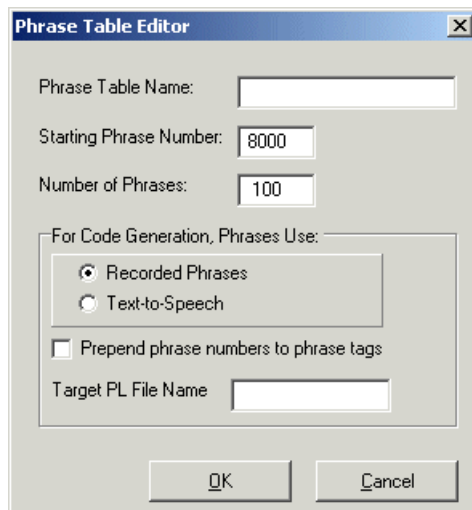
[Creating a TAS phrase table](#) on page 349

[Creating a TAS Time Branch application](#) on page 350

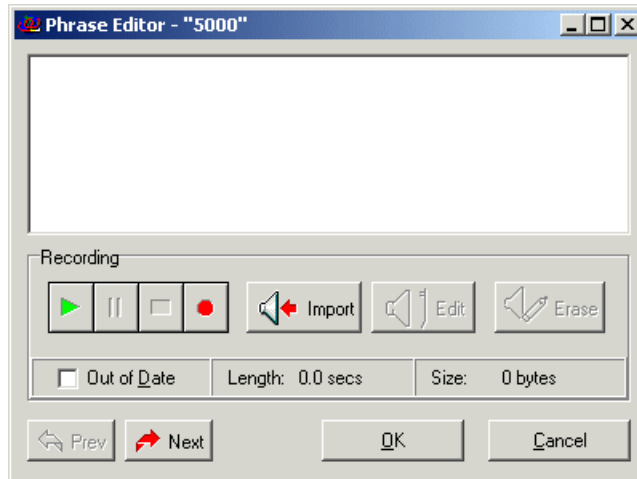
Creating a TAS phrase table

To create a phrase table to use in the application:

1. Create a new TAS application, named incomng.
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.



4. 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.
5. 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:



 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.

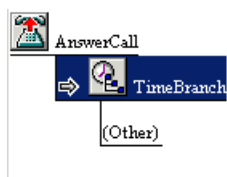
You can use this phrase table in the current application. You can also share it out for use in other applications. To learn more about sharing resources, go to [Sharing TAS resources](#) on page 342.

To continue, see [Creating a TAS Time Branch application](#) on page 350.

Creating a TAS Time Branch application

To continue to create the incoming application, using Time Branch nodes:

1. Drag an Answer Call node onto the design pad and name it AnswerCall.
2. Drag a Time Branch  node onto the open branch and name it TimeBranch, as shown in the following figure.



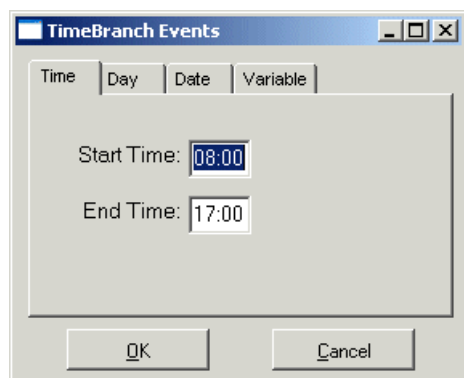
When you add a Time Branch node, IVR Designer creates one branch named (Other). You insert additional branches to meet the requirements of your application. For each branch, you specify the conditions under which the branch is taken. The order of branches is important. The system will take the first branch with true conditions.

3. Insert a branch for the New Year holiday:

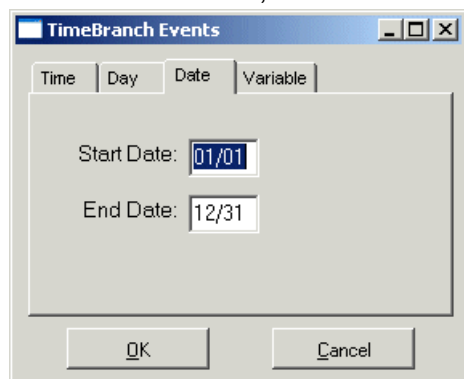
a. Right-click the (Other) branch.

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.



c. Select the Date tab, shown in the following figure.

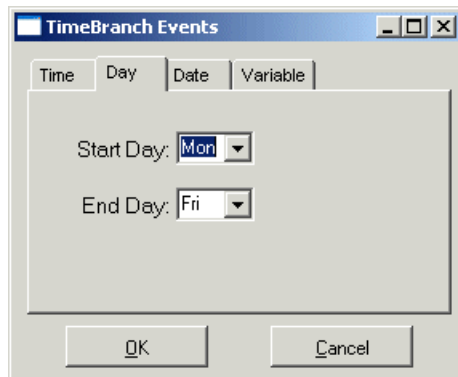


The start and end dates require two digits each for month and day. The system automatically enters the slash.

d. Enter 01/01 in both the Start Date and End Date fields and click OK.

4. 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 before you can specify the branch.

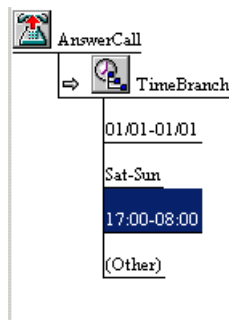


Use the drop-down menus to select the start and end days for the weekend.

5. Use the Time tab to insert a branch for the hours the business is closed (5 p.m. to 8 am).

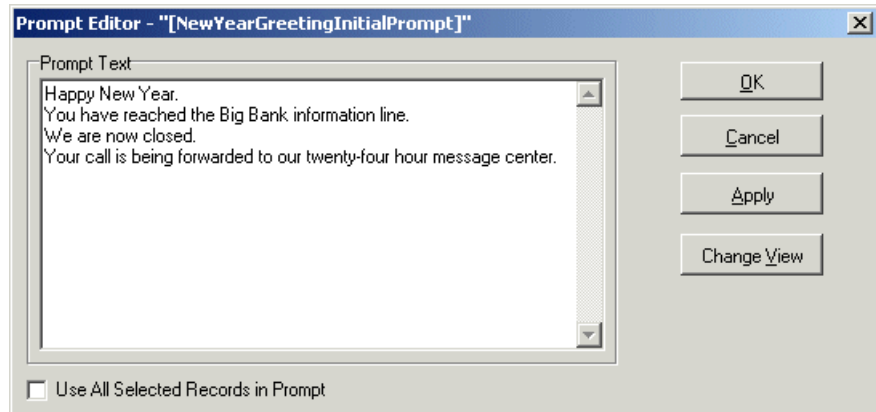
Remember to use a twenty-four hour time format.

This completes the required branches, as shown in the following figure.



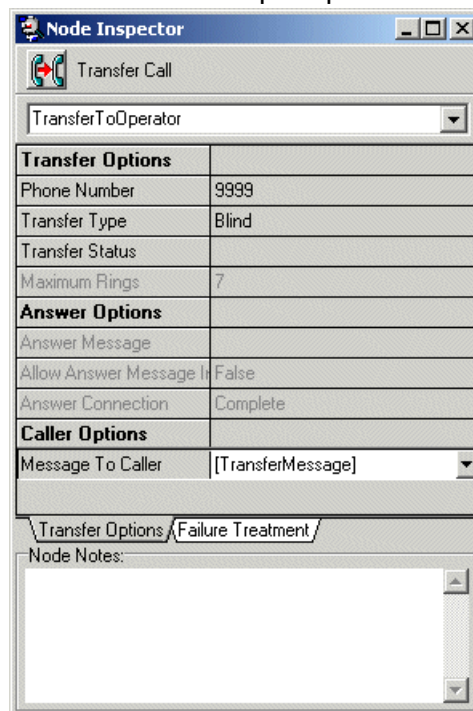
You can insert additional holidays if you want. In the rest of this tour, you will define what happens for each branch.

6. 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 for the New Year greeting (5007, 5008, 5001, and 5002), as shown in the following figure, and click OK.



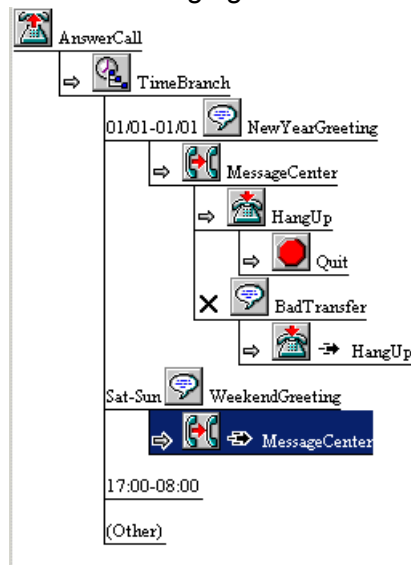
After the announcement plays, the application should transfer the call to the message center. You must add a node to handle the transfer.

7. 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.



8. 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.
9. 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 simpler.
10. 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 to callers in TAS applications](#) on page 359.

To continue, see [Copying TAS nodes and applications](#) on page 355.

Copying TAS nodes and applications

The topics in Copying TAS nodes and applications provide examples of how to copy

- Nodes within and between applications
- Entire applications

**Note:**

This tour makes use of a TAS 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 TAS nodes](#) on page 355.

Related topics:

[Copying TAS nodes](#) on page 355

[Copying TAS applications](#) on page 358

Copying TAS 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 TAS applications](#) on page 355.

Related topics:

[Copying nodes within TAS applications](#) on page 355

[Copying nodes between TAS applications](#) on page 357

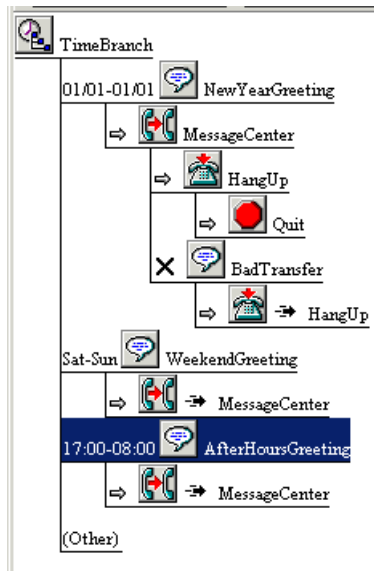
Copying nodes within TAS applications

In this topic, you copy a node within an application. If you developed the incoming application in [Branching based on time, day, or date in TAS applications](#) on page 348, you complete that application. If you did not complete that tour, use the incoming TAS tour application provided with IVR Designer.

1. Open the incoming application or the incoming TAS tech 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 will handle any Jump to Nodes. Click OK to clear this message.

Notice that the node under WeekendGreeting is also highlighted.

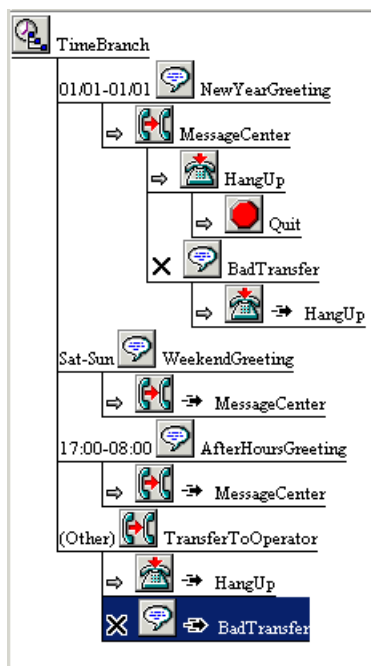
4. Click the 17:00-08:00 branch.
5. Click the Paste  icon on the Toolbar.
6. Name the new node AfterHoursGreeting and click OK.
The node is pasted onto the new branch, as shown in the following figure.



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.



This completes the application.

To continue, see [Copying nodes between TAS applications](#) on page 357.

Copying nodes between TAS 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 is because you need somewhere to place the subordinate nodes.

In Copying nodes between applications, you create a new TAS application and copy nodes from an existing application.

1. Create a new TAS application and name it cpynode.
2. Drag in an Answer Call node, name it AnswerCall.
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. Notice 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 cpynode application.
You do not need to save the existing application because you did not change it.
7. Click the open branch after the AnswerCall node and click the Paste icon.
IVR Designer warns you that you cannot undo the paste.
8. Click Yes.
The copied nodes are pasted into the cpynode 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 TAS applications](#) on page 358.

Copying TAS 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:

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 to callers in TAS applications](#) on page 359.

Presenting menus to callers in TAS applications

Use Menu nodes to create code that presents a list of choices to callers. The topics in Presenting menus to callers in TAS applications show you how to create and use Menu nodes.

You complete an application that gives callers access to loan information or branch locations. The menu application is based on the incoming application that was created in [Branching based on time, day, or date in TAS applications](#) on page 348. The menu application:

- Answers the phone.
- 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 TAS 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 TAS menus](#) on page 359.

Related topics:

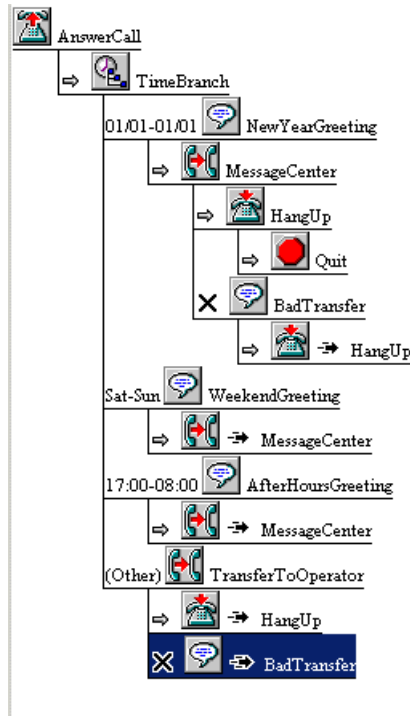
[Using TAS menus](#) on page 359

[TAS Menu error prompts and response handling](#) on page 364

Using TAS menus

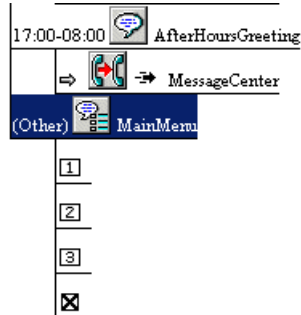
In this tour, you create a menu that callers use to choose the type of information they want to hear. You start with an existing application and add a menu. If you completed the incoming application in [Copying nodes within TAS applications](#) on page 355, you can use that application. If not, use the smenu TAS tour application provided with IVR Designer.

1. Open the incoming application or the smenu TAS tour application, shown in the following figure, and save it as menu.
For information on saving an application under a different name, see [Copying TAS applications](#) on page 358.



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.

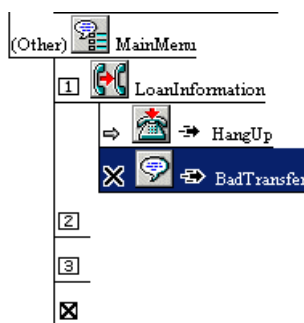
2. Right-click the TransferToOperator node and select Delete Node, confirming that you want to delete the node.
The (Other) branch is now empty.
3. Drag a Menu  node onto the empty branch and name it MainMenu, as shown in the following figure.



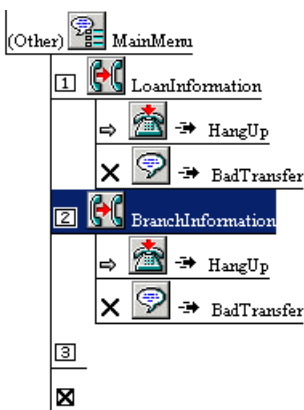
By default, Menu nodes include three numbered branches and a fail branch. You can insert additional numbered branches or remove branches as needed.

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 phone number for the transfer and select [TransferMessage] as the Message to Caller.
Use a phone 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.

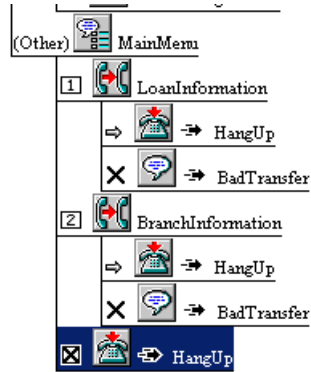


8. Copy the LoanInformation node and paste it on branch [2]. Rename it BranchInformation and change the phone number, as shown in the following figure.
For more information on copying nodes, refer to [Copying nodes within TAS applications](#) on page 355.



This completes the branches to the loan information and branch information lines. The third branch is not needed.

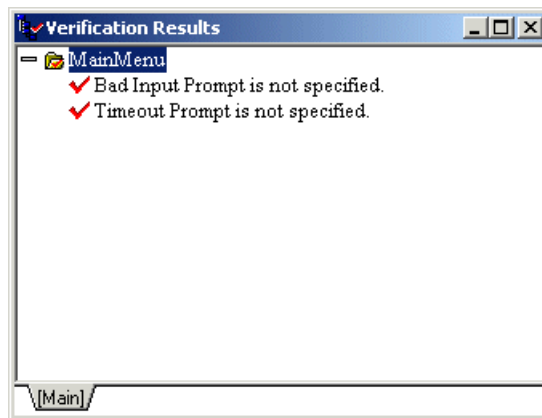
9. Right-click the third branch and select Delete Branch.
The fail branch of the menu node is the only remaining empty node. The application just disconnects if the caller fails to respond to the menu successfully.
10. Jump to the HangUp node from this branch, as shown in the following figure.



This completes the application.

11. Verify the application.

The verification results window, shown in the following figure, indicates a number of items that have not been specified. The application will run without these items. Just close the window.

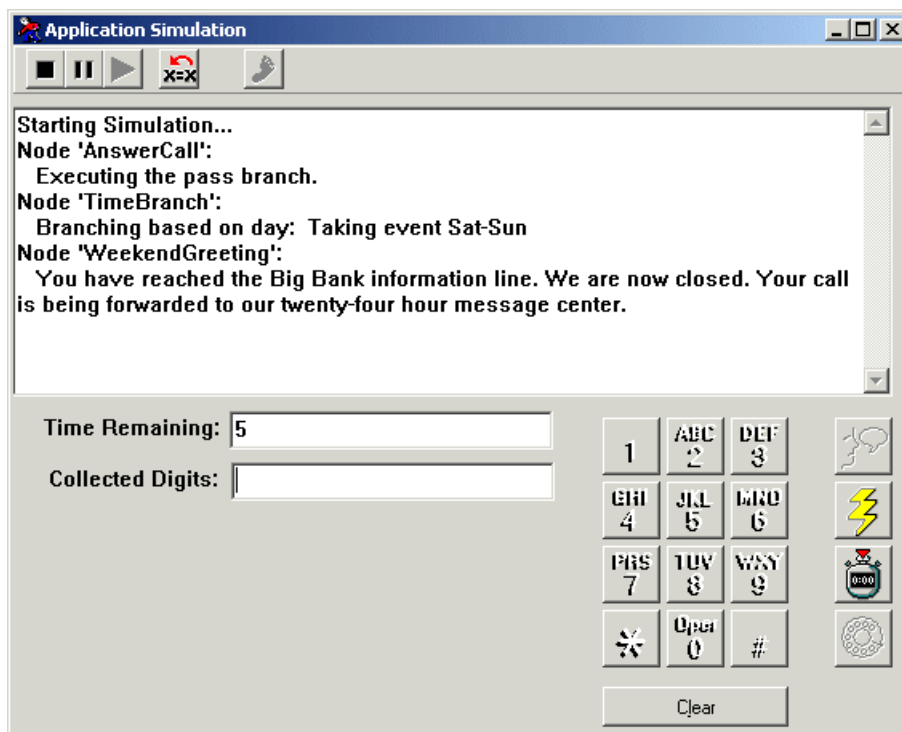


For more information on verifying applications, see [Testing a TAS application](#) on page 330.

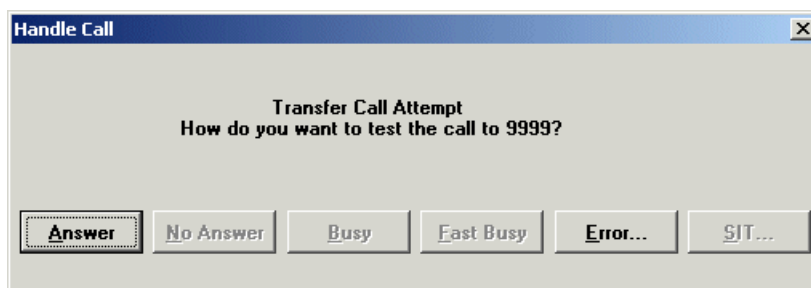
12. Simulate the application, starting from the AnswerCall node.

For more information on simulating applications, see [Simulating TAS application performance](#) on page 332.

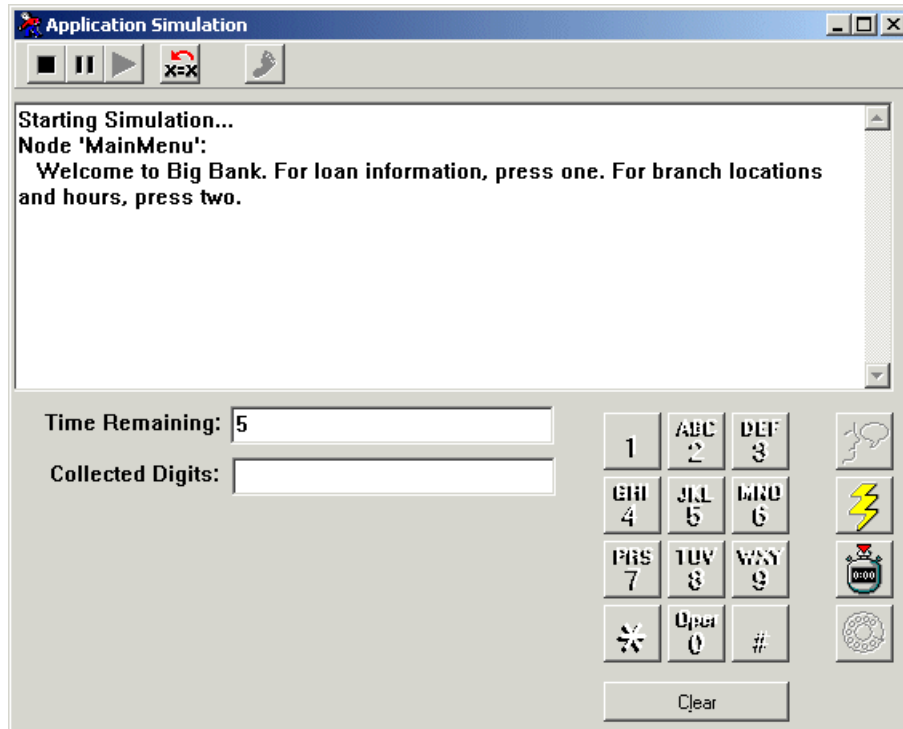
The simulation starts and takes the branch for the current day and time, as shown in the following figure. It counts down and then continues.



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, as shown in the following figure.



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, install, and test the application. Make sure that you test all branches.

To continue, see TAS Menu error prompts and response handling.

TAS 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 editable attributes to 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 to provide input within a specified time.
- A Too Few Digits Prompt plays if the caller does not enter the minimum number of digits you specify.

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 Discard Previous Touchtones, you can specify whether the system should ignore touchtones entered before the prompt begins to play. In most cases, you want to save touchtones. When you save touchtones callers can anticipate the menu choices and type ahead. Regular callers really appreciate this capability.
- Input DPR Parameters identify how the target system can recognize digits dialed by callers using rotary dialed telephones.



Note:

Dial pulse recognition is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

- Input ASR Parameters specify what the system should expect in the way of automatic speech recognition.
- 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 where to save the caller's 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 [Collected digits].

To continue, see [Specifying TAS menu error prompts](#) on page 365.

Related topics:

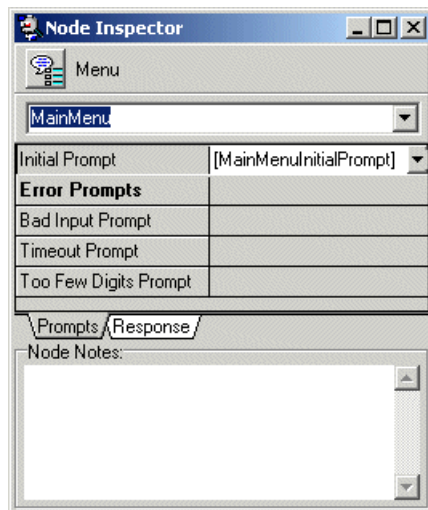
[Specifying TAS menu error prompts](#) on page 365

[Specifying TAS menu response handling](#) on page 367

Specifying TAS menu error prompts

Now you specify the error prompts and response handling for the menu you created in the menu application.

1. Open the Node Inspector, shown in the following figure, for the MainMenu node.



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 didn't 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 TAS menu response handling.

Specifying TAS menu response handling

You have many options when setting response handling parameters, as shown in the following figure.

Input Parameters	
Allow Interrupt	True
Discard Previous Touchtones	True
Input DPR Parameters	
Input DPR Mode	(none)
Input ASR Parameters	
Input ASR Mode	Touchtone
Mode Chosen	
Reserve ASR/DPR Resources	False
Free Reserved ASR/DPR Resources	False
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	[CollectedDigits]
Data Collection	
Track Caller's Response	False

Prompts / Response /

Node Notes:

- You allow the caller to interrupt the prompt, so leave Allow Interrupt set to True.
- You want to discard previous touchtones, so leave Discard Previous Touchtones at True.
- The application does not use dial pulse recognition, so leave the Input DPR Mode field at (none).



Note:

Dial pulse recognition is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

- You use touchtones instead of speech recognition for this application, so Input ASR Mode should remain set to Touchtone. The remaining ASR fields apply to speech recognition, so you leave them alone.

- The default First Digit Timeout, the time the system waits before time out is five seconds. That works for this application. The three-second Inter-Digit Timeout, the allowed time between digits is also OK.
- Tries Available, the number of tries a caller is given is stored in the variable [Tries]. Use the default value.
- Tries Taken, the number of attempts the caller makes is stored in the variable [Attempts]. You do not track attempts in this application.
- After each try, the application replays the initial menu prompt, so Replay Initial Prompt can be left as True.
- The standard is to save the caller response in the [Collected Digits] variable. Use that default for Save Response In.

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, testing different menu options.
3. If you have a development system, generate code and transfer and install the application.

To continue, see [Collecting information in TAS applications](#) on page 368.

Collecting information in TAS 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 telbank application:

- Answers the telephone.
- Welcomes the caller to the telephone banking line.
- Prompts the caller to select from menu options.
- 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, looks up the balance in a table, announces the balance back to the caller, and returns to the menu.
- 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 use existing phrases or create your own phrases by typing them into the Prompt Editor.

*** Note:**

This tour makes use of a TAS 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 Prompt and Collect in TAS applications](#) on page 369.

Related topics:

[Using Prompt and Collect in TAS applications](#) on page 369

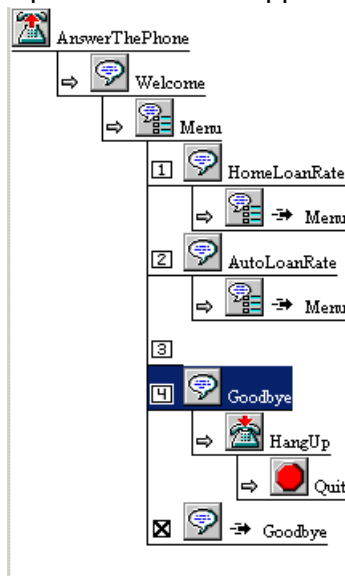
[Prompts, Response, Confirmation, and Validation](#) on page 370

[Locating information in tables](#) on page 375

Using Prompt and Collect in TAS applications

In this topic, you open the partially developed stelbank TAS tech tour application provided with IVR Designer and add a Prompt and Collect node.

1. Open the stelbank application, shown in the following figure, and rename it telbank.



The application includes nodes for providing information on loan rates and for ending the call. You add a Prompt and Collect node to the third branch. This node collects 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.

You will complete the parameters of the Prompt and Collect node in Prompts, Response, Confirmation, and Validation in TAS.

To continue, see Prompts, Response, Confirmation, and Validation in TAS.

Prompts, Response, Confirmation, and Validation

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 and validation 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 to identify how the Avaya IR or CONVERSANT system should deal with caller responses.

- Input Parameters identify how callers interact with the Prompt and Collect node.

- Identify whether caller input can interrupt announcements with Allow Interrupt.
 - Specify whether the system should ignore touchtones entered before the prompt begins to play with Discard Previous Touchtones. In most cases, you want to save touchtones. When you save touchtones callers can anticipate the menu choices and type ahead. Regular callers appreciate this capability.
 - 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.
- 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 where to save the caller's 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].
 - Input DPR Parameters identify how the target system can recognize digits dialed by callers using rotary dialed telephones.

**Note:**

Dial pulse recognition is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

- Input ASR Parameters specify what the system should expect in the way of automatic speech recognition.

On the Confirmation tab, you specify the following parameters to identify how the caller response is to be confirmed.

- Confirmation DPR Parameters identify how the target system confirms digits dialed by callers using rotary dialed telephones.

**Note:**

Dial pulse recognition is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

- Confirmation ASR Parameters specify what the system should confirm for automatic speech recognition. [Yes Digit] is a variable with a default of 1. If this is specified, the system considers the confirmation digit 1 as valid. Any other digit is considered not to confirm.

On the Validation tab, you specify the following parameters to identify how the system validates the caller response.

- Validate Response Against specifies how to validate against a variable, a set, or a table.
- Variable Validation specifies the variable and the relation.
- Set Validation specifies the name of the set.
- Table Validation specifies the table and field for validation.

To continue, see [Specifying TAS Prompt and Collect parameters](#) on page 372.

Related topics:

[Specifying TAS Prompt and Collect parameters](#) on page 372

Specifying TAS Prompt and Collect parameters

To specify the error prompts for the Prompt and Collect node in the telbank application:

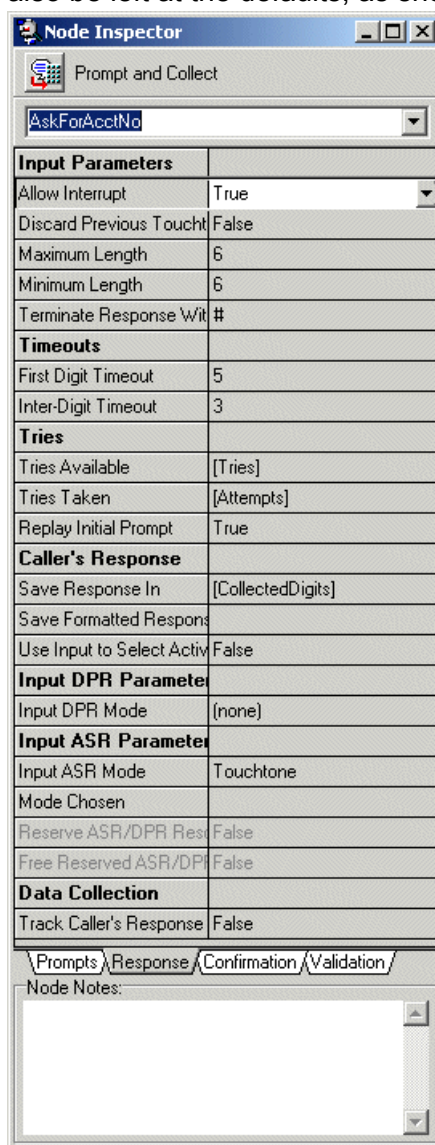
1. Click the AskForAcctNo node, shown in the following figure, and open the Node Inspector.



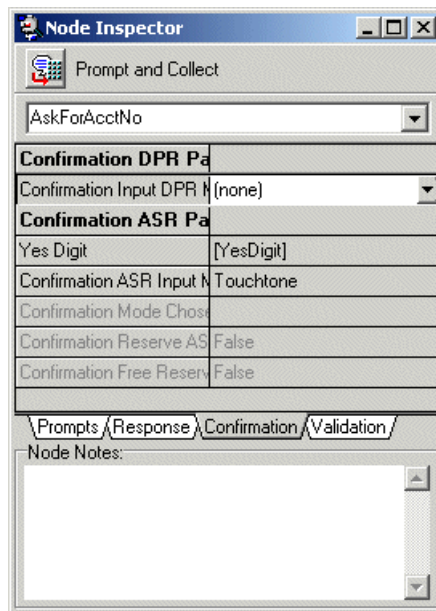
2. Create a Bad Input Prompt with the following message:
Sorry, the account number you entered is not valid.
3. Use the Timeout Prompt from the Menu node.
Select MenuTimeoutPrompt from the drop-down menu.
4. Create a Too Few Digits prompt with the following message:
You entered too few digits.
5. Create 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 [Working with variables](#) on page 99.
 - d. Complete the prompt with the following:
. If this is correct, press one. If this is incorrect, press any other key.
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.

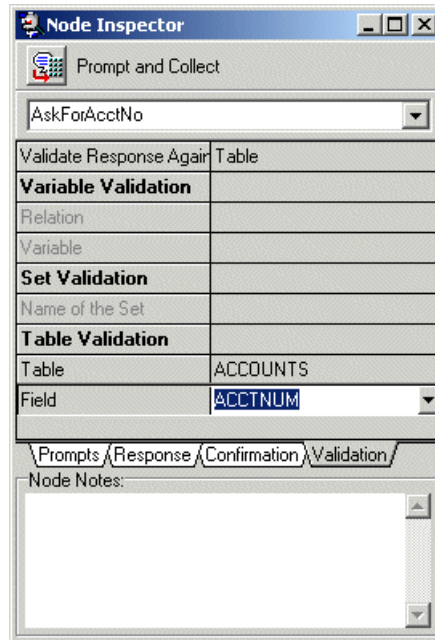
7. For the input parameters:
 - a. Keep the Allow Interrupt and Discard Previous Touchtones parameters at their defaults.
 - b. Set both the Maximum Length and Minimum Length to 6. (Account numbers are six-digit numbers.)
 - c. Change Terminate Response With to #. (Callers press the pound sign when they 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.



9. Take a moment to review the default values.
The default values for this application are acceptable. No action is required on this tab.
10. Click the Validation tab of the Node Inspector.
The application validates the response against account numbers in the ACCOUNTS table. This table was imported into the application. You can learn more about tables in [Creating database applications](#) on page 408.
11. Complete the Validation parameters, as shown in the following figure:
 - a. Select Table from the drop-down menu for the Validate Response Against field.
 - b. Select ACCOUNTS as the Table.
 - c. Select ACCTNUM as the Field.



These parameters cause the application to check the caller input against the ACCTNUM field of the ACCOUNTS table to verify that it is a valid account number. If the account number entered by the caller is not found in the table, the Prompt and Collect node goes to the fail branch.

This completes the Prompt and Collect node. To use the account number entered by the caller to locate the account balance, you must query the ACCOUNTS table.

To continue, see [Locating information in tables](#) on page 375.

Locating information in tables

In this topic, you add a Query Table node to the telbank application to retrieve information stored in tables.

A table has one or more:

- Fields, each of which holds a different type of information
- Records, each of which contains data in each table field.

The table you use in the telbank application includes:

- Fields for account number and balance
- Records containing information for different account holders

View the table

To view the ACCOUNTS table:

1. Right-click ACCOUNTS in the Tables Manager in the Globals Manager.
2. Select Edit Table Data to view the data in the ACCOUNTS table.

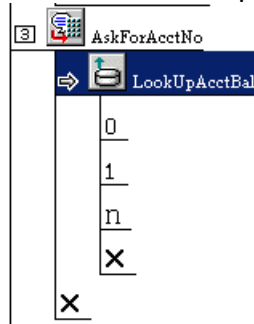
The Table Editor shows the two fields and four records. This table is much simpler than those used in real applications. However, it includes enough detail for training purposes.

3. Click OK to close the Table Editor.

Locate the account balance

To have the application query the table to locate the account balance and announce it to the caller:

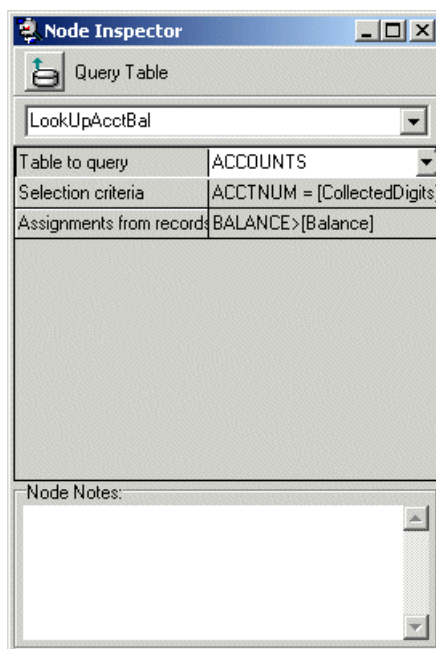
1. Drag a Query Table  node to the open branch under the AskForAcctNo node and name it LookUpAcctBal, as shown in the following figure.



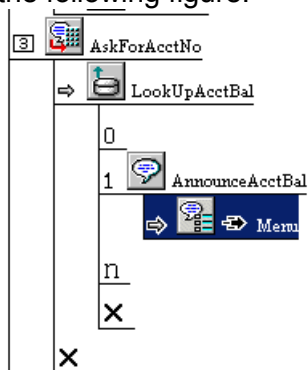
The Query Table node brings in a number of branches:

- 0 if the selection criteria identify no matches.
- 1 if exactly one match is found.
- n if more than one match is found.
- X if the database is not accessible.
To simplify the application, assume that the database is available and that the application always finds only one branch.

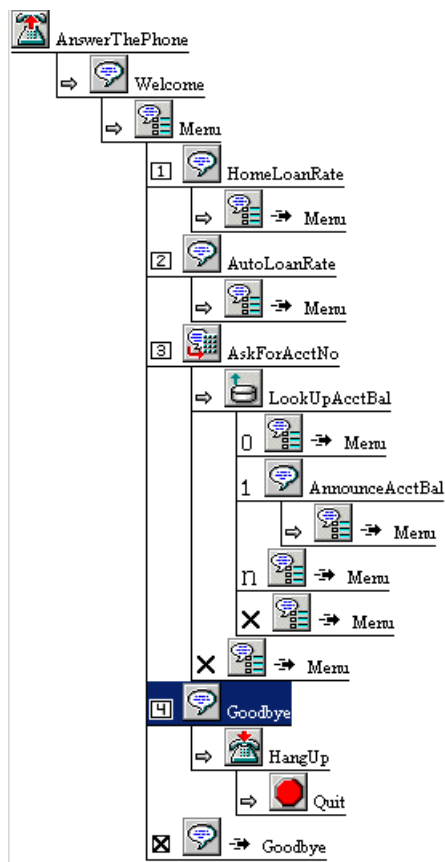
2. In the Node Inspector, select ACCOUNTS as the Table to query.
The application puts the contents of the Balance field for the selected record into a variable named [Balance].
3. Click Selection Criteria and then click
4. Complete the Query Table Editor:
 - a. Select = from the drop-down menu for Relation.
 - b. Select [CollectedDigits] as the Variable/Literal for the ACCTNUM field.
This variable setting causes the application to select the record whose ACCTNUM field is equal to the value in the [CollectedDigits] variable. That is where the caller input was stored.
 - c. Select [Balance] as the Assign value to variable.
The application assigns the value from the BALANCE field to this variable, which was created earlier as part of the application. For information on creating variables, see [Using variables and speech in TAS applications](#) on page 385.
 - d. Click OK to accept the selection criteria.
The Node Inspector, shown in the following figure, shows how the LookUpAcctBal node works.



5. Drag an Announcement node to the [1] branch and name it AnnounceAcctBal.
6. Create the following initial prompt for the AnnounceAcctBal node:
Your current balance is [Balance] .
7. Jump back to the Menu node after announcing the account balance, as shown in the following figure.



8. Complete the application by jumping back to the Menu node from all of the open branches, as shown in the following figure.



9. Save, verify, and simulate the application.
See [Testing a TAS application](#) on page 330 for more information.
10. If you have a development system, generate code, transfer, install, and test the application.
See [Generating code and transferring and installing a TAS application](#) on page 333 for more information.

To continue, see [Using calculations in TAS applications](#) on page 379.

Using calculations in TAS applications

Sometimes information that you want to give callers is not available directly from a variable or table. 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 TAS applications is setting the value of a variable based upon a calculation.

The application you develop:

- Welcomes callers and prompts them to enter their account numbers.
- Validates account numbers.
- Takes action based on the validity of the account numbers entered by callers:
 - For valid account number, prompts callers to enter a deposit amount, calculates annual interest on the amount entered, and announces that interest back to the caller.
 - For invalid account numbers, performs an intelligent transfer to an operator.

With blind transfers, applications do not check to see if the transfer completes successfully. With intelligent transfers, applications track the success of the transfer and play a message if the transfer fails. The Node Inspector for an intelligent transfer lets you specify the maximum number of rings before considering the call not answered. You can also specify failure treatment options for busy, no answer, or error conditions.

Note:

This tour makes use of a TAS 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 TAS](#) on page 379.

Related topics:

[Performing calculations in TAS](#) on page 379

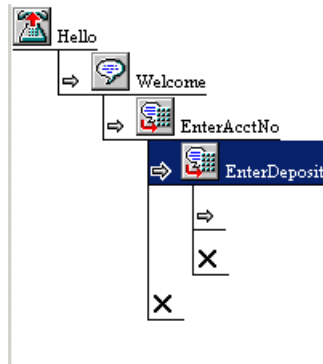
[Using intelligent transfer](#) on page 383

Performing calculations in TAS

In this topic, you add a Set and Test node to a partially completed application. The added node computes annual interest. You will complete this application in [Using intelligent transfer](#) on page 383.

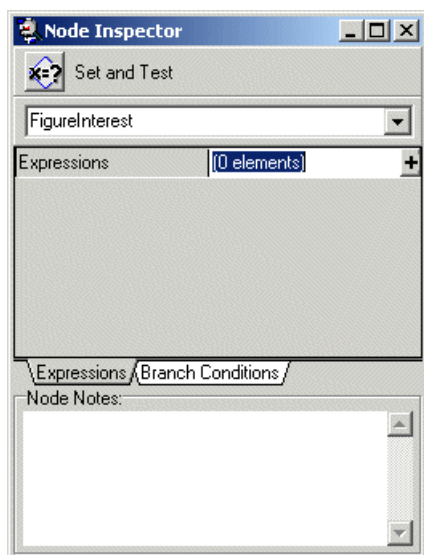
To develop the earning application:

1. Open the searn TAS tour application provided with IVR Designer, shown in the following figure, and rename it earning.

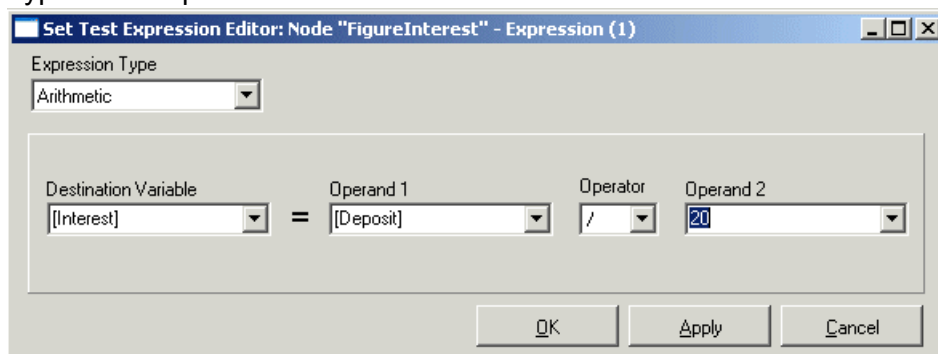


2. Review the application as it currently exists:
 - a. Review the Variables Manager and notice the custom variables [Deposit] and [Interest].
 Caller input is always considered to be characters, so caller input cannot be used directly in calculations. The earning application saves the caller input into both the [CollectedDigits] variable and a custom variable called [Deposit]. [Deposit] is a number variable 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 ACCOUNTS table in the Table Manager.
 - d. Review the Node Inspector for the Welcome node.
 Allow Interrupt is set to True and Discard Previous Touchtones is set to False. Callers can interrupt prompts.
 - e. Review the Node Inspector for the EnterAcctNo node.
 This node asks callers to input their account numbers. It includes prompts for bad input, timeout, and too few digits. You do not confirm input for this node, so the Confirm Input prompt is left blank. For Responses, the maximum and minimum lengths are set to 5. The response terminates with #. The remaining Response parameters are left at their defaults. The application does not confirm caller input, so that tab is left at its defaults. The Validation tab indicates that the application validates caller input against the ACCTNUM field of the ACCOUNTS table.
 - f. Review the Node Inspector for the EnterDeposit node.
 This node asks callers to input a deposit amount. It uses the same Timeout prompt as the EnterAcctNo node. For Responses, the Maximum Length is set to 10, with # to terminate responses. The caller response is saved in [CollectedDigits]. A formatted response is also saved in [Deposit] so it can be used in a calculation. The input is not confirmed or validated.
3. Drag a Set and Test node  to the pass branch under the EnterDeposit node and name it FigureInterest.

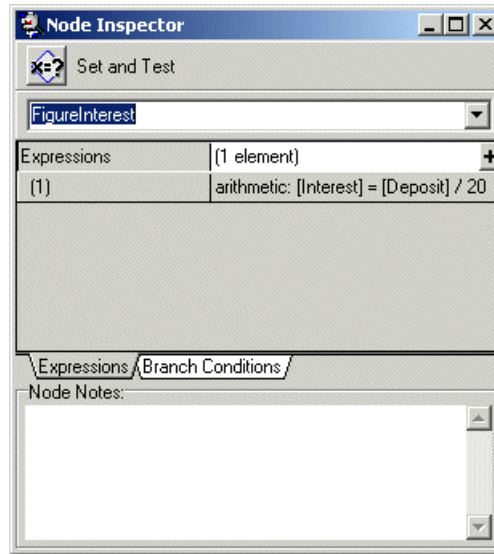
On the node inspector for the FigureInterest node, shown in the following figure, you can specify expressions or branch conditions. In this case, you specify one expression.



4. Click + at the right side of the expression field to add an element.
5. Double-click the open field or click ... to open the Expression Editor.
6. 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.

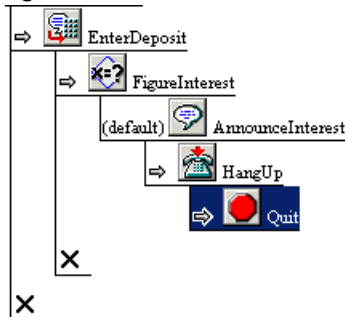


- f. Click OK to accept the value and close the window.
The Node Inspector, shown in the following figure, now shows the expression you created.



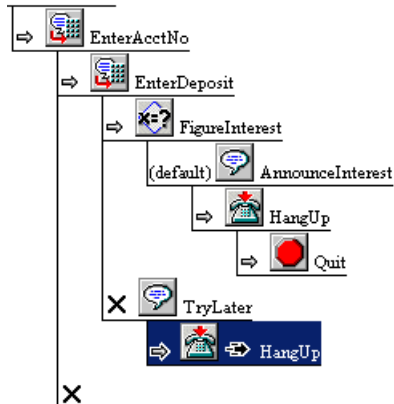
This completes the Set and Test node.

7. 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.
8. 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 and Discard Previous Touchtones to True.
9. 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.



10. Drag an Announcement node to the fail branch of the EnterDeposit node and name it TryLater.
11. Complete the TryLater node:
 - a. Add an initial prompt that says:
There was a problem with your input. Try again later.
 - b. Set Allow Interrupt to False and Discard Previous Touchtones to True.

12. Jump to the HangUp node from the open branch under the TryLater node, as shown in the following figure.



13. Save the application.
What remains is to transfer the caller to an operator if the caller does not enter a valid account number. You do this in [Using intelligent transfer](#) on page 383.

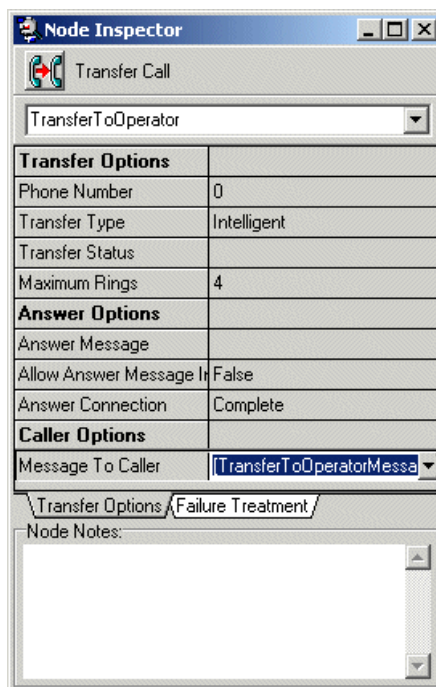
To continue, see [Using intelligent transfer](#) on page 383.

Using intelligent transfer

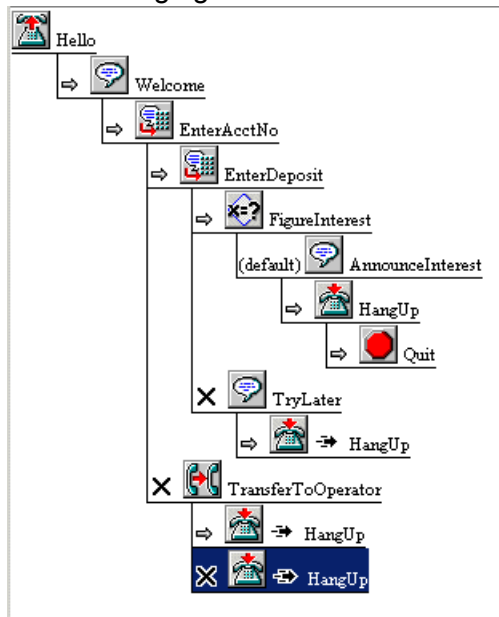
In this topic, you complete the earning application. Using intelligent transfer, you add code to transfer callers to an operator if they enter an invalid account number. The fail branch of the EnterAcctNo node performs the transfer.

To plan for this transfer, you must decide how many times the telephone should ring while the caller waits for the operator to answer. If the operator does not pick up within the specified number of rings, the call is treated as a No answer. Be aware that the Avaya IR and CONVERSANT systems use the industry-standard ringing cycle. That means that callers hear a ring every six seconds, not every four seconds, as on the Definity PBX.

1. Drag a Transfer Call node to the fail branch of the EnterAcctNo node, and name it TransferToOperator.
2. Complete the TransferToOperator node, as shown in the following figure:
 - a. Enter a telephone number.
 - b. Set the Transfer Type to Intelligent.
 - c. Set the Maximum Rings to 4.
 - d. Create a Message to Caller that says:
There was a problem with your entry. Please try again later.



- e. Create a Failure Message that says:
Sorry, no operators are available at this time. Please try again later.
3. Complete both of the open branches with a jump to the HangUp node, as shown in the following figure.



This completes the earning application.

4. Verify and simulate the application.

5. Generate code.
6. If you have a development system, you can transfer, install, and test your application.

To continue, see [Using variables and speech in TAS applications](#) on page 385.

Using variables and speech in TAS 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 tables or 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. The Using variables and speech in TAS applications 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 [Balance]. In this tour, you:

- Learn more about variables.
- Create custom variables with different formats to see how they are spoken.
- Learn more about editing prompts and phrases.

You use two applications:

- spkvar provides callers with the opportunity to enter a number and choose the format to hear it spoken back.
- servic provides callers with customer service menu choices. This application gives you practice in editing prompts and phrases.



Note:

This tour makes use of a TAS 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 [TAS Variable formats](#) on page 386.

Related topics:

[TAS Variable formats](#) on page 386

[Creating and speaking TAS variables](#) on page 391

[Creating phrases in TAS applications](#) on page 395

[Editing prompts in TAS applications](#) on page 397

TAS Variable formats

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.

To continue, see [TAS Variable types and formats](#) on page 386.

Related topics:

[TAS Variable types and formats](#) on page 386

[Using TAS variable attributes](#) on page 387

[TAS Variables and arithmetic operations](#) on page 387

[Converting TAS variable values](#) on page 388

[TAS Spoken As formats](#) on page 388

[Speaking caller input in TAS](#) on page 390

[Using Set and Test to change TAS variable types](#) on page 390

TAS Variable types and formats

Each type of variable has a default length and value, as well as default formats for how it receives input (from the result of a calculation, from a database table read, etc.) and for how it's spoken.

As an example of a Spoken As format, the date 1/1/99 may be spoken as "one, one, ninety-nine," "January one, nineteen-ninety-nine," or in several other ways. You may edit the Spoken As attribute of variables to have their values spoken as your application requires.

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: char or words Input As: char Length: 24 Default: blank

When using Text-to-Speech, the default is to speak the value of a character-type variable as a word. For US English, the character-variable default for Speak Using is Text-to-Speech. If a character variable is defined as Speak Using Recorded, then the default is Spoken As characters.

- Currency accepts numerals (0 through 9), a dollar sign, a leading minus sign, and a decimal point as input.
Spoken As: dollars and cents Input As: #*00 Length: 15 Default: 0

The default is to speak whole dollars. The Speak Using field defaults to Text-to-Speech for US English.

- Date accepts numerals (0 through 9), dashes, slashes, and spaces as input.
Spoken As: (month, day, year) Input As: mmddyy Length: 8 Default: variable creation date in yyyyymmdd format

The Avaya IR or CONVERSANT system stores dates in yyyyymmdd format, despite the fact that the default Spoken As format and the default format for input are mmddyy. Also, Date variables default to the creation date of the application.

- Number accepts numerals (0 through 9) and a leading minus sign as input.
Spoken As: (number) Input As: # Length: 10 Default: 0
- Real accepts numerals (0 through 9), a leading minus sign, and a decimal point as input.
Spoken As: (decimal number) Input As: #.# Length: 10 Default: 0.0
- Time accepts numerals (0 through 9) and the letters A, P, and M as input.
Spoken As: (hours and minutes a.m. or p.m.) Input As: hhmm(AP)M Length: 6 Default: 000000

The actual Spoken As formats are specified in a code, which you learn about later in this lesson. The exact code specification you select depends on whether you are using recorded speech or Text-to-Speech.

To continue, see [Using TAS variable attributes](#) on page 387.

Using TAS variable attributes

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 date be spoken with a two-digit or a four-digit year, for instance, the change is immediately obvious when you call in to test the application.

You may also want to specify a default length or 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.

To continue, see [TAS Variables and arithmetic operations](#) on page 387.

TAS Variables and arithmetic operations

You need to consider operation and variable types when setting up arithmetic operations. The table below summarizes the rules.

Addition and subtraction rules	Multiplication and division rules
• In most cases, addition and subtraction can be performed on only Currency, Number, and Real variables, and only on variables of the same type (for example, Number + Number, Real + Real). • Exception: A Number variable may be added to a Date or a	You can multiply or divide: • Two Number variables, two Real variables, or a Real and a Number variable. • A Number and a Currency variable if the destination variable is a Currency variable.

Time variable. For example, you could find out what date is 32 days past January 12, 1998 by adding the Number value 32 to the Date value 19980112. The resulting value is 19980213, or February 13, 1998. • You cannot add or subtract a Real and a Number variable.	

To continue, see [Converting TAS variable values](#) on page 388.

Converting TAS variable values

When the value in one variable type is assigned to a variable of a different type, IVR Designer attempts to convert the value to the type of the variable receiving the value. Certain types of conversions are not permissible, for example, date to time. If the conversion cannot be accomplished, IVR Designer populates the target variable with the previous value it contained, if there is one. If there is not a previous value, it uses the default value for the target variable type.

How IVR Designer handles the conversion depends on the type of data in the original variable. Issues to consider are:

- **Truncation.** If the length of the variable receiving the value is less than the value being assigned to it, IVR Designer truncates the received value to fit in the target variable.
- **Stripping out.** Certain types of characters that are not permitted in the target variable are removed by the system.
- **Formatting.** The received value is reformatted. For example, when the Character-type value 12-31-97 is received in a Date variable, it is converted to yyyyymmdd format, 19971231. When it is received in a Real type variable, it is converted to 0.0000 format, 12.3197.

Knowing the conversion rules helps you plan prompts for caller input, subsequent operations using the input data, and storage of the caller data in variables of the different types.

To continue, see [TAS Spoken As formats](#) on page 388.

TAS Spoken As formats

The value that appears in the Spoken As column of the Variables Manager depends on the type of variable and the value you have set in the Speak Using column. The options for this column are recorded speech or TTS.

When you make certain changes to variable values, IVR Designer automatically makes corresponding changes, summarized in the table below.

Change	Result
Select a new type for the variable or reassign the Speak Using value.	The variable is assigned a default value in the Spoken As column.
Change the value in the Speak Using column from TTS to recorded or from recorded to TTS.	The value in the Spoken As field automatically changes to the default value for the new Speak Using option.
Change the language of an application.	The Spoken As column is changed to the default value for that language.

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. When you place the cursor over an option on the menu, a context-sensitive pop-up help message displays a description of the option.

Character variables

When you use recorded speech with Character type variables, you can choose from four formats with different inflections:

- Crmf.Speaks the first character with rising inflection, the last character with falling inflection, and all other characters with medial inflection
- Crmm.Speaks the first character with rising inflection and all other characters with medial inflection.
- Cmmf.Speaks the last character with falling inflection and all other characters with medial inflection.
- Cmmm.Speaks all characters with medial inflection.
- C (default).Speaks all characters with medial inflection (same as Cmmm).

When you use Text-to-Speech with Character type variables, you can choose from two formats:

- AC.Spells out the characters in the string. For example, the character string Avaya is spoken as "A V A Y A".
- A (default).Speaks the character string as words, if possible. For example, the character string Avaya is spoken as "Avaya".

Number variables

Your choices for Number type variables are similar.

For recorded speech, the choices are:

- Nrmf.In one thousand five hundred twenty-two, speaks the one with rising inflection, the two with falling inflection, and all other digits with medial inflection.
- Nmmf.In one thousand five hundred twenty-two, speaks the one with falling inflection and all other digits with medial inflection.
- Nrmm.In one thousand five hundred twenty-two, speaks the one with rising inflection and all other digits with medial inflection.
- Nmmm.In one thousand five hundred twenty-two, speaks all digits with medial inflection.

- N (default). Same as Nmmm.
- NX. Speaks the phrase having this number. For example, if the number is 1522, then the phrase recorded at phrase number 1522 from the designated phrase table is spoken, not the number. You do not need to designate a phrase table in the Phrase Table column. The Avaya IR or CONVERSANT system looks in the list of speech phrase tables and find this phrase number. If the phrase is in multiple speech phrase tables, it speaks the first one it finds.

For Text-to-Speech, the choices are:

- AN (default). Speaks the number as a number. For example, "One thousand five hundred twenty-two"
- AC. Speaks the number as individual digits. For example, "One five two two"

Currency variables

For Currency variables, you can specify the number of decimal places to be spoken. The formats have two basic forms. In both of these forms, # equals the number of decimal places you want spoken. The two forms are:

- N\$D# for recorded speech. For example, N\$D2 is recorded speech for currency in dollars with two decimal places. In US English, this variable is spoken as dollars and cents, not as a decimal number.
- AN\$D# for Text-to-Speech. For example, AN\$D0 is Text-to-Speech for currency in whole dollars with no decimal places. If the Spoken As format is AN\$D1, AN\$D3 or AN\$D4, it is spoken as a decimal number.

To continue, see [Speaking caller input in TAS](#) on page 390.

Speaking caller input in TAS

Caller input is always stored in a character-type variable. You cannot speak the value of this variable back in any Spoken As format other than those available for character-type variables. This is true even if you use the default system variable [CollectedDigits], or create a custom variable to store the caller input.

To speak caller input as another variable type, for instance, a date or a dollar amount, you assign the value of the caller input to another variable of the type you want. You use the attribute Save Formatted Response to specify this variable as part of handling the caller response.

To continue, see [Using Set and Test to change TAS variable types](#) on page 390.

Using Set and Test to change TAS variable types

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. For instance, you can assign the value from one variable type to more than one other variable type.

To continue, see [Creating and speaking TAS variables](#) on page 391.

Creating and speaking TAS variables

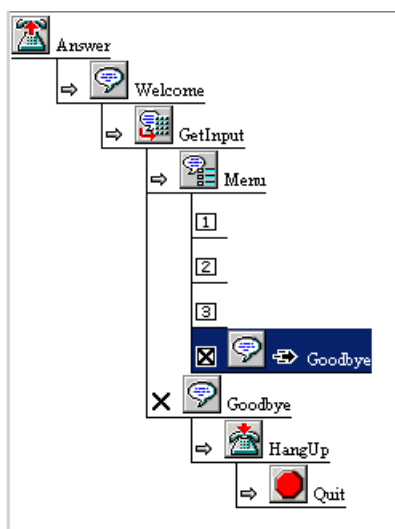
In this topic, you create an application, `spkvar`, that uses different variable types and Spoken As formats. The application asks callers to enter a number. Then callers have a choice of how to have the number spoken back. This is not a realistic application, but it helps you understand the ways that variables are spoken.

The `spkvar` application:

- Answers the telephone.
- 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 `spkvar` application:

1. Open the `sspkvar` TAS tour application provided with IVR Designer, shown in the following figure, and rename it `spkvar`.



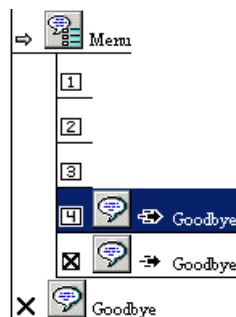
This application already includes many of the required nodes. You add and specify nodes for the menu branches.

2. Review the current application:

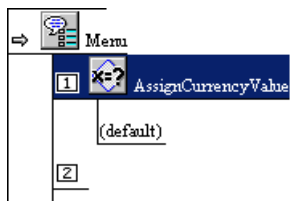
- a. Review the prompts in the Prompt Manager and the phrases in the Phrases Manager.
 - b. Review the parameters for the GetInput node.
Notice the Save Response In parameter on the Response tab. The application has a menu following this Prompt and Collect node. The caller response to the menu overwrites the value in the [CollectedDigits] variable before it can be assigned to another type of variable. To keep the value representing the caller response, the application saves the caller's response in another character variable, called [CharacterEntry].
3. Create the additional custom variables, as shown in the following figure:
- 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. Select AN\$D0 from the drop-down menu for Spoken As.
 - e. Create the remaining custom variables, with the indicated Types and Spoken As formats:
- [MoneyEntryD2] is currency, AN\$D2. [NumberEntry] is number, AN.
[RealEntry] is real, AND2.

Variables	Value	Type	Speak Using	Spoken As	P
[HOURS_CLOSED]		number	tts	AN	
[LANGUAGE]		character	tts	A	
[MoneyEntry]		currency	tts	AN\$D0	
[MoneyEntryD2]		currency	tts	AN\$D2	
[NoDigit]		character	tts	A	
[NumberEntry]		number	tts	AN	
[PivotYear]		number	tts	AN	
[RealEntry]		real	tts	AND2	
[RecordCounter]		number	tts	AN	

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.

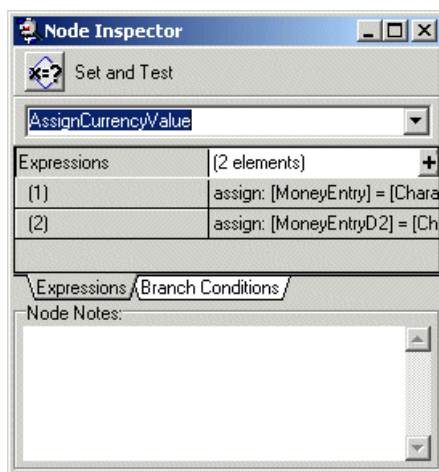


5. Drag a Set and Test node to the first Menu branch and name it AssignCurrencyValue, as shown in the following figure.



This node assigns the value from the [CharacterEntry] variable to the two currency variables, [MoneyEntry] and [MoneyEntryD2].

6. Define the first Assign expression:
 - a. Click + in the Node Inspector.
 - 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.
7. Define the second Assign expression, as shown in the following figure:
 - a. Click + in the Node Inspector.
 - b. Click the new element and click
 - c. Select Assign as the Expression Type.
 - d. Select [MoneyEntryD2] 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 two different currency variables. Now the application needs to announce the values of the variables to the caller.

8. Drag an Announcement node to the open branch under the AssignCurrencyValue node and name it AnnounceCurrencyValue.
9. Drag in the phrases and variables to create the prompt, shown in the following

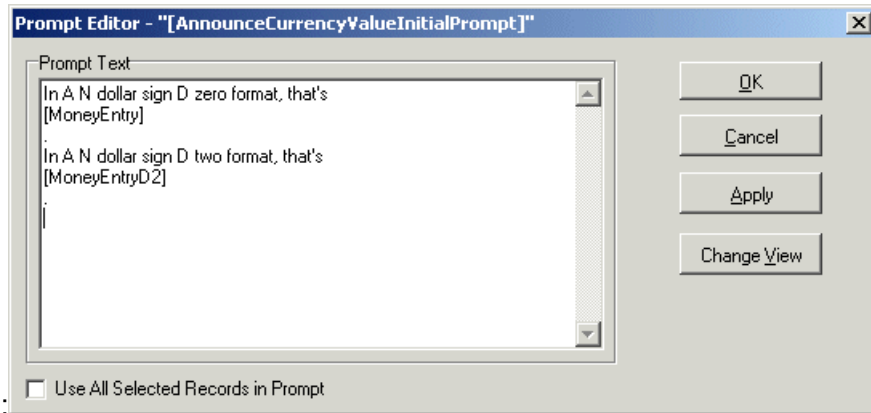
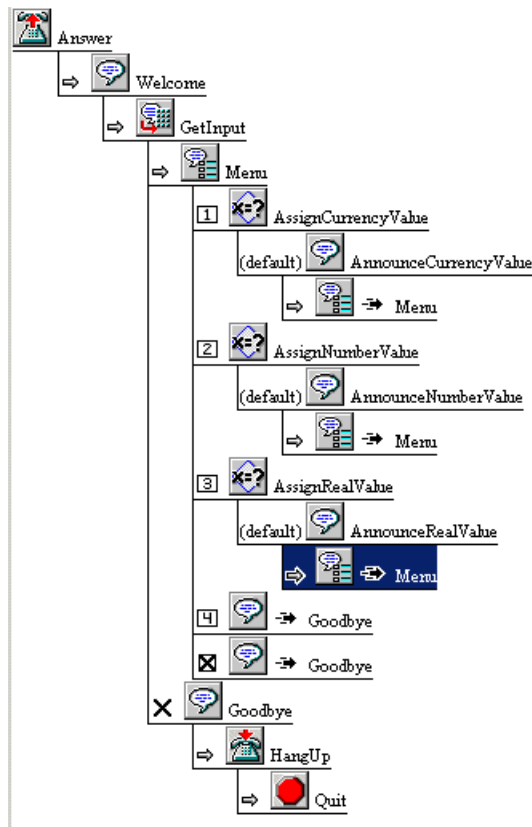


figure:

10. Drag a Set and Test node to the second Menu branch and name it AssignNumberValue.
11. In the Node Inspector, add an element that assigns [CharacterEntry] to [NumberEntry].
12. Add an Announcement node to the open branch under AssignNumberValue, name it AnnounceNumberValue and create an announcement that says:
In A N format, that's [NumberEntry] .
13. Drag a Set and Test node to the third Menu branch and name it AssignRealValue.
14. In the Node Inspector, add an element that assigns [CharacterEntry] to [RealEntry].
15. Add an Announcement node to the open branch under AssignRealValue, name it AnnounceRealValue, and create an announcement that says:
In A N D two format, that's [RealEntry] .
16. Complete the open branches under each of the Set and Test nodes with jumps to the Menu node, as shown in the following figure.



This completes the spkvar application.

17. Verify and simulate the application.
18. If you have a development system, generate code and transfer and install the application.
19. 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 TAS applications](#) on page 395.

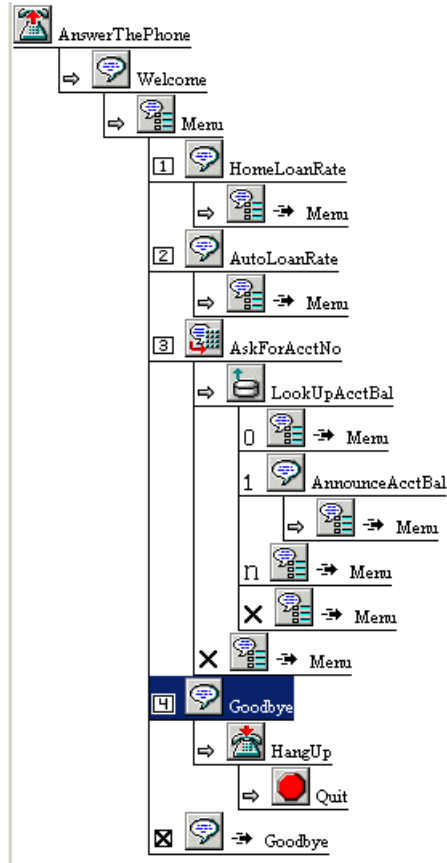
Creating phrases in TAS applications

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 TAS phrase table](#) on page 349 in [Branching based on time, day, or date in TAS applications](#) on page 348. This can be helpful if your design specification includes a list of 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.

1. Open the stelbnk1 TAS tour application provided with IVR Designer, shown in the following figure, and save it as telbank1.



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.

Phrases	Phrase Text
5000	Welcome to the Big Bank telephone banking line.
5001	For account balance, press three.
5002	For today's best home loan rate, press one.
5003	For today's best auto loan rate, press two.
5004	To end the call, press four.
5005	Please enter your six digit account number followed by the pound sign.
5006	The lowest current three year auto loan rate is eight percent.
5007	The lowest current thirty year home loan rate is seven percent.
5008	You must enter a response.
5009	Goodbye
5010	Sorry, the account number you entered is not valid.
5011	You entered too few digits.
5012	I didn't understand you.
5013	You entered
5014	. If this is correct, press one.
5015	If this is incorrect, press any other number.
5016	Your current balance is
5017	.

3. Right-click anywhere in the Phrases Manager and select New.
4. Type the new phrase below and click OK.
To transfer funds between accounts, 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 TAS applications](#) on page 397.

Editing prompts in TAS applications

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.

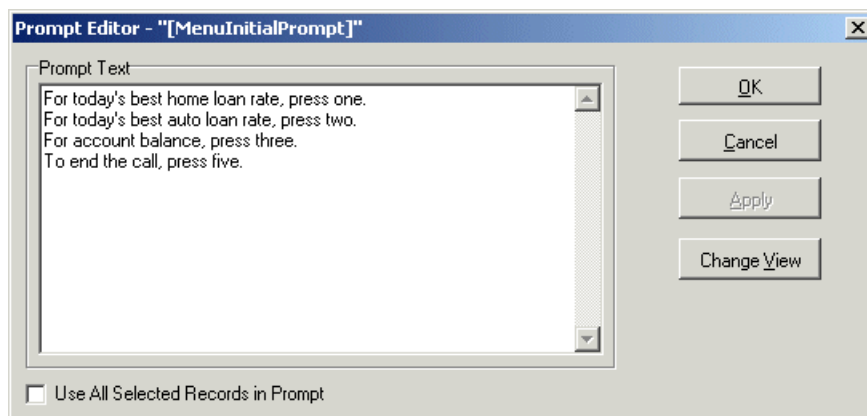
In this topic, you continue developing the telbank1 application that you created in [Creating phrases in TAS applications](#) on page 395. You edit the Menu prompt to include the new option and edit another prompt.

1. Reposition the Prompts Manager so you can view the prompts, as shown in the following figure.

Prompts	Value
[AnnounceAcctBalln]	Your current balance is [Balance] .
[AskForAcctNoBadlr]	Sorry, the account number you entered is not valid.
[AskForAcctNoConfi]	You entered [CollectedDigits] . If this is correct, press one. If this is ir
[AskForAcctNoInitial]	Please enter your six digit account number followed by the pound sig
[AskForAcctNoTooF]	You entered too few digits.
[AutoLoanRateInitial]	The lowest current three year auto loan rate is eight percent.
[HomeLoanRateInitial]	The lowest current thirty year home loan rate is seven percent.
[MenuBadInputProm]	I didn't understand you.
[MenuInitialPrompt]	For today's best home loan rate, press one. For today's best auto ...
[MenuTimeoutProm]	You must enter a response.
[WelcomeInitialProm]	Welcome to the Big Bank telephone banking line.

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, shown in the following figure, is displayed. Notice that the phrase to end the call has been replaced by the edited phrase.

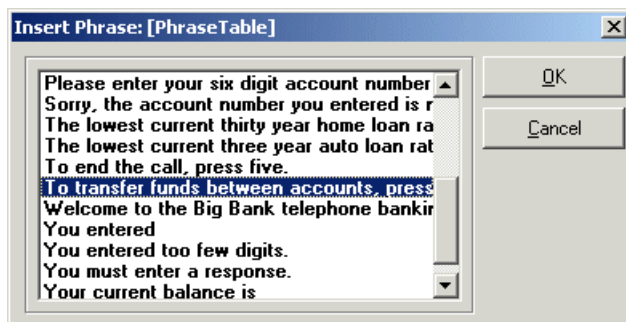


3. Right-click at the beginning of the phrase for ending the call.
Notice that you can insert variables, phrases, phrase breaks, and controls, as shown in the following figure.

Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Delete	Del
Select All	Ctrl+A
Insert Variable...	Ctrl+B
Insert Phrase...	Ctrl+P
Insert Phrase Break	Ctrl+I
Insert Control	Ctrl+L

4. Select Insert Phrase...

The available phrases are listed, as shown in the following figure.

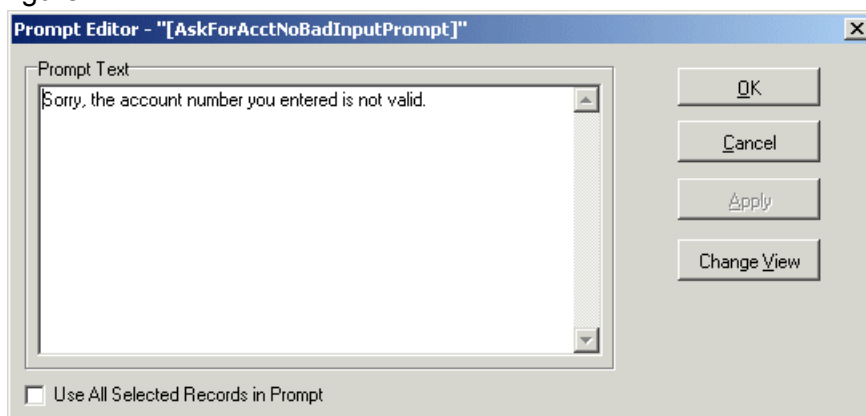


5. Select the phrase to transfer funds and click OK.

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.



You edit this prompt to insert a variable and a pause.

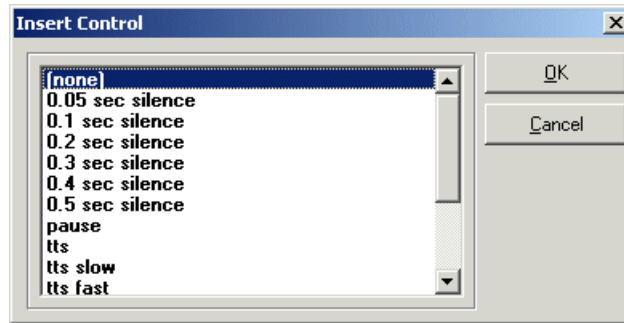
8. Insert the variable [CollectedDigits] after the word entered in this prompt.

- a. Right-click between entered and is.
- b. Select Insert Variable.
- c. Select the [CollectedDigits] variable.
- d. Click OK.

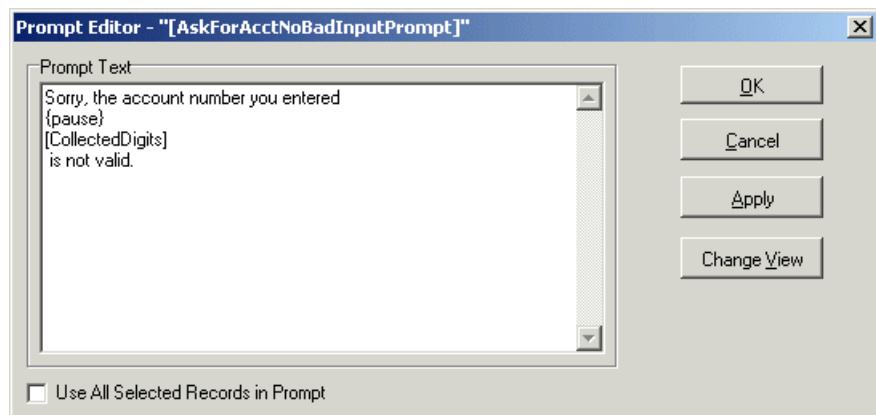
9. Insert a pause before the variable.

- a. Right-click before the variable.
- b. Select Insert Control.

The control options, shown in the following figure, are displayed.



- c. Select pause and click OK.
The variable and pause 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 the application here. If you were to do so, you would:

- Rename the [4] branch to [5].
- Add a [4] branch with the nodes to transfer funds.

To continue, see [Using embedded call flows in TAS applications](#) on page 400.

Using embedded call flows in TAS 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 allows you 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 TAS 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 TAS](#) on page 401.

Related topics:

[Creating an embedded call flow in TAS](#) on page 401

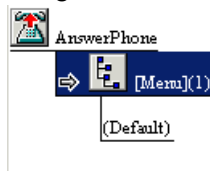
[Encapsulating a TAS call flow](#) on page 403

[Reusing and Copying TAS Call Flows](#) on page 405

Creating an embedded call flow in TAS

In this topic, you create an application, callflw, that uses a Call Flow node.

1. Create a new TAS application and name it callflw.
2. Drag in an Answer Call node and name it AnswerPhone.
3. Drag in a Call Flow node  and name it Menu, as shown in the following figure.

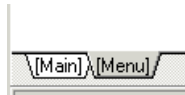


The number 1 in parentheses (1) after the node name indicates that this is the first instance of the node in the application. If you use the [Menu] call flow again later in the application, it will include a two in parentheses (2).

Any return node or 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.

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 is displayed when you edit it.

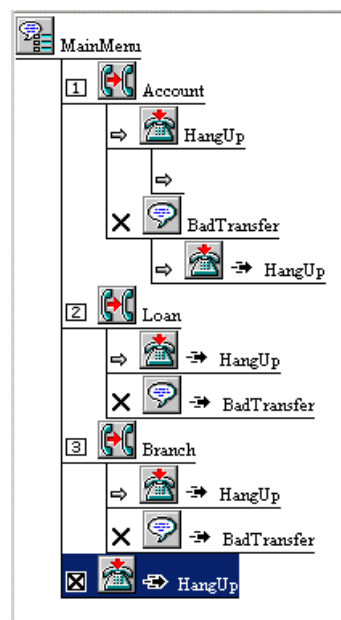
4. Right-click the [Menu] node and select Edit Flow.
There is now a blank design pad for the [Menu] call flow. Notice the new [Menu] tab, shown in the following figure.



5. 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 acts 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.

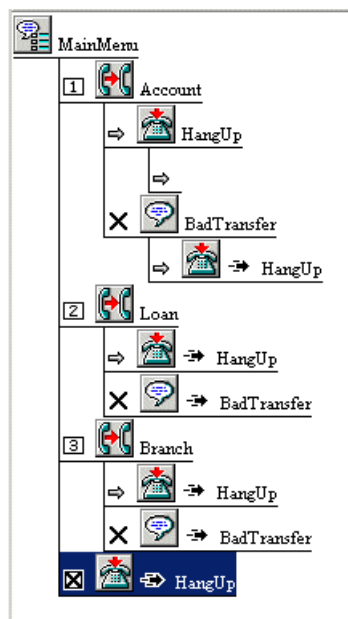
6. Click the [Main] tab to return to the [Main] call flow.
7. Drag a Return node to the open (Default) branch and name it Quit. This completes the menuflw application.
8. Save the application.

To continue, see [Encapsulating a TAS call flow](#) on page 403.

Encapsulating a TAS call flow

The menuflw application is a simple application. You might decide that it does not need a separate call flow for the menu. In this case, you can 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, shown in the following figure. The Quit Return node appears on what was the open branch.



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.

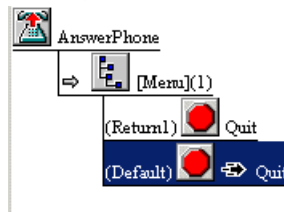
2. Click the Call Flow Inventory  icon to see the call flows in this application, shown in the following figure.



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.

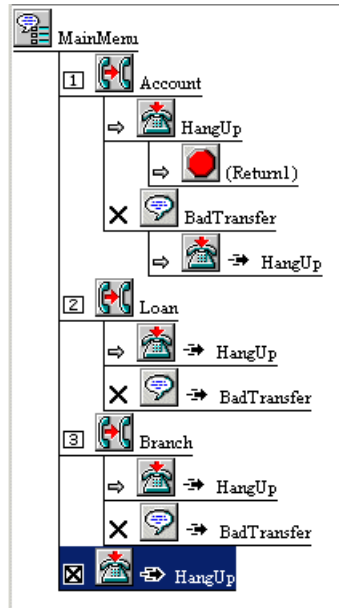
4. Right-click the MainMenu node and select Encapsulate. Name the new call flow Menu, as shown in the following figure.



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.

5. Edit the [Menu] call flow, as shown in the following figure.
Remember that Edit Flow is a right-click option.



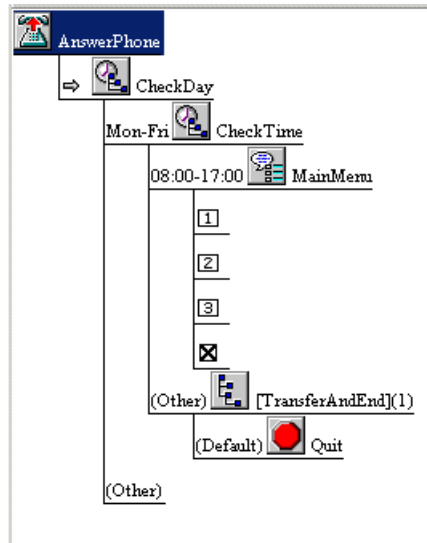
Notice the Return node on the branch under HangUp. This is what caused the (Return1) branch to be added to the [Main] call flow.

To continue, see [Reusing and Copying TAS Call Flows](#) on page 405.

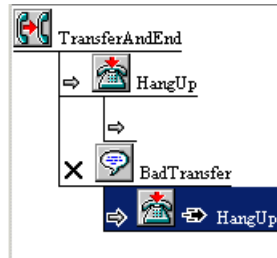
Reusing and Copying TAS Call Flows

You can copy nodes or use a Jump to Node function to repeat actions. Other options are to reuse or copy call flows. In this topic, you complete an application by reusing and copying call flows.

1. Open the stimeflw TAS tour application, shown in the following figure, and rename it timeflw.

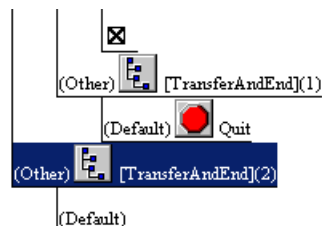


2. Review the existing application.
Notice that this application checks for day of the week. Monday through Friday, it checks for business hours. During work hours, it presents a menu. Outside of work hours, the application transfers the call and ends.
3. Click the [TransferAndEnd] tab to view the call flow, shown in the following figure.



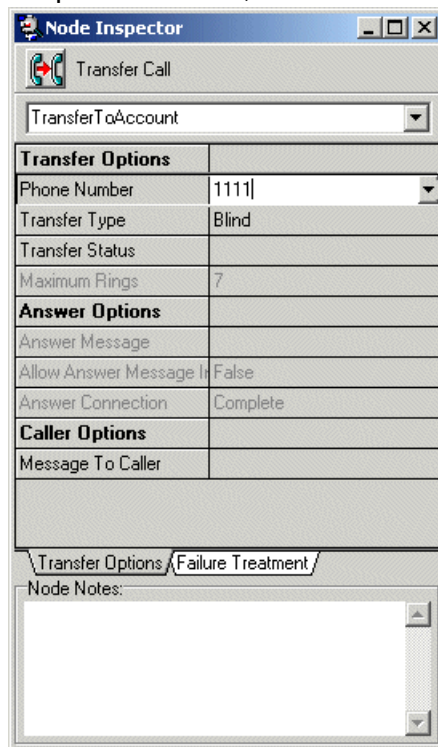
This call flow transfers the call to the message center and hangs up.

4. Return to the [Main] call flow.
Now you specify the (Other) branch under the CheckDay node. This branch should also transfer the call and end. You 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, shown in the following figure, uses the same call flow. Because it is the second instance, it has a 2 at the end.



The new [TransferAndEnd] call flow node uses the same call flow. Because it is the second instance, it has a 2 at the end.

7. Complete the open (Default) branch with a jump to the Quit node.
Next you 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 [Creating database applications](#) on page 408.

Creating database applications

You use the Query Table node to read information from database tables. You can also use database tables to:

- Automatically generate menus.
- Support administrator access for database administration.
- Track system events.

Tracking system events is an important part of administering the Avaya IR or CONVERSANT system. For example, you can track the number of calls that come in on a holiday or identify how many requests for transfers were successful. Such information about system use is critical to making decisions about how to configure your system, and when to add capacity.

Instead of showing all these database functions separately, the topics in this section take you through developing a complex, phased project using databases. The application that you develop:

- Consults a database table to automatically generate a menu of employee names and extensions for callers.
- Consults a database table to announce a special message for holidays.
- Gives an administrator tools to modify the holidays in the database from a Touchtone telephone.
- Tracks system events so that an administrator can generate reports that contain appropriate information.



Note:

This tour makes use of a TAS 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 a database](#) on page 409.

Related topics:

[Creating a database](#) on page 409

[Generating a menu from database records](#) on page 411

[Tracking system events](#) on page 417

[Branching based on table data](#) on page 422

[Providing remote access to database records](#) on page 426

Creating a database

In this topic you create a table that you use in later topics.

The table schema is the form of the table.

- Tables have columns, also called fields. There are two fields in the example you create. The first field is Employee Name and the second is Extension.
- Each row of a table is called a record. When the table is filled with data, or records, each cell of the table contains data associated with a particular field and a particular record.

Imagine that someone calls a company's headquarters and asks to speak to an employee. A receptionist looks up the employee's extension in a list, tells the caller the number, and connects the caller to the employee's extension. If the employee moves to a different extension number, the receptionist updates the list. IVR Designer automates this process by putting the employee names and telephone numbers in a database.

To create the EMPLOYEES table in the data application:

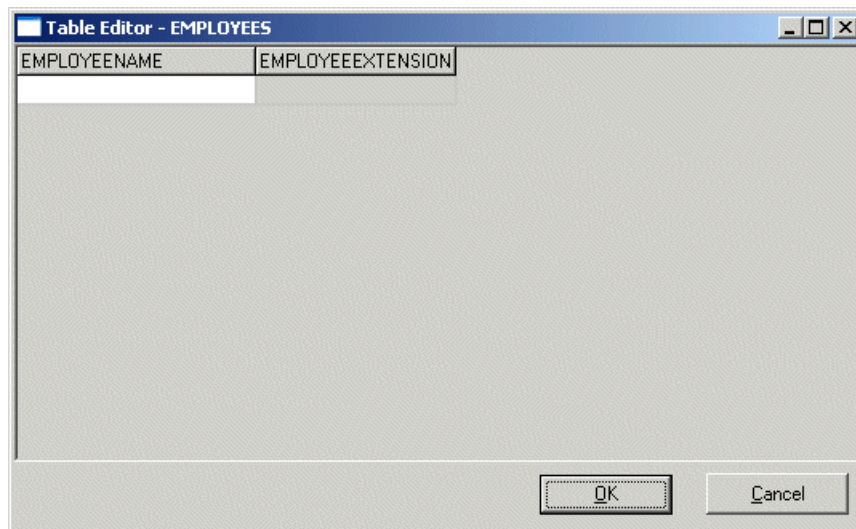
1. Open the sdata application and name it data.
This incomplete application contains most of the prompts and phrases you need for your application and some of the nodes in the [Main] and [List Employees] call flows.
2. Review the application to become familiar with what is already there for you.
3. Right-click anywhere in the Tables Manager and select New.
4. Name the table EMPLOYEES.
The table name must have fewer than 11 characters and begin with a letter.

Use the Table Schema window, shown in the following figure, to define the structure of the table. This table will contain employee names and their telephone extensions. The table has two fields: EMPLOYEE_NAME and EMPLOYEE_EXTENSION.

Key	Name	Class	Format	Length
Nc	UNTITLED1	Character	char	24

5. Under Name, click Untitled One and enter the name of the first field, EMPLOYEENAME.
If you enter the name in lower-case, IVR Designer changes it to upper case.

The EMPLOYEENAME field has the Character Class and the char Format. The default Length is 24 characters. That should be enough for any name.
6. Right-click anywhere below EMPLOYEENAME in the Table Schema window.
7. Select Add Field.
8. Create the EMPLOYEEEXTENSION field with Character Class, char Format, and a Length of 4.
Leave the Key fields set to No for both fields. The Key field is used to prevent duplicate records. If a Key field is set to Yes, no one can create a duplicate of that field. For example, if someone tries to create a second employee record with the name Fred Jones, and the EMPLOYEENAME field has the Key field set to Yes, the application issues an error message.
9. Click OK.
The schema for the EMPLOYEES table is defined. You can now add employee records to the table.
10. Right-click the Table Manager and select Edit Table Data.
Use the Table Editor, shown in the following figure, to add data to the table.



11. Enter the name, John Smith, in the EMPLOYEENAME field.
12. Tab to the EMPLOYEEEXTENSION field and enter 1234.
13. Press Tab to create a new record.
14. Add the following employee names and extensions to the table:
Fred Jones 2345

Mary Evans 3456

Jane Douglas 4567

Four records are enough for now.

15. Click OK to complete the table data.
16. Save the application.

To continue, see [Generating a menu from database records](#) on page 411.

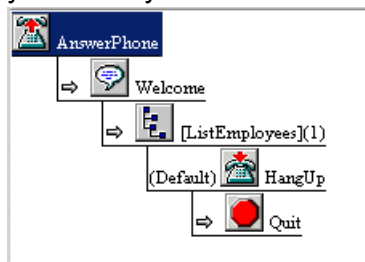
Generating a menu from database records

You use the Automenu node to create menus for callers automatically, based on the contents of a database table that you specify. For example, you can create an application that presents a list of employees from which callers can choose to whom to be transferred. If the employee list changes, you don't need to change the application. You change only the database table of employees.

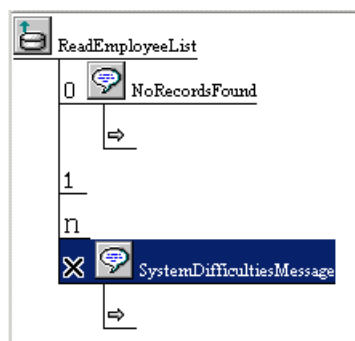
Use the Automenu node to build an application that gives callers the option to transfer to an employee extension. The employee names and extensions are kept in a database so they can be changed without the need to go into the application code. You created this table in the previous topic. It's called EMPLOYEES.

The application queries the table EMPLOYEES and builds a menu to offer callers access to the number for a particular employee by pressing a key on a touchtone telephone. When the caller makes a selection, the employee's extension is confirmed. Then the caller can transfer to that employee extension, select another employee, or end the call.

1. Rename the application, shown in the following figure, data1 to preserve the code you already created.

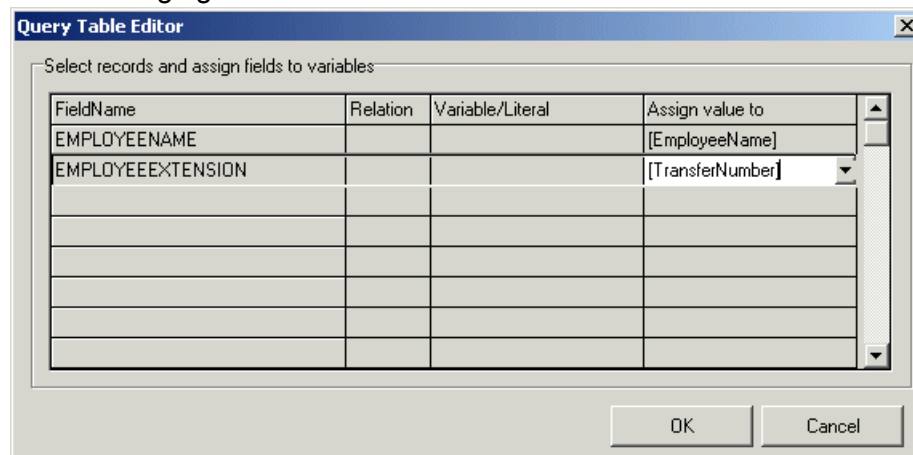


2. Click the [ListEmployees] tab to edit the call flow, shown in the following figure. See [Creating an embedded call flow in TAS](#) on page 401 for more information on editing call flows.



Automenu can work only after the database table has been queried, so the call flow begins with a Query Table node called ReadEmployeeList.

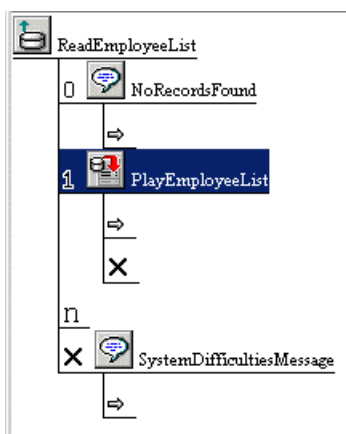
3. View the Node Inspector for the ReadEmployeeList node.
4. Select EMPLOYEES as the Table to query.
When the table is queried, the records it finds are assigned to custom variables. The custom variables store the names and extensions to be read.
5. Click Assignments from Record. Then click ... to edit the assignments.
The Query Table Editor shows the fields available in the table EMPLOYEES.
6. Assign the variable [EmployeeName] to the results of field EMPLOYEEENAME and the variable [TransferNumber] to the field EMPLOYEEEXTENSION, as shown in the following figure.



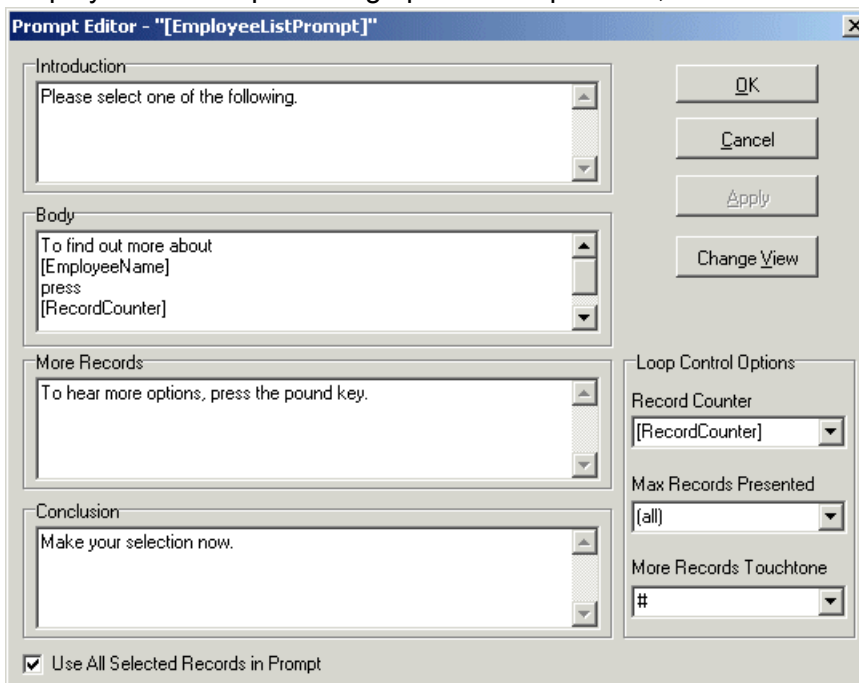
The Announcement node on the Zero Records branch plays a message if there are no records in the database.

When the Query Table node reads a record, there is an automatic match when it assigns the contents to the custom variables [EmployeeName] and [TransferNumber]. Since you do not specify any criteria in this example, every record in the table is selected. The automenu node automatically steps through each record.

7. Drag an Automenu Node  onto the 1 branch of the ReadEmployeeList node and name it PlayEmployeeList, as shown in the following figure.



8. Select EmployeeListPrompt as the initial prompt and double-click EmployeeListPrompt to bring up the Prompt Editor, shown in the following figure.



The Prompt Editor window for '[EmployeeListPrompt]' is shown. It has four main sections for text input:

- Introduction:** "Please select one of the following."
- Body:** "To find out more about [EmployeeName] press [RecordCounter]"
- More Records:** "To hear more options, press the pound key."
- Conclusion:** "Make your selection now."

On the right side, there are buttons: OK, Cancel, Apply, and Change View. Below these are the **Loop Control Options**:

- Record Counter:** [RecordCounter]
- Max Records Presented:** (all)
- More Records Touchtone:** #

At the bottom, the checkbox **Use All Selected Records in Prompt** is checked.

When EmployeeListPrompt was created, the Use All Selected Records in Prompt box was checked, and the Prompt Editor split into four sections:

- **Introduction.** Callers hear the Introduction one time.
- **Body.** The Body repeats, announcing each selected record in the database. This is the automatically generated menu part of Automenu. When you select the Use All Selected Records in Prompt check box, you set up this process.
Notice the two variables in the Body: [EmployeeName] is the variable that is assigned to the table record EMPLOYEE_NAME. [Record Counter] is a system variable that counts each record selected. It is the menu number that the caller hears.
- **More Records.** You can choose not to present all the records at once. Instead, you can bundle records into groups to be announced. When you

bundle records, callers hear the phrase in the More Records portion of the prompt. Lengthy lists should be read out in groups so the caller is not overwhelmed with too many choices. But you do not have to worry about that yet. In this example, the table has only a few names.

In the Max Records Presented field, the (all) option is selected so all of the names in the EMPLOYEES table are read out at once. If a number is entered here, records are read out in groups. For example, if 5 is the entry and there are ten names in the table, the first five are read out to the caller. Then the system plays the More Records message. When the caller presses the pound key, the last group of five names are read. You can change the touchtone from the pound key to some other key by putting it in the More Records Touchtone field.

- Conclusion. The Conclusion field contains the prompt that asks the caller to choose the number of one of the employees.

9. Set Discard Previous Touchtones to False on the Response tab of the PlayEmployeeList node.

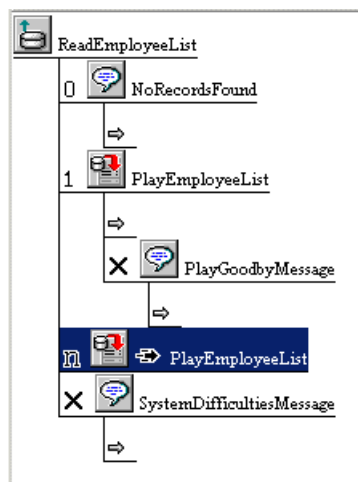
The application can now present a list of employees from which the caller can select. However, it cannot yet fulfill the other requirements, like confirming the selected extension, transferring to the employee extension, or repeating the list. You take care of that later by adding more code for the first branch under PlayEmployeeList.

First, you define the second branch. This branch provides alternatives when the requirements for the menu are not met.

10. Drag an Announcement node to the fail branch of PlayEmployeeList, name it PlayGoodbyeMessage, and select GoodbyeMessage from the drop-down menu for the initial prompt.

This prompt plays if callers do not respond in time to the PlayEmployeeList Automenu or if they key in an inappropriate response to the menu.

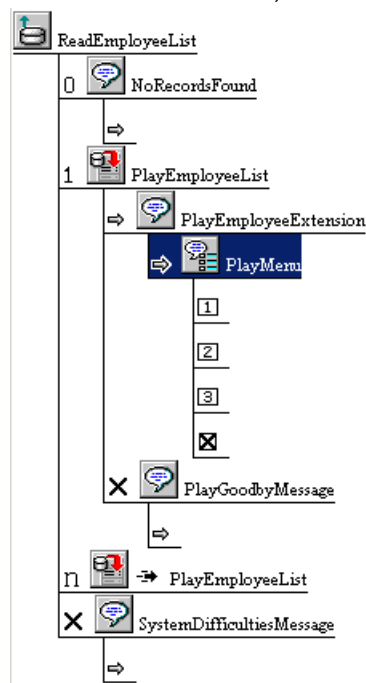
11. Jump to PlayEmployeeList from the n branch, as shown in the following figure. The application takes the n branch of the ReadEmployeeList node when there are multiple matches to the table record selection criteria. Since you did not specify selection criteria in the ReadEmployeeList node, the query returns every record in the table. The same action is required for a single match or multiple matches.



The SystemDifficultiesMessage node that is already on the fail branch of the ReadEmployeeList node plays a message if the table cannot be found.

Now you can return to the first branch under the PlayEmployeeList node to complete it.

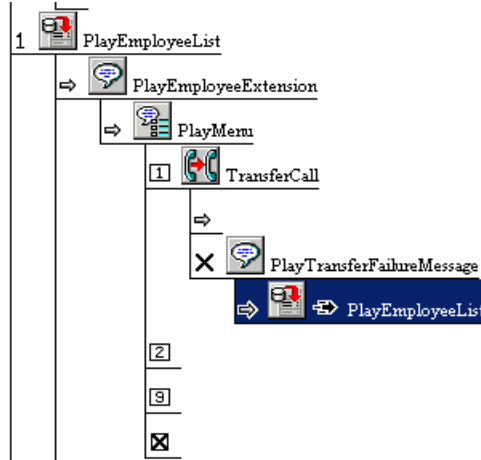
12. Drag an Announcement Node to the first branch under PlayEmployeeList and name it PlayEmployeeExtension. Set the initial prompt to EmployeeExtensionPrompt. When the caller hears the selected employee's name and extension, the system provides three choices: press 1 to transfer the call to the extension, press 2 to play the employee list again, and press 9 to quit.
13. Drag a Menu node to the open branch under PlayEmployeeExtension and name it PlayMenu. Set its initial prompt to MenuPrompt and set Discard Previous Touchtones to False, as shown in the following figure.



By default, the menu provides three touchtone responses to the caller: 1, 2, and 3. This application uses 1, 2, and 9.

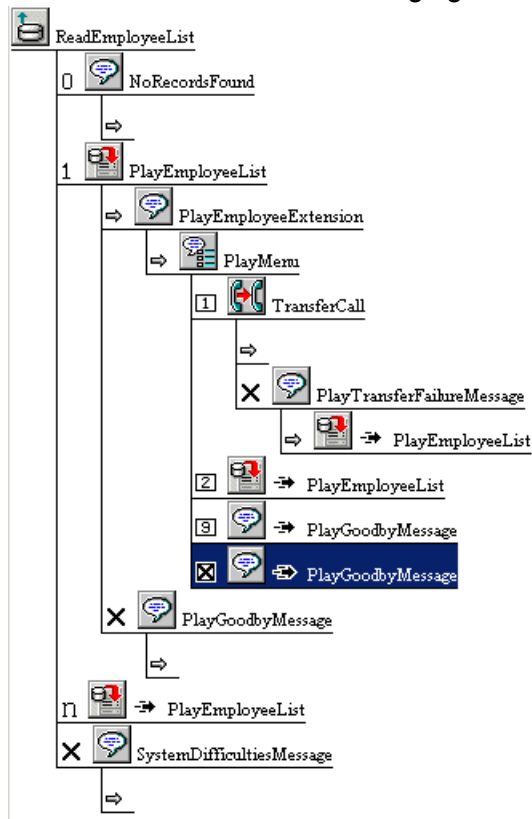
14. Use the right-click option to rename branch 3 to branch 9. Now there are three touchtone options for callers: 1, 2, and 9. If the caller takes too long to respond or enters an inappropriate response, the fail branch is taken.
15. Drag a Transfer Call node to [1] branch and name it TransferCall.
16. Select the [TransferNumber] variable as the Phone Number. The node transfers callers to the extension contained in the variable [TransferNumber].
17. Select TransferMessage as the Message to Caller.
18. Drag an Announcement node to the fail branch of TransferCall and name it PlayTransferFailureMessage.

19. From the open branch under PlayTransferFailureMessage, jump to PlayEmployeeList, as shown in the following figure.



If the transfer fails, the system plays a failure message and then repeats the employee list.

20. Jump to PlayEmployeeList from branch [2].
21. Jump to PlayGoodbyeMessage from branch [9] and from the fail branch of the menu, as shown in the following figure.



This completes the data1 application.

22. Save the application.

23. Verify and simulate the application.
24. If you have a development system, generate code, transfer, and install the application.

To continue, see [Tracking system events](#) on page 417.

Tracking system events

When a call goes to a Avaya Interactive Response system, the application that is run stores the values of any variables that are designated as Call Data Events. At the end of each call, the application writes these variable values to a database on the Avaya Interactive Response system. The system stores the values for use in Call Data Summary and Call Data Detail reports.

Call data information is part of a database of system events that also includes error and administrative messages, call classification (call disposition) information, and call volume information. The system uses the database information when you create reports on system usage or system problems.

There are two types of call data events:

- Generic data applies to every call and is captured automatically by the system.
- Application-specific data is captured and saved as Call Data Events only if you specify that the system should do so.

You use the IVR Designer Events String Editor tool to:

- Tailor call data reports so that they show the last value stored in specific variables in your application.
- Add, delete, and order the variables that are handled as application-specific Call Data Events.

Call Data Summary report

The Call Data Summary Report provides an hour-by-hour summary of calls for the day specified. The report includes the number of calls, the service (application), and the average hold time. If you specify application-specific call data events, these also appear in the Call Data Summary report. You may specify the name of the call data variable, an event number, and the total number (or value) of these events for all calls during the period. The Call Data Summary report is run nightly for the previous 24 hours.

Call Data Detail report

The Call Data Detail report presents information from the last seven days on:

- Date, time, and duration of each call
- Service (application) run on the call
- Channel that received the call
- Assigned database record number of each call

Calls are ordered in the report from most recent to oldest. The data for the current day is also available in this report.

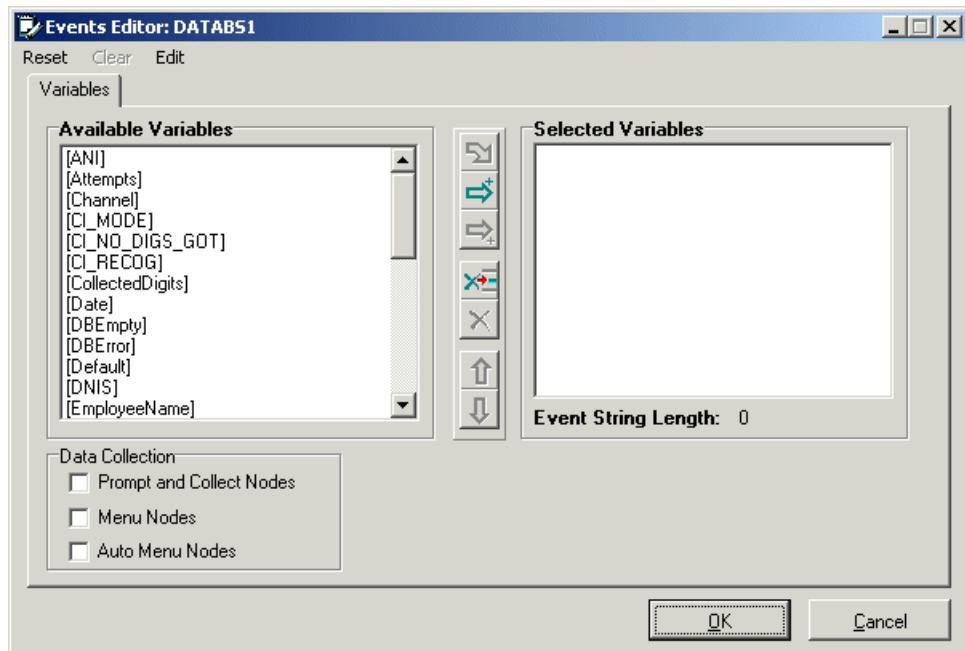
Modifying the application

In Tracking system events, you modify the data1 application to add the capability to record certain events that occur when handling calls. The system records:

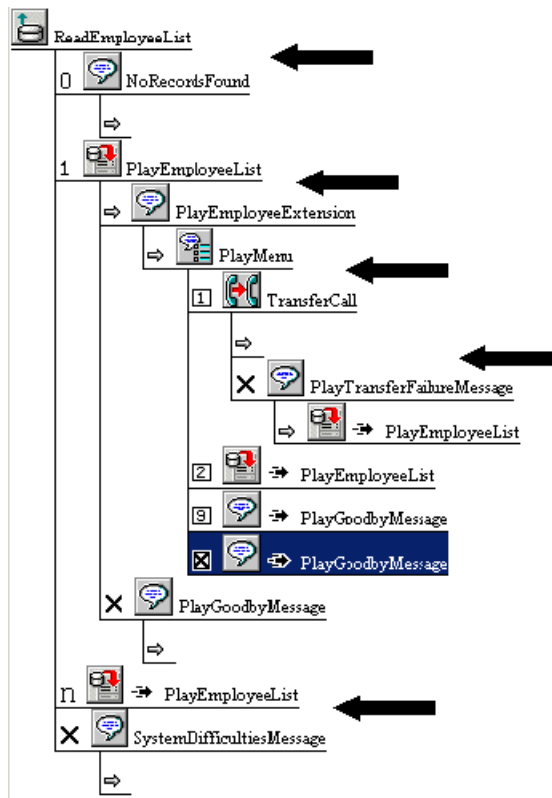
- Instances when no employee records are found in the database
- Errors that occur during database access
- Call transfer information, that is, when a transfer is attempted and if the transfer fails.

To modify the application:

1. Rename the data1 application data2.
2. Go to the [ListEmployees] call flow.
3. Add the following custom variables:
 - [DBEmpty] with Type number, Speak Using tts, Spoken As AN, Length 10
 - [DBError] with Type number, Speak Using tts, Spoken As AN, Length 10
 - [EmployeesSelected] with Type number, Speak Using tts, Spoken As AN, Length 10
 - [EmployeeTransfers] with Type number, Speak Using tts, Spoken As AN, Length 10
 - [TransferFailures] with Type number, Speak Using tts, Spoken As AN, Length 10
4. Click the Event String Editor  icon on the Toolbar.
Using the Event String Editor, shown in the following figure, you can specify variables to be tracked as Call Data Events. The values of these Call Data Events can be retrieved from the Avaya IR system by executing Call Data Summary and Call Data Detail reports.

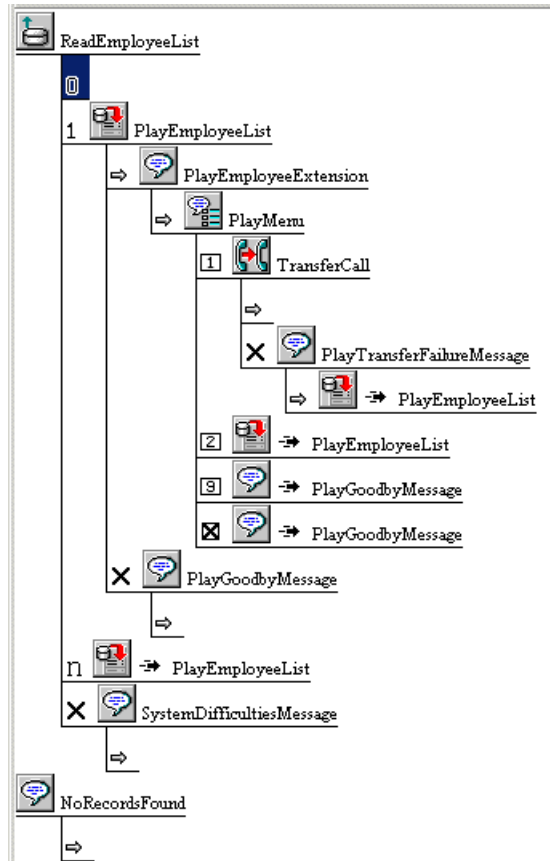


5. Select each of the variables you just created and click the Append Variable to Event String  icon to append each variable to the event string so can be reported. Set and Test nodes are required to monitor whether particular events have occurred and to assign values to these variables. The arrows in the figure below indicate where the Set and Test nodes must be added.

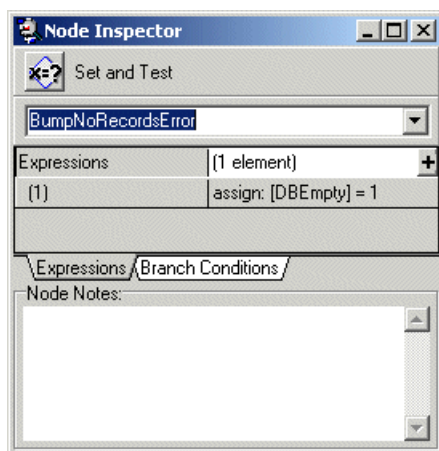


Before you can insert a Set and Test node, you must make space for it in the call flow. To do this, disconnect a node and later reconnect it.

6. Right click the NoRecordsFound node and select Disconnect Node. The NoRecordsFound node is disconnected and is displayed at the bottom of the design pad, as shown in the following figure. You may have to scroll to see it.



7. Drag a Set and Test node onto the 0 branch of ReadEmployeeList and name it BumpNoRecordsError. BumpNoRecordsError means that the application increments, or bumps up, the count by one every time the system cannot find records in the EMPLOYEE database.
8. Define one expression for the BumpNoRecordsError node. Assign the value of 1 to the [DBEmpty] variable, as shown in the following figure. This expression assigns the value 1 to the variable [DBEmpty] if there are no records in the database. The failure to find any records is an event that can happen only once per call. The value of the variable is either zero or one, so the easiest way to record the event is to assign the value of 1 if it happens.



9. Reconnect the NoRecordsFound node by dragging it to the (default) branch under the new Set and Test node, as shown in the following figure.

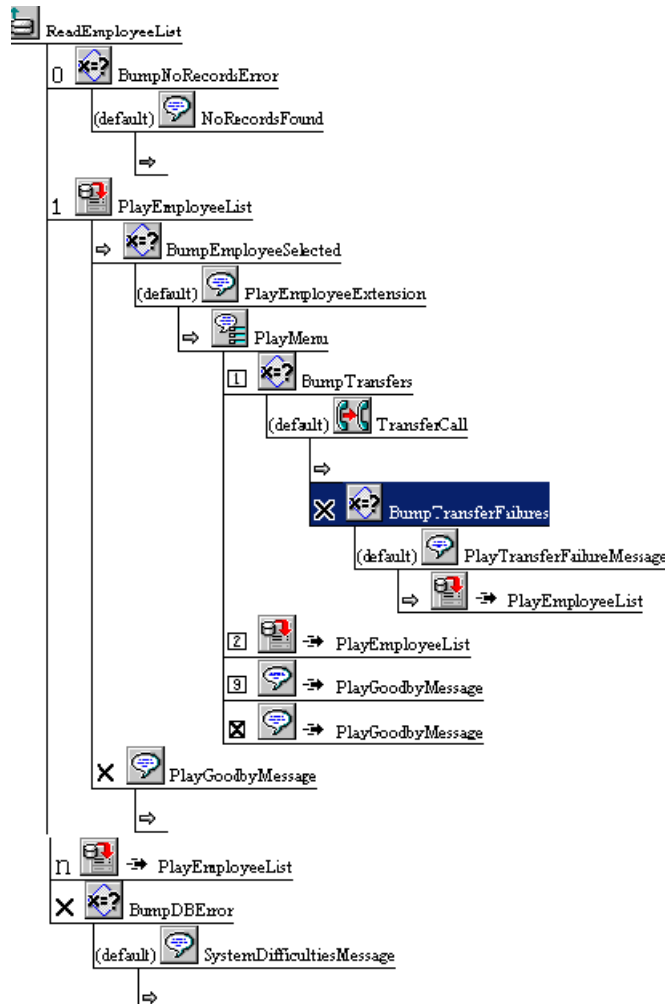


The next Set and Test node records when the caller has selected an employee from the list.

10. Disconnect the appropriate node, add each of the remaining Set and Test nodes, and reconnect the disconnected node as shown in the following table:

Located before	Node name	Expression
PlayEmployeeExtension	BumpEmployeeSelected	arithmetic: [EmployeesSelected] = [EmployeesSelected] + 1
TransferCall	BumpTransfers	arithmetic: [EmployeeTransfers] = [EmployeeTransfers] + 1
PlayTransferFailureMessage	BumpTransferFailure	arithmetic: [TransferFailures] = [TransferFailures] + 1
SystemDifficultiesMessage	BumpDBError	assign: [DBError] = 1

1. Save the application.
The completed application is shown in the following figure.



2. Verify and simulate the application.
3. If you have a development system, generate code and transfer and install the application.
After you install the application, you can access the values stored in the Call Data Event variables by running Call Data Summary Reports and Call Data Detail Reports on the Avaya IR system.

To continue, see [BBranching based on table data](#) on page 422.

Branching based on table data

The data application should handle calls that come in on holidays differently from calls that come in on days that are not holidays. Holiday calls should be gracefully terminated and the event should be logged. You can use either Time Branch nodes or Query Table nodes to direct application flow.

Branching with Time Branch nodes

You can use a Time Branch node with a number of branches, each for a specific holiday date. This is a good approach if you want to hard-code the holidays, or if you don't mind going into

the code and changing the branches as the holiday dates change. It's especially useful if you want a unique flow for each individual holiday, such as playing a different greeting for each one.

Branching with Query Table nodes

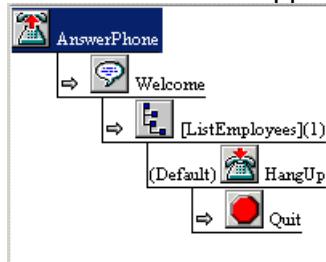
If you want the same application flow for any holiday, use a Query Table node that searches a database table of holidays. You define the Query Table node to search the table. In the Node Inspector, you set selection criteria to select a record if the value of the field in the table matches the value of the system date variable. If there is a match, the application branches in a way that is appropriate for holidays. If there is not a match, the application branches as it would when it is not a holiday. You can update the database table of holidays without revising the application code. The bank holidays change from year to year, so Query Table nodes work better for this requirement.

In Branching based on table data, you:

- Create a new prompt to tell callers that the bank is closed on the holiday.
- Define new custom variables that:
 - Hold the value of the holiday date extracted from a database of holidays.
 - Record the number of holiday calls.
- Create a new database table that contains the list of holidays.
- Add new code that:
 - Checks to see if the day is a holiday.
 - Bumps up the holiday call counter if the day is a holiday.
 - Plays a message to the caller before hanging up.

To complete the above tasks:

1. Rename the data2 application, shown in the following figure, data3.



2. Add the following variables:

- [HolidayCalls] with Type number, Speak Using tts, Spoken As AN, Length 10
- [HolidayDate] with Type date, Speak Using tts, Spoken As ADMSPDY, Length 8

3. Click the Event String Editor icon and add the variable [HolidayCalls] to the Call Data Events.
4. Add a new [ClosedMessage] prompt that says:
I'm sorry. We are closed for the holiday. Please try your call again later.
5. Create a database table for holidays with the following schema and data:
Table: HOLIDAYS

Field Name: HolidayDate with Class date, Format yyymmdd, Length 8, Key No

Contents:

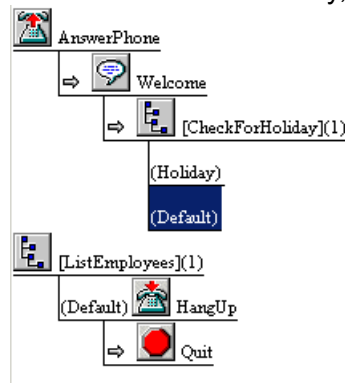
1 20020101

2 20020704

3 the date for today

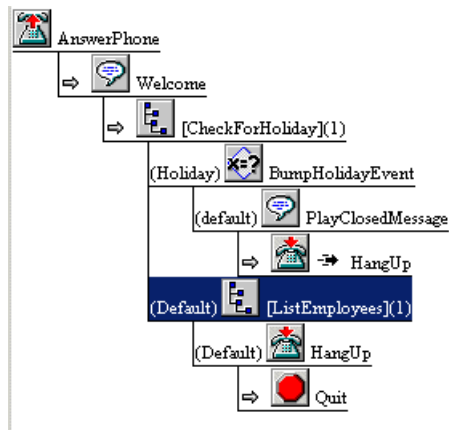
The best place in the call flow to check for holidays is after the Welcome node but before the employee names are listed.

6. Disconnect the ListEmployees node.
7. Drag a Call Flow node to the open branch under Welcome and name it CheckForHoliday.
Do not edit this new call flow yet. First, you add branches to handle two possible events. These branches take action when there is a match with a holiday in the table, and when there is no match. These branches represent the two different possible outcomes of the CheckforHoliday call flow. The first branch increments the [HolidayCall] counter and hang up. The other lists employees if it is not a holiday.
8. Right-click the (Default) branch under CheckForHoliday and select Insert Branch. Name the branch Holiday, as shown in the following figure.

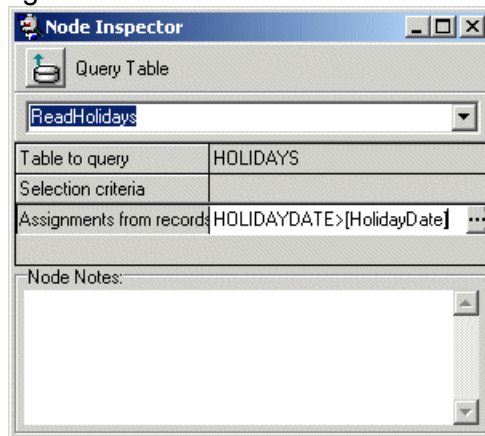


If there is a match with the HOLIDAY table, the application logs this event.

9. Drag a Set and Test node to the (Holiday) branch and name it BumpHolidayEvent. Assign the number 1 to the Call Data Event variable [HolidayCalls].
10. Drag an Announcement node to the (Default) branch below the BumpHolidayEvent node and name it PlayClosedMessage. Select [ClosedMessage] as the initial prompt.
11. Jump to the HangUp node from the open branch below the PlayClosedMessage node.
12. Reconnect the ListEmployees node to the (Default) branch under the CheckForHoliday node, as shown in the following figure.



13. Edit the CheckForHoliday node.
Because you added the (Holiday) branch to the [Main] call flow, a Return node, named Holiday, appears in this call flow.
14. Drag a Query Table node before the Holiday node and name it ReadHolidays.
Notice that the Return node is not attached to this node. You must define where it goes.
15. Define the Assignments from records for this node. ReadHolidays looks in the HOLIDAYS table to see if any of the records in HOLIDAYDATE field match the system date, contained in the system variable [Date], as shown in the following figure.



If there is a match on a holiday, the application returns to the Holiday branch on the [Main] call flow.

16. Connect the detached Return node onto the 1 branch.
If there is one match, this branch is taken. If there are duplicate records, there may be more than one match on a holiday.
17. Jump to the Holiday node from the n branch.
If there are no matches, the call flow returns to the default ListEmployees branch on the [Main] call flow.
18. Save the application.

19. Verify and simulate the application.
20. If you have a development system, generate code and transfer and install the application.

To continue, see [Providing remote access to database records](#) on page 426.

Providing remote access to database records

You may want to provide access to database administration functions over the telephone. For example, in the case of holidays, you may want an administrator to change the holidays without having to go to a computer. You coded holiday message announcements in the data application using Query Table nodes. You can give an administrator telephone access to change the data in the table. If you had used Time Branch nodes, you would not be able to do this.

In this topic, you modify the data application to add the capability for an administrator to use a touchtone telephone to add, delete, and list holidays in the HOLIDAYS database table. Access to these administrative tasks is by password only.

The code you add:

- Greets the caller, pauses to give an administrator the opportunity to enter a password, and checks to see if the day is a holiday.
- Takes action based on password validity:
 - If the password is valid, gives the caller access to the holiday administrative functions.
 - If the password is invalid or if no password is entered, continues as in the previous version.
- Presents a menu of holiday administrative functions:
 - Keypress 1, add a holiday
 - Keypress 2, delete a holiday
 - Keypress 3, list the holidays
 - Keypress 9, end the call
- Provides the holiday administration functions.

In Providing remote access to database records, you:

- Add 13 prompts to guide the administrator through the new tasks.
- Create 2 custom variables—one to hold the administrative password and one to contain a new holiday to be added by the administrator.
- Create 5 call flows that handle administrative tasks.

Adding variables and prompts

To modify the application by adding variables and prompts:

1. Rename the data3 application data4.
2. Add the following variables:
 [AdminPassword] with Type character, Speak Using tts, Spoken As A, Length 6, Default *12345

 [NewHoliday] with Type date, Speak Using tts, Spoken As ADMSPDYY, Length 8
3. Add the following new prompts:
 [AddHolidayErrorMessage] The new holiday [NewHoliday] was not added.

 [ConfirmNewHoliday] You have entered [NewHoliday] {pause} If this is correct, press one. If not, press two.

 [DeleteHolidayErrorMessage] The holiday [HolidayDate] was not deleted.

 [DeletedHolidayMenuPrompt] [HolidayDate] {pause} To delete, press one. To skip, press two.

 [Goodbye] Goodbye.

 [HolidayAddedMessage] [NewHoliday] has been added.

 [HolidayDeleted_Message] [HolidayDate] has been deleted.

 [HolidayMenu] To add a holiday, press one. To delete a holiday, press two. To list the holidays, press three. To end the call, press nine.

 [HolidayPrompt] [HolidayDate] {pause}

 [NewHolidayPrompt] Enter the new holiday in Y Y Y Y M M D D format.

 [NoHolidaysMessage] There are no holidays in the table.

 [NoMoreRecords] There are no more holidays.

 [TooFewHolidays] You must enter four digits for the year, two digits for the month, and two digits for the day.

Modifying the [Main] call flow

To accept an administrator password, you replace the Welcome Announcement node with a Prompt and Collect node.

To modify the [Main] call flow:

1. Disconnect the Welcome node.
 There is no way to transform one type of node into another type, so you remove this node.
2. Drag a Prompt and Collect node onto the open branch and name it PlayGreeting.
 The initial prompt is the one you used in the original Welcome node. Everyone hears this prompt, including the administrator. However, only the administrator knows he or she can enter a password at this time.
3. On the Response tab, set Allow Interrupt to True, Discard Previous Touchtones to False, and Maximum Length and Minimum Length to 6.
4. On the Validation tab, set the Validate Response Against field to Variable, the Relation field to =, and the Variable field to [Admin Password].

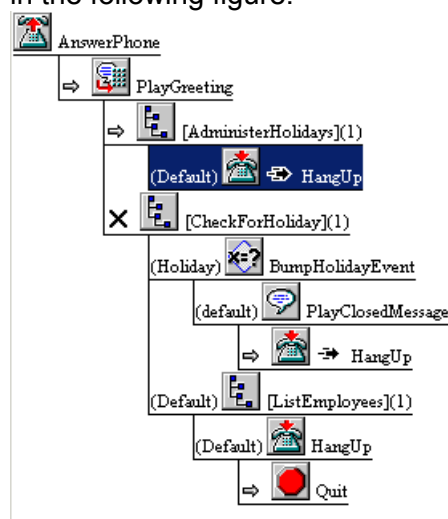
This means that the caller must enter characters that match what is stored in [Admin Password]. When you created the [Admin Password] variable, you set its default value to the password *12345.

With the correct password, the caller is sent to the first branch. This is the way administrators access the administrator functions. Otherwise, the caller goes to the fail branch. That is, the application checks if it is a holiday and processes the call as before. That is where to put the detached CheckForHoliday node.

5. Drag the CheckForHoliday node to the fail branch under PlayGreeting.
6. Right-click the Welcome node and select Delete Node.
The [Main] call flow is now reassembled with the substituted PlayGreeting Prompt and Collect node. Next you add the code that executes if the password is correctly entered and the first branch is taken.

The administrator should be able to add, delete, and list holidays and then end the call. That means four separate routines.

7. Drag a Call Flow node to the open branch and name it AdministerHolidays.
8. Jump to the HangUp node from the open branch under AdministerHolidays, shown in the following figure.



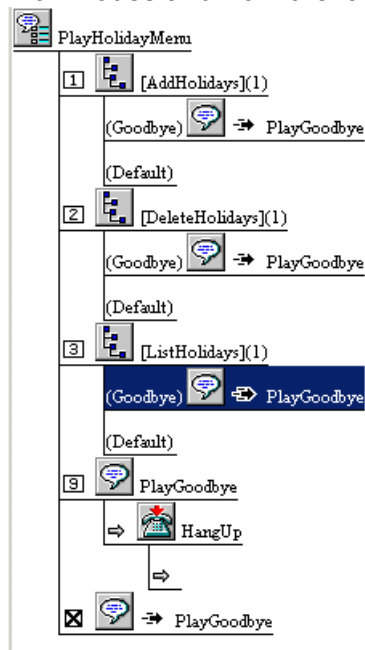
Creating the [AdministerHolidays] call flow

In the [Main] call flow, you insert a call flow to administer holidays. It is composed of another call flow that includes a menu of the tasks that the administrator can perform. Each task is handled by another call flow. The holiday administrative call flow presents a menu of holiday administrative functions:

- Keypress 1, add a holiday
- Keypress 2, delete a holiday
- Keypress 3, list the holidays
- Keypress 9, end the call

To create the [AdministerHolidays] call flow to provide the administrative options:

1. Edit the [AdministerHolidays] call flow.
2. Drag a Menu node into the call flow, name it PlayHolidayMenu, and select HolidayMenu as the initial prompt.
This prompt offers the caller four choices: 1 to add a holiday, 2 to delete a holiday, 3 to list the holidays, and 9 to end the call.
3. Add the branch for 9.
4. Add Call Flow nodes for the routines to add, delete, and list holidays and name them AddHolidays, DeleteHolidays, and ListHolidays.
Choice 9 plays a Goodbye message and disconnects.
5. Add an Announcement node to the [9] branch, name it PlayGoodbye, and select Goodbye as the initial prompt.
6. Drag a Disconnect Call node to the open branch under PlayGoodbye and name it HangUp.
When the administrator returns from any of the call flows, the default branches jump to the holiday menu again. If callers fail to enter a response or provide bad input inside one of these call flows, they return to this call flow, play the Goodbye message and disconnect. This requires additional branches under each call flow node.
7. Add branches under each of the Call Flow nodes and name them Goodbye.
8. Jump to the PlayGoodbye node from the Goodbye branches under each of the Call Flow nodes and from the fail branch of the menu, as shown in the following figure.

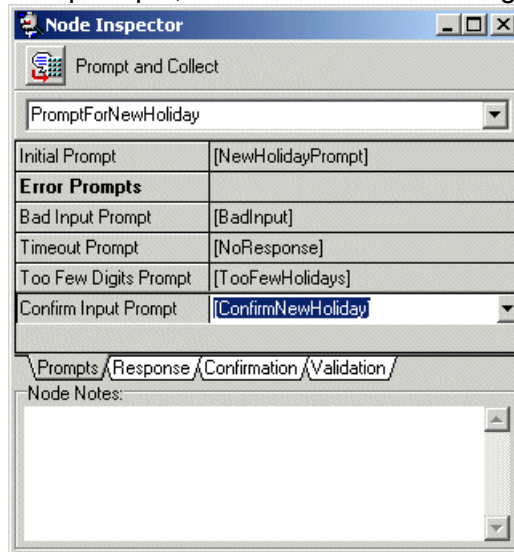


Creating the [AddHolidays] call flow

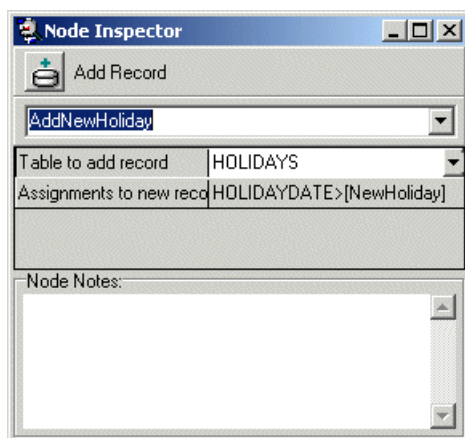
The call flow to add a holiday directs the administrator to use the telephone keypad to enter the date of the new holiday. The format for the date is specified in the announcement. After the administrator keys in the date, the application confirms that it has added the holiday or announces an error. If the administrator does not respond or keys in inappropriate data, the application plays a Goodbye announcement.

To create the [AddHolidays] call flow in which the administrator can add holidays to the table:

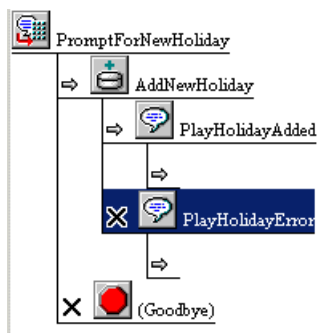
1. Edit the [AddHolidays] call flow.
2. Drag a Prompt and Collect node before the Return node and name it PromptForNewHoliday.
PromptForNewHoliday collects the new holiday the administrator wants to add to the HOLIDAYS table. The caller enters the digits requested by the prompt.
3. Add prompts, as shown in the following figure:



4. On the Response tab, set the following:
 - Allow Interrupts to True
 - Discard Previous Touchtones to False
 - Maximum Length and Minimum Length to 8.
 - Save Formatted Response in to [NewHoliday]
This variable is used to announce the added holiday and to enter the new date into the table.
5. Drag an Add Record  node to the pass branch of the PromptForNewHoliday node and name it AddNewHoliday.
The Add Record node puts the new date into the HOLIDAYS database table.
6. In the Node Inspector, specify HOLIDAYS as the table and assign the contents of the [NewHoliday] variable to the HOLIDAYDATE field in the table, as shown in the following figure.



7. Drag an Announcement node to the pass branch of the AddNewHoliday node, name it PlayHolidayAdded, and use HolidayAdded as the initial prompt.
 8. Drag an Announcement node to the fail branch of the AddNewHoliday node, name it PlayHolidayError, and use AddHolidayErrorMessage as the initial prompt.
 9. Drag the Goodbye Return node to the fail branch of the PromptForNewHoliday node.
- This completes the [AddHoliday] call flow, shown in the following figure.



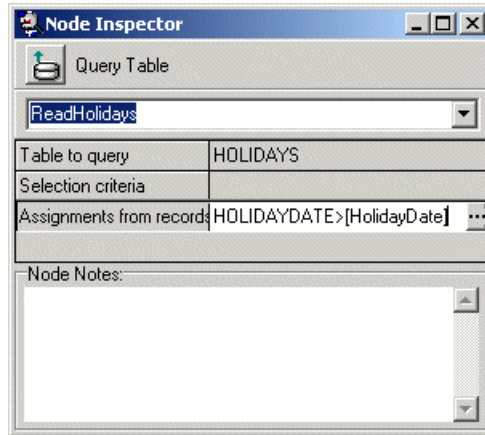
Create [DeleteHolidays] call flow

The [DeleteHolidays] call flow does the following:

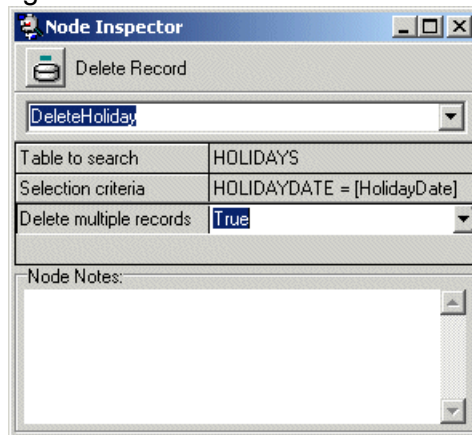
- Checks the HOLIDAYS table.
- If no records are found, plays a message and returns to the previous call flow.
- If there are any records, selects a date from the HOLIDAYS table and announces it, giving the administrator the option to delete it or skip it.
- If there are system troubles, plays a message and exits.

To create the [DeleteHolidays] call flow in which the administrator can delete holidays from the table:

1. Return to the [AdministerHolidays] call flow and edit the [DeleteHolidays] call flow.
2. Drag a Query Table node before the Return node and name it ReadHolidays.
3. In the Node Inspector, specify HOLIDAYS as the table, and assign the contents of the next HOLIDAYDATE field in the table to the [HolidayDate] variable, as shown in the following figure.



4. Drag an Announcement node to the 0 branch, name it PlayNoHolidays, and select NoHolidaysMessage as the initial prompt.
5. Drag a Menu node to the 1 branch, name it PlayDeleteHolidayMenu, and select DeleteHolidayMenuPrompt as the initial prompt.
The caller presses 1 to delete the announced holiday or 2 to skip to the next holiday in the table.
6. Drag a Delete Record  node to the [1] branch of the PlayDeleteHolidayMenu node and name it DeleteHoliday.
The Delete Record node finds any records that match the selected holiday and deletes them.
7. Complete the Node Inspector for the DeleteHoliday node as shown in the following figure.

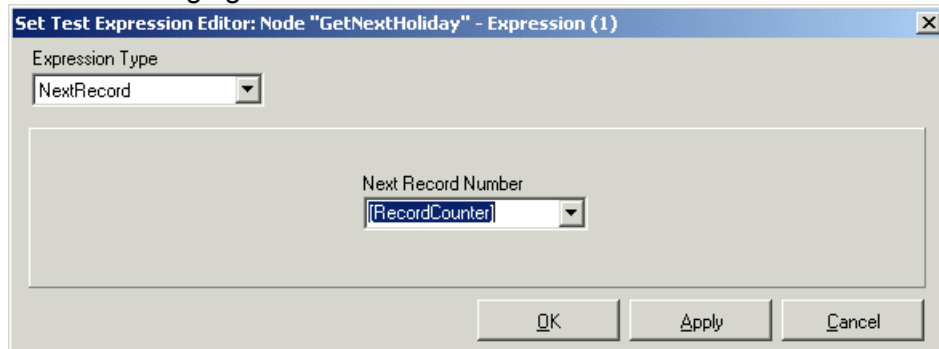


The parameters shown in the figure above cause the application to search for any record that matches the contents of the variable [HolidayDate] and delete them. If there are multiple records for the same holiday, all are deleted.

8. Drag an Announcement node to the pass branch under DeleteHoliday, name it PlayHolidayDeleted, and use HolidayDeletedMessage as the initial prompt.
9. Drag an Announcement node to the fail branch under DeleteHoliday, name it PlayDeleteError, and use DeleteHolidayErrorMessage as the initial prompt.
10. Drag a Set and Test node to the [2] branch and name it GetNextHoliday.

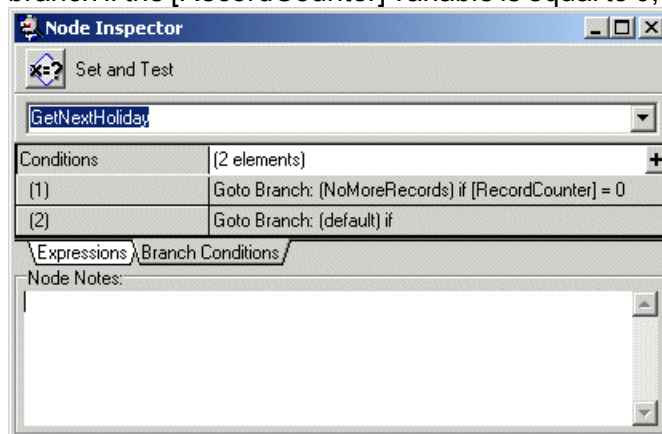
If the caller presses 2 to skip the current holiday record and hear the next one, this node searches for the next record.

11. Add a branch before the (Default) branch and name it NoMoreRecords. This branch is taken when there are no more records in the HOLIDAYS table.
12. In the Node Inspector for the GetNextHoliday node, create an expression of NextRecord type that uses [RecordCounter] for the Next Record Number, as shown in the following figure.



The expression looks for the next record number in the system variable [RecordCounter].

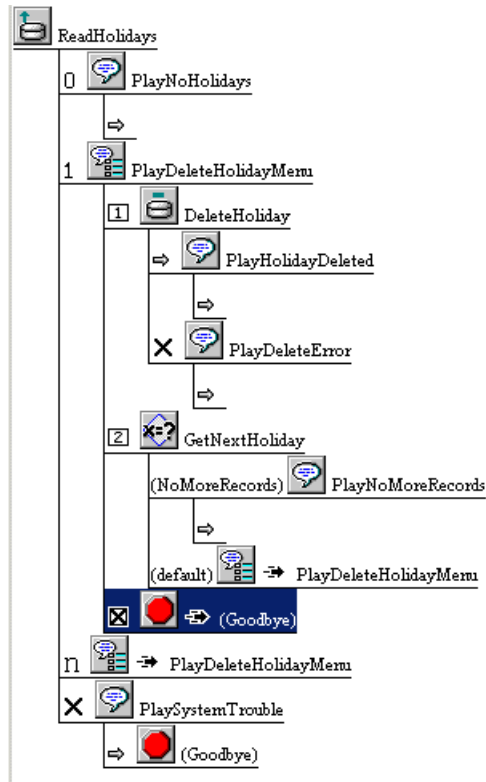
13. On the Branch Conditions tab, add a condition that branches to the NoMoreRecords branch if the [RecordCounter] variable is equal to 0, as shown in the following figure.



14. Drag an Announcement node to the NoMoreRecords branch, name it PlayNoMoreRecords, and use NoMoreRecords as the initial prompt.
15. Jump back to the (Default) branch of the GetNextHoliday node. This returns to the menu to give the administrator the option to delete the next holiday from the table.
16. Delete the [3] branch.
17. Jump to PlayDeleteHolidayMenu from the n branch. The administrator has the same delete options if more than one record is found.
18. Drag an Announcement node to the fail branch of the ReadHolidays node, name it PlaySystemTrouble, and use SystemDifficultiesMessage as the initial prompt.

19. Connect the Goodbye Return node to the open branch under PlaySystemTrouble.
20. Complete the open fail node of PlayDeleteHolidayMenu with a jump to the Goodbye node.

This completes the [DeleteHolidays] call flow, shown in the following figure.

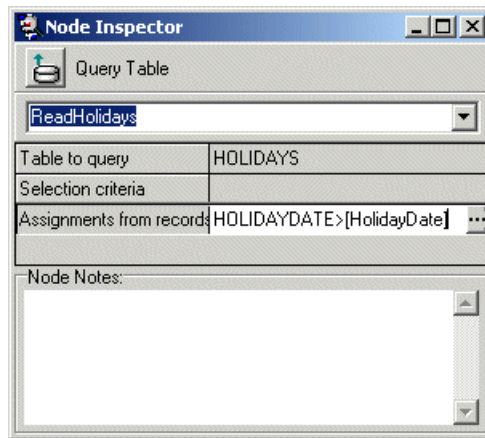


Creating the [ListHolidays] call flow

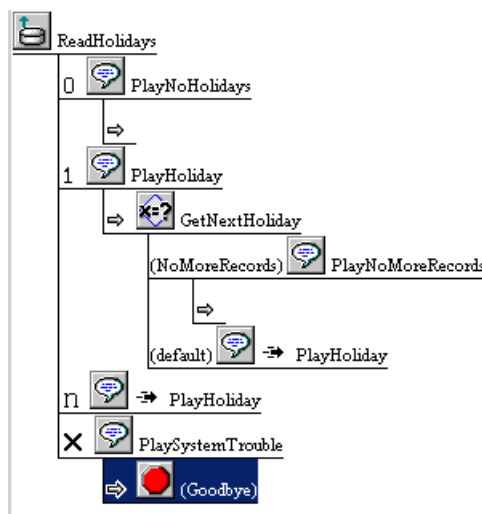
The call flow to list holidays reads all the dates in the table. If there are no holidays in the list, when the last holiday is reached, or if there is database trouble, the call flow makes an appropriate announcement.

To create the [ListHolidays] call flow in which the administrator can hear all the holidays in the table:

1. Return to the [AdministerHolidays] call flow and edit the [ListHolidays] call flow.
2. Drag a Query Table node before the Return node and name it ReadHolidays.
3. In the Node Inspector, specify HOLIDAYS as the table and assign the contents of the next HOLIDAYDATE field in the table to the [HolidayDate] variable, as shown in the following figure.



4. Drag an Announcement node to the 0 branch, name it PlayNoHolidays, and select NoHolidaysMessage as the initial prompt.
5. Drag an Announcement node to the 1 branch, name it PlayHoliday, and select HolidayPrompt as the initial prompt.
6. Copy the GetNextHoliday node from the DeleteHolidays call flow to the open branch under PlayHoliday.
7. Complete the (Default) branch of the GetNextHoliday node with a jump to PlayHoliday.
8. Complete the n branch with a jump to PlayHoliday.
9. Drag an Announcement node to the fail branch of ReadHolidays, name it PlaySystemTrouble, and use SystemDifficultiesMessage as the initial prompt.
10. Connect the Goodbye Return node to the open branch under PlaySystemTrouble. This completes the [ListHolidays] call flow and the data4 application, shown in the following figure.



11. Save the application.
12. Verify and simulate the application.

If you simulate from the beginning of each call flow, you can test the call flows.

13. If you have a development system, generate code and transfer and install the application.

To continue, see [Advanced TAS application development tours](#) on page 436.

Advanced TAS application development tours

The Advanced TAS application development tours provide you with instructions for creating applications that give you practice with some of the more advanced features of Avaya IVR Designer. You should have a good understanding of the basic IVR Designer features before you complete these tours.

You can view and print the entire Advanced TAS application development tour section in PDF format.

To continue, see [Handling strings in TAS applications](#) on page 436.

Related topics:

- [Handling strings in TAS applications](#) on page 436
- [Using asynchronous events in TAS applications](#) on page 451
- [Using external functions in TAS applications](#) on page 456
- [Passing data between TAS applications](#) on page 480
- [Using stored procedures in TAS applications](#) on page 483
- [Capturing voice from callers in TAS applications](#) on page 487
- [Creating host applications](#) on page 496
- [Avaya Interaction Center integration](#) on page 508
- [Predictive Dialing System \(PDS\) integration](#) on page 514
- [Testing and troubleshooting TAS applications](#) on page 521
- [Securing user data](#) on page 526

Handling strings in TAS 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, retrieved from a database table, 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 tour provide examples of how to manipulate strings in applications.

To continue, see [Using the TAS Length expression](#) on page 437.

Related topics:

[Using the TAS Length expression](#) on page 437

[Using the TAS ItemCount and ItemOf expressions](#) on page 440

[Using the TAS Concat expression](#) on page 442

[TAS Substring and Parse comparison](#) on page 446

[Using the TAS Substring expression](#) on page 447

[Using the TAS Parse expression](#) on page 450

Using the TAS Length expression

The Using calculations in TAS applications tour 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 specified 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.

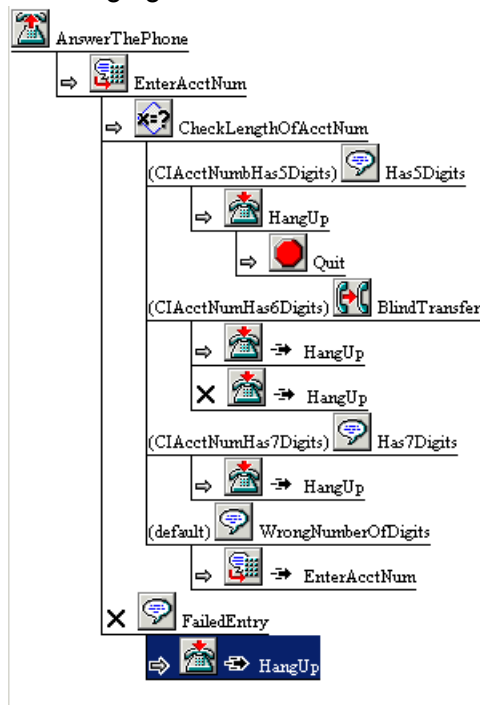
The application length:

- Answers the telephone.
- 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 character-type custom variable [CIAcctNum].
- Counts the number of characters in [CIAcctNum] and stores the number in a number-type custom 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 of 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.

To create the length application:

1. Create a new TAS application named length.
2. Create the following custom variables:
 - [CIAcctNum]. A character variable with a Maximum Length of 7 and the Spoken As field set to AC
 - [LengthOfCIAcctNum]. A number variable with the Spoken As field set to AC
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 EnterAcctNum node so that:
 - The caller hears:
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 [CIAcctNum].
 - The default settings are used for all the other attributes.

5. Edit the CheckLengthOfAcctNum node as follows:

- Expressions tab:
Length: [LengthOfCIAcctNum] = [CIAcctNum]
- Branch Conditions tab:
If [LengthOfCIAcctNum] = 5 Goto Branch (CIAcctNumHas5Digits)
If [LengthOfCIAcctNum] = 6 Goto Branch (CIAcctNumHas6Digits)
If [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 the + icon 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 BlindTransfer 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 FailedEntry node associated with the fail branch of the EnterAcctNum node, so that the caller hears the Text-to-Speech message:
You did not enter a valid account number. 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, install, 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 TAS ItemCount and ItemOf expressions](#) on page 440.

Using the TAS ItemCount and ItemOf expressions

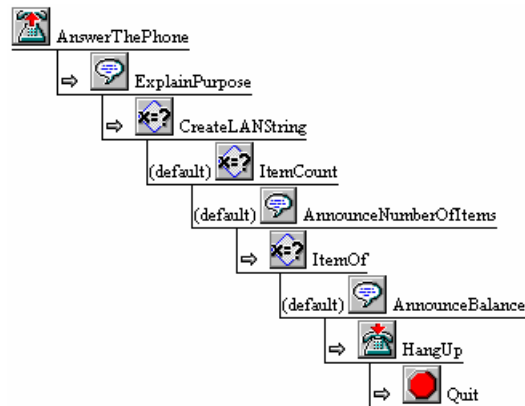
In this topic, you develop an application that uses the Set and Test ItemCount and ItemOf expressions to handle strings.

The application:

- Answers the telephone.
- 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 item application:

1. Create a new TAS application named item.
2. Add and name the nodes as shown in the following figure.

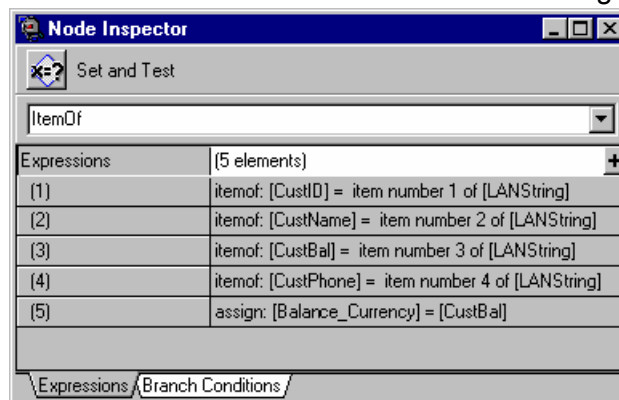


3. Create the following variables for use in the Set and Test nodes:

- [LANString]. Character variable with a maximum length of 256 characters
 - [NumberOfItems]. Number variable with default length
 - [CustID]. Character variable with a maximum of 9 characters
 - [CustName]. Character variable with default length
 - [CustBal]. Character variable with default length
 - [CustPhone]. Character variable with default length
 - [Balance_Currency]. Currency variable with the Spoken As field set to AN \$D2
- See [Creating and speaking TAS variables](#) on page 391 for more information on creating custom variables.

4. Specify the nodes, using the default values, except as indicated below:

- Edit the CreateLANString node to assign the following string value to the [LANString] variable:
123456789,DaleSmith,77.77,3035551212
- Edit the ItemCount node to count the number of items in the [LANString] variable, using the ItemCount expression and assign the result to the [NumberOfItems] variable.
- Edit the ItemOf node as shown in the following figure:



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] {pause}
 - ItemOf initial prompt:
The customer [CustName] has a balance of [Balance_Currency] {pause}
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, install it, assign it to a channel, and test it.

To continue, see [Using the TAS Concat expression](#) on page 442.

Using the TAS 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 complete number is available, the application:

- Uses the complete account number to look up balance information in a table.
- Announces the requested balance to the customer.

The concat application does the following:

- Answers the telephone.
- Welcomes the caller.
- Prompts the caller to enter an account number followed by the pound sign, and saves the information 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 and places it in a complete account number variable with a maximum length of 10.
 - If 2 is pressed, concatenates the caller-input account number with the savings extension sv and places it in a complete account number variable with a maximum length of 10.
 - If the Prompt and Collect or the account-type menu query fails, plays a problem message, hangs up, and quits.
- Queries a table of bank accounts for the account balance associated with the complete account number.

If the database query fails, encounters no matches, or encounters more than one match, the application plays the following database problem message: "Sorry, we are experiencing database problems. Please try again later. Goodbye." It then hangs up and quits.
- Announces the account balance, then hangs up and quits.

To develop the concat application:

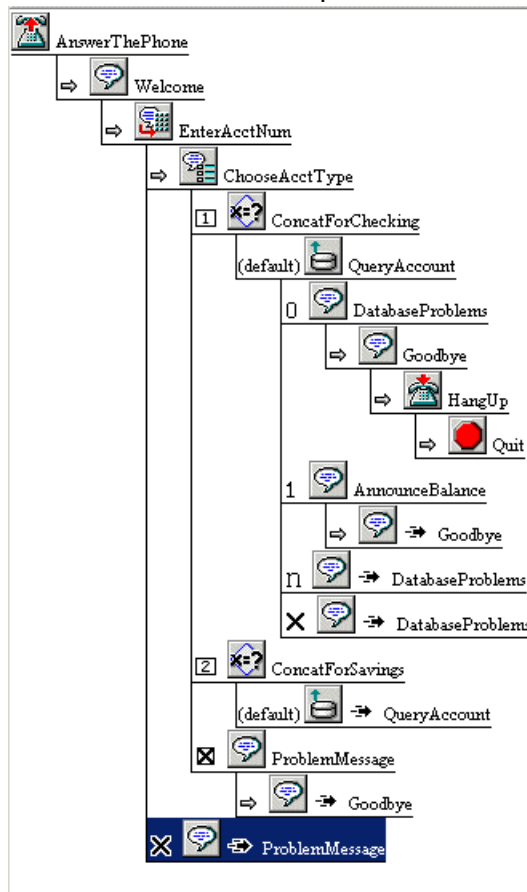
1. Create a new TAS application named concat.
2. Create the custom variables required by the application requirements:
 - [CIAccountNumber]. Character variable with a maximum length of 8
 - [CompleteAccountNumber]. Character variable with a maximum length of 10
 - [CheckingExtension]. Character variable with a maximum length of 2 and a default value of ck
 - [SavingsExtension]. Character variable with a maximum length of 2 and a default value of sv
 - [VBalance]. Currency variable for the account balance to be announced as AN\$D2
3. Create a database table named ACCTS

The table [ACCTS] has two fields:

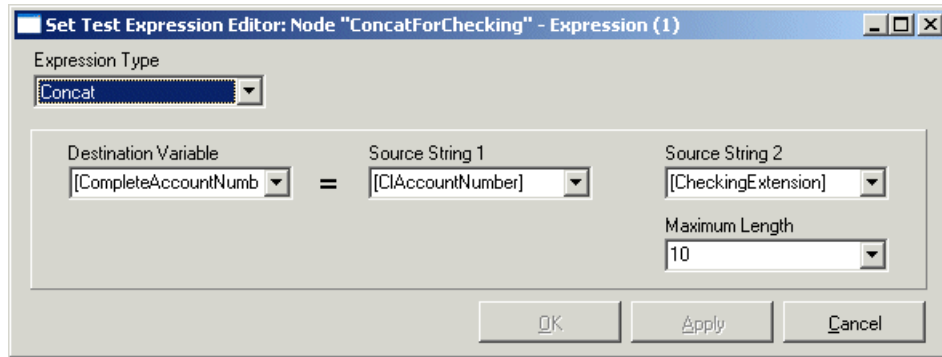
 - ACCTNUM is a character field with a length of 10
 - BALANCE is a currency field with a length of 15
4. Create the following data entries in [ACCTS]:

ACCTNUM	BALANCE
12345ck	\$45.00
12345sv	\$3,000.00
654321ck	\$25.00
654321sv	\$75.75
7777777ck	\$50.00
7777777sv	\$1000.00
33344477ck	\$3333.33
33344477sv	\$4444.44

1. Create the call flow to look like the sample call flow in the following figure. Name the nodes as in the sample.



2. 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].



3. 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.
4. 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.
 - DatabaseProblems initial prompt:
I'm sorry, we are experiencing difficulties, please try again later.
 - AnnounceBalance and AnnounceBalance2 initial prompt:
Your balance is: [VBalance]
 - ProblemMsg initial prompt:
I'm sorry, we are experiencing difficulties, please try again later.
5. Verify and simulate the application, using the following scenarios:
 - When prompted for the account number, enter: 12345#. When prompted for an account type, enter: 1. The balance should show as \$45.00.
 - When prompted for the account number, enter: 33344477#. When prompted for an account type, enter: 2. The balance should show as \$4444.44.
 - When prompted for an account number, do not respond. The application should prompt you three times for an account number, and then respond with the ProblemMsg announcement.

- When prompted for an account number, enter: 654321#. When prompted for an account type, do not respond. The application should prompt you three times for the account type, and then the simulation should end.
 - When prompted for an account number, enter: 12345678#. When prompted for an account type, enter either 1 or 2. The application should respond with the DatabaseProblems announcement.
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 channel.
 7. Test and trace the application.
 - a. Test all 16 known combined account numbers in the [ACCTS] table and make sure the correct balance is returned.
 - b. Test by entering 55555 and then selecting 1. The ProblemMsg prompt should be heard.

To continue, see [TAS Substring and Parse comparison](#) on page 446.

TAS 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:

Counting from left to right	0	1	2	3	4	5	6	7	8

Character string (for example: SSN)	1	3	5	6	6	6	0	8	5
Counting from right to left	-9	-8	-7	-6	-5	-4	-3	-2	-1

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 TAS Substring expression](#) on page 447.

Using the TAS Substring expression

In this topic, you develop an application that uses the Set and Test Substring expression. The application collects a nine-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.

This application:

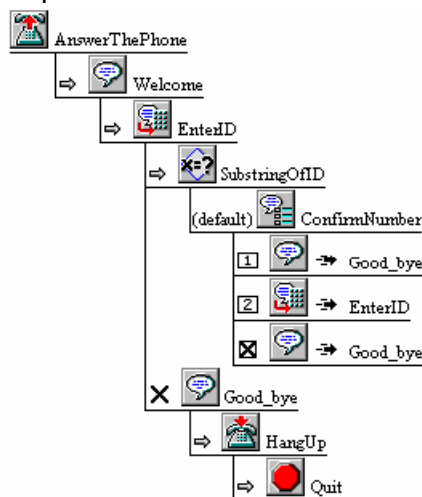
- Answers the telephone.
- Makes an announcement explaining the purpose of the application.
- Prompts the caller to enter a nine-digit ID number.
- Reprompts if the caller does not enter a 9-digit number within the default time-out period.

- 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.
- 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 substrg application:

1. Create a new TAS application named substrg.
2. Create the custom variables as specified below:
 - [CI_ID]. Character variable with a maximum length of 9
 - [FirstThree]. Character variable with the first three digits of the ID number, the Spoken As field set to AC, and a maximum length of 3
 - [MiddleTwo]. Character variable with the second two digits of the ID number, the Spoken As field set to AC, and a maximum length of 2
 - [LastFour]. Character variable with the final four digits of the ID number, the Spoken As field set to AC, and a maximum length of 4

To select the values for the variables holding the digits of the ID number, the Substring expression counts from left to right.
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 I D 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 the same kinds of responses you used for the application simulation.
 - a. When prompted, enter: 123456789
Verify that you hear the correct response and can view the information in the trace.
 - b. When asked for confirmation, enter: 3
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 TAS Parse expression](#) on page 450.

Using the TAS Parse expression

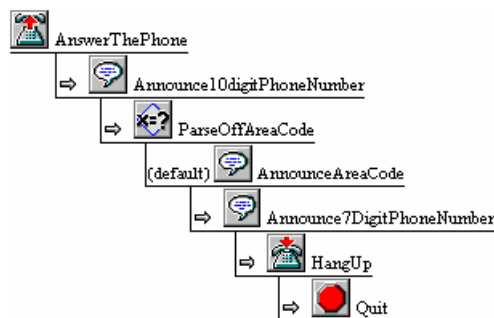
In this topic, you create an application, `pars`, that shows how the Parse expression alters the source string. The application contains a custom variable named `[PhoneNumber]` that has the default the value of 303-555-1212. The Parse expression is used to strip off the area code.

The `pars` application does the following:

- Answers the telephone.
- 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 `pars` application:

1. Create an application, naming it `pars`.
2. Create the following required custom variables.
 - `[PhoneNumber]`. Character variable, with the Spoken As field set to AC, a maximum length of 12 and a default value of 303-555-1212
 - `[AreaCode]`. Character variable, with the Spoken As field set to AC, and a maximum length of 3
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.

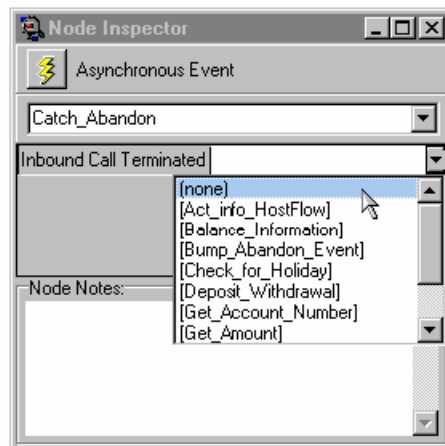
To continue, see [Using asynchronous events in TAS applications](#) on page 451.

Using asynchronous events in TAS applications

The Asynchronous Event node puts the system in a state where, upon detecting a hang up, it drops the call, executes a specified call flow, and terminates processing of the call. Terminating call processing is particularly useful for applications that connect to a host system. The Asynchronous Event node can execute a call flow that clears the host screen, resets the host to the transaction base screen, and terminates the script. Then the host system is ready for the next caller. In this tour, you use the Asynchronous Event node to capture information about calls that end in inbound hang ups.

Using and specifying the Asynchronous Event node is simple:

- First, create the call flow to be executed in the event of a hang up. The Node Inspector for the Asynchronous Event node contains the field Inbound Call Terminated, as shown in the following figure.



- From the drop-down menu, select the call flow to execute. The drop-down menu includes the possibility [none], which resets the system so that it does not execute a call flow when it detects a hang up.

This underlines an important aspect of the Asynchronous Event node. From the time it is encountered in an application, this node sets the system to execute a specified call flow until

such time as the application encounters another Asynchronous Event node to reset the system to a different call flow (or [none]).

Because of this, you often see call flows in which the Asynchronous Event node is the first node in the [Main] call flow. The node does, in fact, execute at that point, setting the system. But it does not cause the system to execute the specified call flow unless the system encounters an asynchronous event (a caller hang up).

To continue, see [Defining TAS asynchronous events](#) on page 452.

Related topics:

[Defining TAS asynchronous events](#) on page 452

[Resetting the system](#) on page 455

Defining TAS asynchronous events

In this topic, you create an 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 executes a call flow that adds a record, that contains the current date and identification number (if accepted), to a database table of calls.

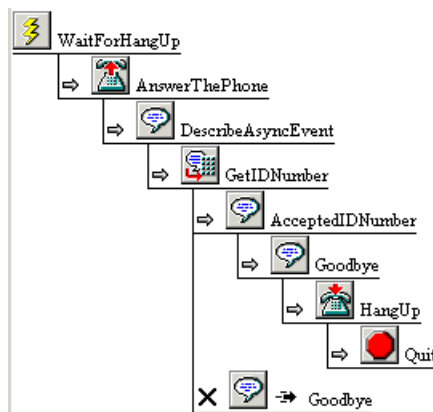
The application you create, called async:

- Sets the system to look for a hang up.
- Answers the telephone.
- 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 adds a record containing information about the call to a table.

To create the async application:

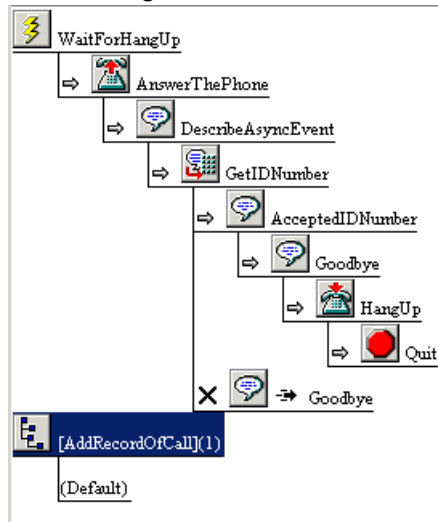
1. Create a new TAS application, naming it async.
2. Create the table CALLS with two fields:
 - CALLED is a date field with the format yyyyymmdd.
 - IDIFRECEIVED is a character field with a length of 9.
3. Drag an Asynchronous Event  node to the design pad and name it WaitForHangUp.
4. Add an Answer Call node and name it AnswerThePhone.
5. Add an Announcement node and name it DescribeAsyncEvent. Edit it as follows:

- a. Create the Initial Prompt to say:
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.
 - b. Set the Allow Interrupt field to False.
6. Create a new variable named CIID and set its attributes as follows:
Type: character
Spoken As: AC
Length: 9
7. 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 Formatted Response In: [CIID]
8. 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.
9. Add an Announcement node and name it Goodbye. Have it say something appropriate for ending the call.
10. Add a Disconnect Call node and name it HangUp.
11. Add a Return node and name it Quit.
12. Jump to the Goodbye node from the fail branch of GetIDNumber.
When done, the call flow should look like the following figure:

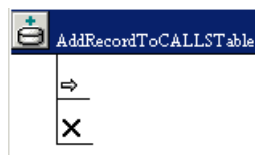


With the exception of the Asynchronous Event node named WaitForHangUp, the [Main] call flow is complete.

13. Drag a Call Flow node to the bottom of the [Main] call flow and name it AddRecordOfCall, as shown in the following figure. Note that it does not connect to the existing call flow.



14. Edit the [AddRecordOfCall] call flow.
15. Drag an Add Record node to the design pad and name it AddARecordToCALLSTable.
16. Edit the Add Record node as follows:
 - Identify CALLS as the Table.
 - Assign [Date] to CALLDATE and [CIID] to IDIFRECEIVED.
 When you are finished, your [AddRecordOfCall] call flow should look like the following figure:



You do not need to add anything to either the pass or fail branch of the AddARecordToCALLSTable node.

17. Return to the [Main] call flow.
18. Edit the Asynchronous Event node to select the [AddRecordOfCall] call flow for Inbound Call Terminated.
19. Save, verify, and simulate the application.
20. If you have a development system, generate code, transfer and install the application.
21. 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.
- Verify that a record is added to the database table in all the cases.

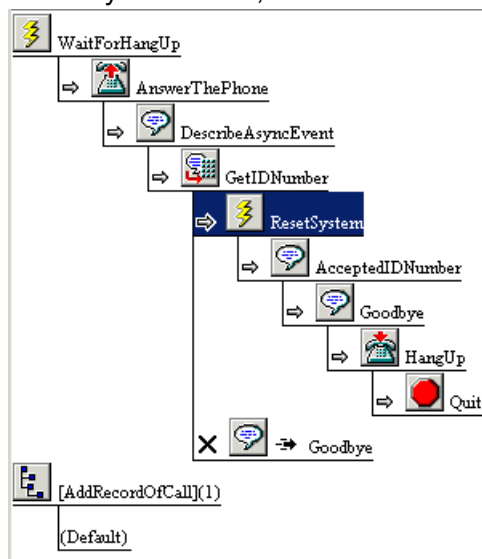
To continue, see [Resetting the system](#) on page 455.

Resetting the system

As it is currently written, the application async adds a record to the table CALLS whenever a caller hangs up. Frequently, you want to reset the system so that it no longer executes a call flow when it detects either a hang up or the normal end of a call. To reset the system, add an Asynchronous Event node at the point in the application where you want it to reset and specify [none] as the call flow to execute in the Inbound Call Terminated field.

To reset the system:

1. In the [Main] call flow, disconnect the AcceptedIDNumber node.
2. Drag an Asynchronous Event node to the open branch and name it ResetSystem, leaving the Inbound Call Terminated field blank. This resets the system so that it no longer executes the [AddRecordOfCall] call flow.
3. Reconnect the AcceptedIDNumber node to the open branch under the ResetSystem node, as shown in the following figure.



4. Modify the AcceptedIDNumber node so that it says:
The system has been reset so no record will be written to the calls received table if you hang up now.
5. Save, verify, and simulate the application.
6. If you have a development system, generate, transfer, install, and test the application.

To continue, see [Using external functions in TAS applications](#) on page 456.

Using external functions in TAS applications

An external function is code that resides on the Avaya IR or CONVERSANT system, outside of an IVR Designer application. Use external functions to perform application tasks that are not available directly in IVR Designer. The system includes several standard external functions, and you can create custom external functions.

IVR Designer uses both standard Avaya IR or CONVERSANT external functions and custom external functions. To use external functions, you include an External Function node in your application to act as the interface with the external function on the system.

You can create external functions in two ways:

- With the IVR Designer External Function Source Editor
- Directly on the Avaya IR or CONVERSANT system

In this tour, you learn about a variety of standard external functions. You also create a custom external function.

To continue, see [Using the Make_Call external function](#) on page 457.

Related topics:

[Using the Make_Call external function](#) on page 457

[Using the Make_Call and Sleep external functions](#) on page 460

[Using the Datetime_u and Getday external functions](#) on page 461

[Using the Mon_year external function](#) on page 464

[Using the Execute external function](#) on page 465

[Using the Execute and Getarg external functions](#) on page 468

[Using the FAX external functions](#) on page 471

[Viewing external functions](#) on page 474

[Using Inline Code](#) on page 475

[Creating custom external functions](#) on page 476

Using the Make_Call external function

Use the Make_Call external function to initiate outbound calls. In essence, Make_Call is the outbound equivalent of Answer Call.

Specifying a Make_Call node is similar to specifying a Transfer Call node. You enter the following information in the Node Inspector:

- Phone Number. Number to be called.
- Outcome of Call. Variable where the return code goes.

Following are the possible Make_Call return codes :

- Voice Detected : V
- Answering Machine: M
- Answer : A
- Busy : B
- No Answer : N
- Reorder : F
- Modem : T
- SIT : S

As with call transfers, a status value is returned to your application to indicate the disposition of the call.

- Type. Blind, Intelligent, or Full Call Classification Analysis (CCA)
- Max Rings. Maximum number of rings for Intelligent or Full CCA.

Applications that begin with a Make_Call node instead of an Answer Call node are often scheduled to run at a particular time or be executed by another application. To test an application beginning with a Make_Call node, you can use the soft_srz command, which is described in the soft_srz (command) description.

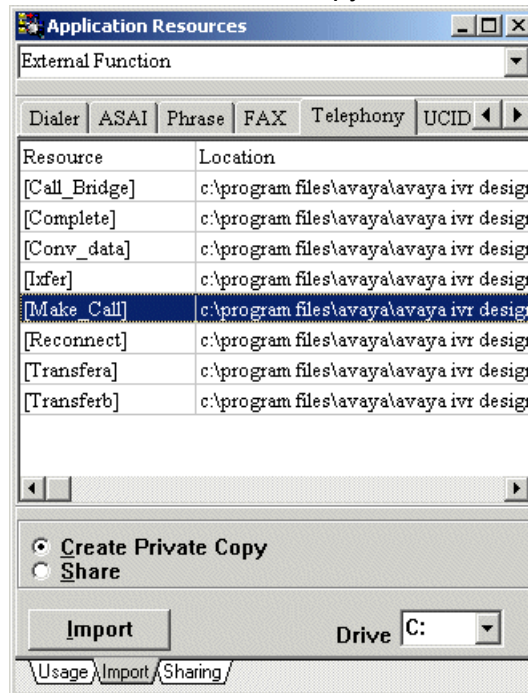
In this topic, you create and test a simple demonstration application that contains a Make_Call external function. This application:

- Initiates an outbound call to the number specified in the Make_Call external function node upon execution of the Avaya IR or CONVERSANT soft_srz command.
- Announces: "I will only talk to you if you say hello."
- Disconnects and quits.

To create the makecall application:

1. Create a new TAS application named makecall.
2. If it is not already visible, select View > External Function Palette to display the External Function Palette.
At this point, the External Function Palette should be empty.

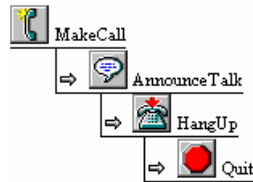
3. Use the Application Resources tool to make the Make_Call external function available to the application:
 - a. Click the Application Resources icon.
 - b. Select the Import tab.
 - c. Select the Telephony tab.
Use the arrows, if necessary, to scroll to the Telephony tab.
 - d. Select Make_Call.
 - e. Select Create Private Copy, as shown in the following figure.



- f. Click Import.
- g. Close the window.
The Make_Call external function node is now displayed on the External Function node palette, as shown in the following figure.



4. Drag a Make_Call  node to the design pad and name it MakeCall.
5. Add an Announcement node and name it AnnounceTalk.
 - Edit the initial prompt to say:
I will talk only if you say hello.
 - Set the Allow Interrupt field to False.
6. Add a Disconnect Call node and name it HangUp.
7. Add a Return node and name it Quit.
The call flow should now look like the following figure:



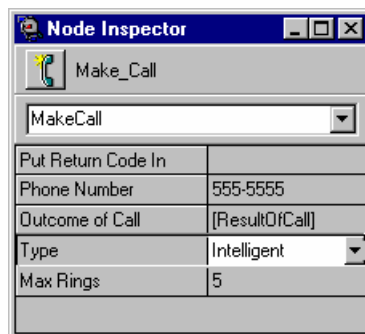
8. Add a new character variable and name it ResultOfCall. This variable holds a single character, but set it at a length of 3 to account for negative values.
This variable is needed because, when the Make_Call node is edited, there must be a variable in which to place the Outcome of Call from the Make_Call external function.

9. Edit the MakeCall node as follows:

- Enter the telephone number for the Make_Call external function to call.
If you can test the application on a development system, use your telephone number.

The telephone number could also be stored in a character variable and assigned using the drop-down menu.

- For Outcome of Call, select [ResultOfCall].
- In Type, enter Intelligent .
You must type this into the field.
- For the Max Rings, type 5 .
When you are finished, the Node Inspector for the Make_Call external function should look like the following figure:



10. Save and verify the application.
11. Simulate and test the application.
12. If you have a development system, generate code and transfer and install the application.
13. Test the application
 - a. Connect to the Avaya IR or CONVERSANT system and log on.
 - b. To start the makecall application, at the system prompt enter the following (where ## is the channel to which the makecall application is assigned):
soft_srz ## makecall

The application should call you.

- c. Answer the telephone and you should hear:
"I will talk only if you say hello."
- d. Trace the application with the following command (where ## is the channel to which the makecall application is assigned):
`trace ## | tee /tmp/makecall-##.out`

For more information on tracing applications, see [Tracing a TAS application](#) on page 521.

Note that tracing an application that is started with a `soft_srz` command requires two windows to view the trace: one in which to execute the trace command (initiate first) and the other in which to execute the `soft_srz` command (after the trace command has been initiated). If you are communicating by telnet, these are telnet windows.

- e. Trace the application with the following command (where ## is the channel to which the makecall application is assigned):
`trace ## area SE | tee /tmp/makecall-##-2.out`

Again, notice the value of A after the `Make_Call` external function completes.

- f. Test the application while running a trace.
Do not answer the call and note the Outcome of Call code returned. If possible, put your telephone in a Busy state and note the Outcome of Call code returned.

To continue, see [Using the Make_Call and Sleep external functions](#) on page 460.

Using the Make_Call and Sleep external functions

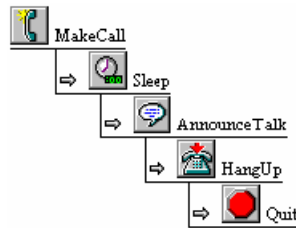
In this topic, you start with the makecall application from the previous topic and add a Sleep external function before the Announcement node. The Sleep external function causes the application to pause for a set period of time. In this case the application pauses for five seconds, so you always hear the Announcement message when you answer the call.

To modify the makecall application:

1. Open the makecall application and save it as makecll2.
2. Disconnect the AnnounceTalk node from the flow.
3. From the Control tab of the External Functions screen in Application Resources, import a private copy of the Sleep external function.
For more information on importing external functions, see [Using the Make_Call external function](#) on page 457.
4. Add a Sleep  node after the MakeCall node and name it Sleep.
5. Edit the Sleep node so that the application sleeps for five seconds.

Use the pop-up hint to find where to enter the 5.

6. Connect the AnnounceTalk node to the Sleep node.
Your call flow should now look like the following figure:



7. Save and verify the application.
8. Simulate the application.
9. If you have a development system, generate code and transfer and install the application.
10. Test the application
 - a. Connect to the Avaya IR system and log in.
 - b. Initiate the application at the system prompt, using the following command (where ## is the channel number assigned to the application):
soft_szt ## makecll2
 - c. Test to make sure that the application returns the proper values for call outcomes by calling a busy line, not answering, picking up but not speaking, and answering the call normally.
 - d. Trace the application.
For more information on tracing applications, see [Tracing a TAS application](#) on page 521.

To continue, see [Using the Datetime_u and Getday external functions](#) on page 461.

Using the Datetime_u and Getday external functions

The Datetime_u external function converts date and time fields to a system date and time format. The Getday external function converts system time to the day of the week.

IVR Designer applications execute on various platforms. Applications must be able to work with time as understood by the system. IVR Designer has the [Date] and [Time] variables available to it by default, but the values in these variables are not usable by the Getday external function.

To provide the Getday external function with the values it needs, you use the Datetime_u external function to convert the values in [Date] and [Time] to system time, which is the number of seconds since January 1, 1970.

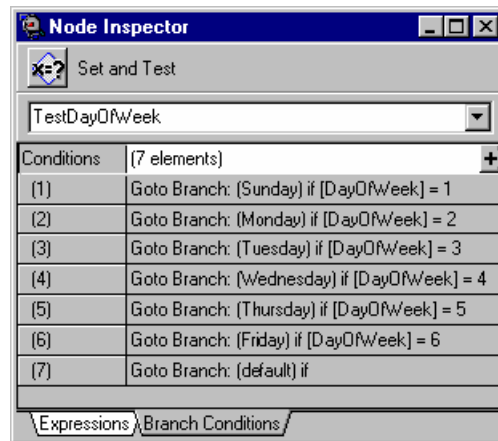
The application you develop in this topic, whatday, announces the day of the week based on the value of the [DayOfWeek] variable, as determined by the Datetime_u and Getday external functions.

To create the whatday application:

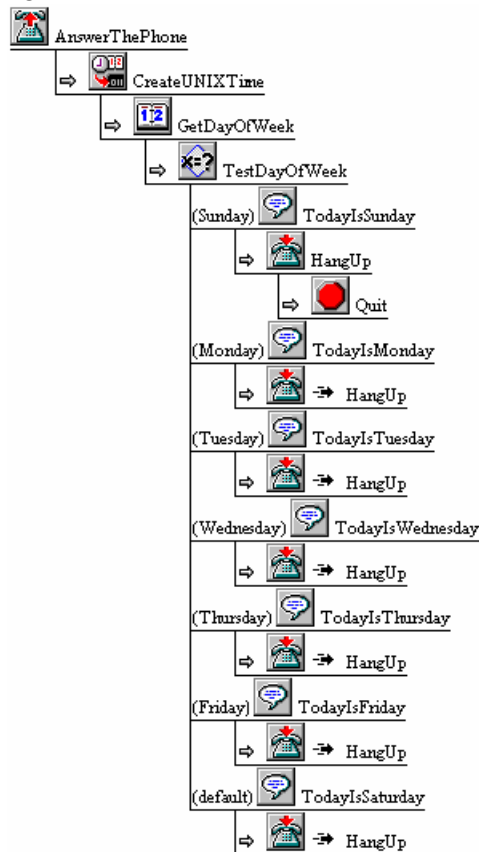
1. Create a new TAS application named whatday.
2. Import private copies of the Datetime_u  and Getday  external functions. For more information on importing external functions, see [Using the Make_Call external function](#) on page 457.
3. Drag an Answer Call node to the design pad and name it AnswerThePhone.
4. Add a Datetime_u node and name it CreateUNIXTime.
5. Add a Getday node and name it GetDayOfWeek.
6. Create a number-type variable and name it NumberOfSecondsSince1970. Both the Datetime_u and Getday external functions use this variable.
7. Create another number-type variable and name it DayOfWeek. This variable is used by the Getday external function only.
8. Edit the Input As fields for the [Date] and [Time] variables as follows: Change [Date] to yyyyymmdd format and [Time] to hhmmss format.
9. Edit the CreateUNIXTime node as follows:
 - Put Return Code In: [NumberOfSecondsSince1970]
 - Date: [Date]
 - Time: [Time]
10. Edit the GetDayOfWeek node as follows:
 - Put Return Code In: [DayOfWeek]
 - Clock: [NumberOfSecondsSince1970]

At this point the application determines what day of the week it is. If [DayOfWeek] is 1, the day is Sunday. If it is 2, the day is Monday, and so on.
11. Add a Set and Test node and name it TestDayOfWeek.
12. Insert six new branches under the TestDayOfWeek node.
 - Name the Branches: Sunday, Monday, Tuesday, Wednesday, Thursday, and Friday.
 - For Saturday ([DayOfWeek] = 7), use the (default) branch.
13. Edit the TestDayOfWeek node, as shown in the following figure:
 - Click the Branch Conditions tab and add six new elements. (No expressions are required.)
 - Set element (1) to go to the (Sunday) branch if the [DayOfWeek] variable = 1.

- Similarly, set elements (2) through (6) to go to Monday through Friday, respectively.
- Remember, if the [DayOfWeek] variable = 7 (Saturday), the default branch is taken.



14. Using Announcement nodes, a Disconnect Call node, and a Return node, complete the branches associated with the TestDayOfWeek node, as shown in the following figure:



15. Edit the Announcement nodes and the [PhraseTable] to announce the days using Text-to-Speech.

16. Save and verify the application.
17. If you have a development system, generate code and transfer and install the application.
18. Test the application.
The application should say the current day of the week.

To continue, see [Using the Mon_year external function](#) on page 464.

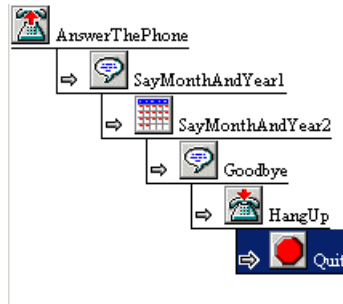
Using the Mon_year external function

You can use the Mon_year external function to announce the month and year to callers. Combined with the Datetime_u and Getday external functions, it provides a complete announcement of the date.

Unlike the Datetime_u and Getday external functions, the Mon_year external function does the announcement itself. You do not have to add an Announcement node that says the value in a variable. However, the Mon_year external function announces only the month and (optionally) the year. If you want any surrounding text to put the month and year in context, you must place Announcement nodes before and after the Mon_year external function node. You have the option of speaking the year as either a two-digit or four-digit number.

To create a simple application, monyear, that says the month and the four-digit year:

1. Create a new TAS application named monyear.
2. Import a private copy of the Mon_year external function.
For more information on importing external functions, see [Using the Make_Call external function](#) on page 457.
3. Drag an Answer Call node to the design pad and name it AnswerThePhone.
4. Add an Announcement node and name it SayMonthAndYear1, creating an initial prompt that says:
Thank you for calling. The month and year are: {pause}
5. Change the Input As attribute for the [Date] system variable to yyyyymmdd.
6. Add a Mon_year  node to the call flow, name it SayMonthAndYear2, and edit it as follows:
 - Date: [Date]
 - Format: 2
7. Add another Announcement node, name it Goodbye, and create an initial prompt that says:
Goodbye.
8. Add a Disconnect Call and a Return node to complete the application.
When finished, the 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.
11. Test the application.
The application should announce the correct month and year.

To continue, see [Using the Execute external function](#) on page 465.

Using the Execute external function

The Execute external function calls and executes any kind of Avaya IR or CONVERSANT script on the current channel. The script can be one written in:

- IVR Designer
- Another non-IVR Designer application
- TAS code

You can use the Execute external function to designate arguments to be passed to the script to be executed. You use the Getarg external function in conjunction with the Execute external function to pass arguments to a script. These arguments must be character-type variables. [Using the Execute and Getarg external functions](#) on page 468 explains how to use the two external functions in conjunction with each other.

An alternative to Execute and Getarg

IVR Designer includes another means of calling and executing scripts. You can use the Save and Exec node and the Load Variables node to call another IVR Designer application script and pass arguments to it. However, the Save and Exec and the Load Variables nodes work

only with IVR Designer applications. [Passing data between TAS applications](#) on page 480 explains how to use these nodes.

In this topic, you create an application, exec, that uses the Execute external function to pass control to one of two existing applications: ivrapp1 or ivrapp2.

The exec application does the following:

- Answers the telephone.
- Prompts the caller to press 1 to execute application number 1, or to press 2 to execute application number 2.
- Takes action based on caller response:
 - If the caller presses 1, the application passes control to the ivrapp1 application, which starts executing.
 - If the caller presses 2, the application passes control to the ivrapp2 application, which starts executing.

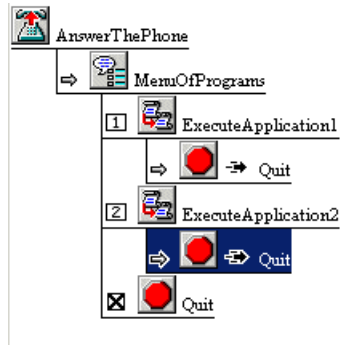
Both of the secondary applications, ivrapp1 and ivrapp2, do the following:

- Announce that exec passed control to the new TAS application (identifying which application has been called).
- Disconnect and quit.

To create the exec application:

1. Create a new TAS application named exec.
2. Drag an Answer Call node onto the design pad, and name it AnswerThePhone.
3. Add a Menu node, and name it MenuOfPrograms.
4. Add a Return node to the fail branch, and name it Quit.
5. Delete the [3] branch.
6. Create an initial prompt for the MenuOfPrograms Menu node that says:
To transfer to I V R application one, press one now. To transfer to I V R application two, press two now.
7. Import a private copy of the Execute external function.
For more information on importing external functions see [Using the Make_Call external function](#) on page 457.
8. Add an Execute  node to branch [1], and name it ExecuteApplication1.
9. Set the following ExecuteApplication1 node parameters:
 - Appl. Name: ivrapp1
Alternately, you could place the character string ivrapp1 in a character variable, and the variable would then be available using the drop-down menu.
 - Write CDR Now?: no
10. Add an Execute node to branch [2], and name it ExecuteApplication2.
11. Set the following ExecuteApplication2 node parameters:

- Appl. Name: ivrapp2
 - Write CDR Now?: no
12. Make the empty branches of the ExecuteApplication1 and ExecuteApplication2 nodes jump to the Quit node.
The finished application call flow should look like the following figure:

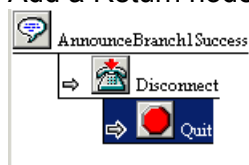


13. Save and verify the application.
14. If you have a development system, generate code and transfer and install the application.

Create the ivrapp1 and ivrapp2 applications

To create each of the secondary (ivrapp1 and ivrapp2) applications:

1. Create a new TAS application named ivrapp1 or ivrapp2.
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 exec application has passed control to I V R 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.

Test the Applications

1. Connect to the Avaya IR or CONVERSANT system and log in.
2. Enter the following command:
display services

The applications exec, ivrapp1, and ivrapp2 should be included in the list of installed applications.

3. Assign the exec application to a channel or channels.
4. Start a trace with the following command (where ## is the voice channel number to which exec is assigned):
trace ## area SE | tee /tmp/exec- ## .1out
 - a. Call the assigned number.
 - b. Select 1 when prompted and make sure you hear the announcement from ivrapp1.
Remember: To end the trace, press the Delete key.
 - c. Notice where and how the control is passed to the ivrapp1 application.
5. Start a trace with the following command (where ## is the voice channel number to which exec is assigned):
trace ## area SE | tee /tmp/exec- ## .2out
 - a. Select 2 when prompted and make sure you hear the announcement from ivrapp2.
 - b. Notice that control is passed to the ivrapp2 application.

To continue, see [Using the Execute and Getarg external functions](#) on page 468.

Using the Execute and Getarg external functions

The Getarg external function is used in conjunction with the Execute external function to get arguments that are passed to a script. These arguments must be character-type variables.

In this topic, you use the Execute and Getarg external functions to pass strings (character values contained in variables) between applications. You create three IVR Designer applications: exec2, ivrapp1, and ivrappid.

The primary application, exec2, uses the Execute external function node to pass control to one of two other applications. It does the following:

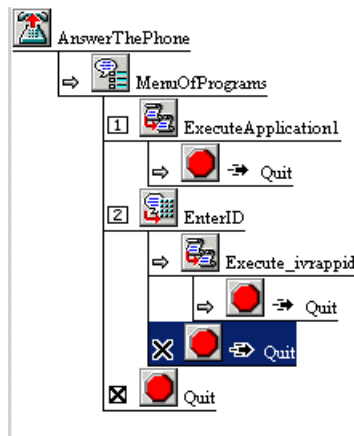
- Answers the telephone.
- Prompts the caller to press 1 or to press 2.
- Takes action based on caller response:
 - If the caller presses 1, the application passes control to ivrapp1, which starts executing, announces that ivrapp1 has control, and quits.
 - If the caller presses 2, the application prompts the caller for a 9-digit identification (ID) number, and passes control and the ID number to ivrappid, which starts executing, executes a Getarg external function that accepts the ID number, announces the ID number, and quits.

To create the exec2 application:

1. Open the exec application from the previous topic and save it as exec2.

Note that the ivrapp1 application and branch already exist and do what you want, so there is no need to modify them.

2. Delete the ExecuteApplication2 node.
3. 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 I D number.
 - On the Response tab, set the minimum and maximum lengths to 9.
4. Add an Execute node to the pass branch of the EnterID node. Name it Execute_ivrappid and edit it as follows:
 - Appl. Name: ivrappid
 - Write CDR Now?: no
 - Argument 1: [CollectedDigits]
5. Complete the exec2 application by making the pass branch of the Execute_ivrappid node and the fail branch of the EnterID node jump to the Quit node. When finished, the exec2 call flow should look like the following figure:



6. Save and verify the application.
7. If you have a development system, generate code and transfer and install the application.

Create and install the ivrappid application

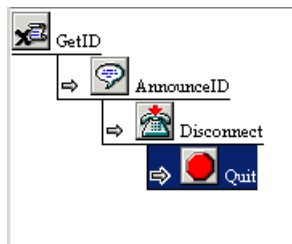
For this topic, you can reuse the ivrapp1 application that you created for branch [1] of the Execute topic. You do, however, create a new application for branch [2].

To create the new application:

1. Create a new TAS application, named ivrappid.
2. Import a private copy of the Getarg external function.
It should be displayed on the External Functions Palette and in the External Functions Manager.

For more information on importing external functions, see [Using the Make_Call external function](#) on page 457.

3. Create a new variable named ReceivedID with the following attributes:
 - Type: character
 - Maximum length: 9
 - Spoken As: AC
4. Drag a Getarg  node to the design pad. Name it GetID and edit it as follows:
 - Put Return Code In: leave empty
 - Arg. Number: 1 (Enter as a literal value.)
This is because the Execute external function (in the exec2 application), which is going to pass the argument to this external function uses Argument 1, whose number is 1, to pass the value of the [CollectedDigits] variable. Using the drop-down menu for this field, notice that all available variables are number variables.
 - Address: [ReceivedID]
This is the variable in which you store the value of the argument that is being passed. Notice that all the variables available for use in this field are character variables.
 - Length: 9
5. Add an Announcement node after the GetID node, name it AnnounceID, and set the initial prompt to:
The I D number passed to this application is, [ReceivedID]
6. Add a Disconnect Call node to the call flow, and name it Disconnect.
7. Add a Return node, and name it Quit.
When you are finished, your call flow should look like the following figure:



8. Save and verify the application.
9. Simulate the application and when prompted, enter: 123456789
10. If you have a development system, generate code and transfer and install the application.
Do not assign the application to a voice channel.

Test the Applications

1. Connect to the Avaya IR or CONVERSANT system and log in.
2. Enter the command: display services and verify that exec2 and ivrappid are included in the list of installed applications.

3. Assign the exec application to a channel or channels.
4. Start a trace with the following command (where ## is the channel number):
`trace ## area SE | tee /tmp/exec2- ## .1out`
5. When prompted, press 1. Make sure you hear the announcement from ivrapp1.
 Remember: To end the trace, press Delete.
 Notice where control is passed to the ivrapp1 application.
6. Start a trace with the following command (where ## is the channel number):
`trace ## area SE | tee /tmp/exec2- ## .2out`
7. When prompted, press 2. Make sure you hear the announcement from the ivrappid application.
 Notice again where control is passed to the ivrappid application.

To continue, see [Using the FAX external functions](#) on page 471.

Using the FAX external functions

The FAX offer is a suite of separately installable software packages from NGFAX software.

IVR Designer provides several external functions to use in applications that provide FAX functionality. These external functions include the following:

- FAX_CNG
- FAX_CovrPage
- FAX_Get
- FAX_Queue
- FAX_Send
- Concat5
- Exec_UNIX



Note:

The FAX_CovrPage and FAX_CNG external functions are currently supported on only CONVERSANT V9 and earlier systems.

To create the faxdemo application:

1. Create a new TAS application and name it faxdemo.
2. Use the Application Resources tool to import private copies of the standard FAX external functions listed above.
3. Drag a FAX_CNG  node to the design pad, name it FAXTone, and set Mode to listen.



Note:

Skip this step if you are developing an application for a Avaya IR system.

FAX CHG is the tone emitted by a transmitting fax machine after Start has been pressed. The FAX_CNG node turns FAX CHG tone detection on and off.

The FAX_CNG node is not required, but it is a good practice to include it in FAX applications. Including the FAX_CNG node ensures that the FAX is not repeatedly sent to a number that is not a receiving FAX device.

4. Add a FAX_Get  node and name it GetAFAX. Set FAX_File to the full pathname for a location on the Avaya IR or CONVERSANT system where you want to store incoming FAXs.

The FAX_Get external function directs the system to receive a fax and to place it into a designated directory and file. This node is used to wait for and accept FAXs on the current call and channel.

5. Add a FAX_CovrPage  node and name it FAXCoverPage. Specify files to be used for the background and foreground of the cover page.



Note:

Skip this step if you do not have cover page files or if you are developing an application for a Avaya IR system.

The FAX_CovrPage external function directs the system to merge two files into a single FAX file. This external function is especially useful for joining graphic images with text information. By doing this, you can create customized cover pages with letterhead or graphics and logos combined with the text address of the intended recipient.

You can use the Exec_UNIX external function to create the text portion of the cover sheet. Once created, the cover sheet is automatically queued and delivered with the next FAX_Send node.

6. Add a FAX_Queue  node and name it QueueTheFAX. Specify the name of the file you want queued for FAX transmission.
The FAX_Queue external function places a fax in order for transmission. The FAX being queued must exist at the requested time of transmission. This external function should be used for each FAX requested by the caller. If the Return Code returns a negative value, it indicates an error. Most likely, the file that was queued could not be found.

You may specify the full pathname of the file, or a character variable that includes the name of the file to be queued. If a FAX file that exists in /usr/faxdb/FR/WORKFAX is specified, the file name is sufficient. The file must be either in ASCII or TIFF Class F format. Any FAXs loaded on to the Avaya IR or CONVERSANT system with the Loading and Printing window are converted to TIFF Class F format.

7. Add a FAX_Send  node and name it SendTheFAX. Edit it as follows:
The FAX_Send external function directs the system to transmit the specified graphic image or text file to a designated fax telephone number. It can be used to schedule a fax transmission for a specified time in the future.

- Put Return Code In. Contains the result status of the FAX_Send node.
A negative value in this field indicates that a problem was encountered, and the caller did not receive the requested FAX. You should check the return

value and inform the caller of this fact. A positive return value is the numeric portion of the job ID associated with the FAX transmission.

You can view the job ID in the FAX Transmission Control window on the Avaya IR or CONVERSANT system while the FAX is in the transmission queue or in the faxlog command output after the transmission has completed.

- FAX Delivery Number. The telephone number to which the FAX is to be sent or the word CURRENT for the current call.
- FAX Delivery Time. If the FAX queued previously with a FAX_Queue node is to be sent immediately, leave this field blank. If the FAX is to be sent at a later time, enter the time in the form: hhmm using a 24-hour clock.
- FAX Retry Interval. The interval (in minutes) between FAX retries for a FAX whose delivery has been attempted but has failed. This field does not apply to current call deliveries.
- FAX Retry Count. The number of times the FAX delivery should be retried in the case of a failure. This field does not apply to current call deliveries.
- TSI String. The user-definable Transmitting Subscriber Identification (TSI) for Group 3 FAXs. The value in this field is printed at the top of each page with the page number, date sent, and so forth.
Use this field to specify the sender's FAX telephone number, company name, or other informational string. This TSI string is also displayed on the receiving FAX machine's LED.
- Equipment Group. The equipment group number used to send. You can dedicate selected channels to be used by the FAX_Send external function. If this field is blank, the system chooses any FAX capable channel available to send the FAX.

8. Add a Concat5  node and name it MakeString. Edit it as follows:
 - Destination: The field for the result
 - String1 through String5: The strings to be concatenated
 - Max_Result: The maximum length of the resultant string
If the Max Result is set to 0 or left blank, a maximum of 255 characters is assumed. The Max Result and the Length set in the Globals Manager should match. Otherwise the created string could be truncated, potentially resulting in an error.
9. Add an Exec_UNIX  node and name it ExecuteUNIXCommand. Edit it as follows:

- Command String. The field for the UNIX command
- Return String. The variable for the command reply

Viewing Loaded Faxes

1. Connect to the Avaya IR or CONVERSANT system and log in.
2. Enter the command: `cvis_menu` and select Application Package Administration > Script Builder FAX Actions.
The system displays the Script Builder FAX Actions window, which contains two options:
 - FAX Loading and Printing
 - FAX Transmission Control
3. Select FAX Loading and Printing.
This display shows FAXs that are loaded and available for FAXing by any IVR Designer application. In the FAX_Queue node, there is a field named FAX_File. FAX_File can be any ASCII or TIFF file on the Avaya IR or CONVERSANT system. If, for example, you wanted to send a file that is already loaded, named fax1, you would enter fax1 in the FAX_File field.

The information for some FAXs may be created during a call. A caller may request information that needs to be retrieved from a database table. A FAX created for a caller in this manner is created dynamically. The FAXs loaded through the Loading and Printing function are static.

To continue, see [Viewing external functions](#) on page 474.

Viewing external functions

In this topic, you view the location of external function files on the Avaya IR or CONVERSANT system. Within the IVR Designer software, all available external functions can be added to any application. However, some external functions are available only when an optional software package is installed on the system. Viewing the functions tells what is available for creating applications.

To view the location of external function files:

1. Connect to the Avaya IR or CONVERSANT system and log in.
2. Change the directory to: `/vs/bin/ag/`
3. At the system prompt, enter the command: `file * | grep dir`
You should find lib and eascripts listed as subdirectories.
4. To change the directory, enter the command: `cd lib`
This (lib) is the subdirectory that contains the standard external functions available for use by IVR Designer.
5. To view the contents of the subdirectory, enter the command: `ls`

6. To change to the other directory, enter the command: `cd ../../eascripts`
7. To view the contents of the subdirectory, enter the command: `ls`
This directory also contains the external functions that are included when specific option software packages are installed on the system.

When developing IVR Designer applications, make sure that the external function you want to use is also available on the Avaya IR or CONVERSANT system where your IVR Designer application runs.

To continue, see [Using Inline Code](#) on page 475.

Using Inline Code

With the Inline Code node, you can insert TAS instructions directly into an IVR Designer application. When the code is generated and the application is transferred, the TAS code is inserted at the point where the node is placed in the application.

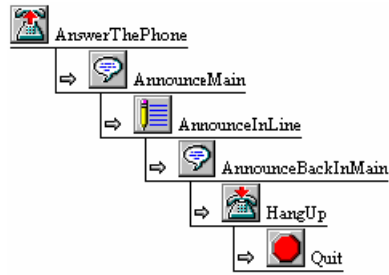
The application you create in this topic includes an Inline Code node that includes a single TAS instruction to say a Text-to-Speech sentence:

1. Create a new TAS application named `incode`.
2. Drag an Answer Call node to the design pad and name it `AnswerThePhone`.
3. Add an Announcement node and name it `AnnounceMain`. Create an initial prompt that says:
Hello. You are in the Main call flow, about to execute an in line code instruction. In Line code nodes contain TAS instructions.
4. Add an Inline Code  node and name it `AnnounceInLine`. Edit it as follows:
 - a. Click ... in the Text field of the Node Inspector.
This opens a window where you can enter valid TAS commands for execution within an IVR Designer application.
 - b. In the Text of Code field, enter the following:
`say (im."You are now in the In Line Code TAS instruction.")`
 - c. Click OK.

Note:

TAS instructions and their syntax are beyond the scope of this course material. The `say` instruction is generated for each Text-to-Speech specified Prompt Text.

5. Add an Announcement node and name it `AnnounceBackInMain`. Create an initial prompt that says:
You are now back in the Main call flow.
6. Add a Disconnect Call node and a Return node to complete the application.
When finished, the call flow should look like the following figure:



7. Save and verify the application.
8. If you have a development system, generate code and transfer and install the application.
9. Connect to the Avaya IR or CONVERSANT system and log in.
10. Change the directory to: /lt/trans/sc/inlcode
11. Use cat to view the inlcode.t file.
12. Find the say instruction that is identical to the instruction included in the Inline Code node.
13. Test the application.

To continue, see [Creating custom external functions](#) on page 476.

Creating custom external functions

With IVR Designer, you can create your own custom external functions, including the TAS code required to install and run it on the Avaya IR or CONVERSANT system.

To create a custom external function, you must first create the external function definition and assign it an icon. This places the custom external function node on the External Function Palette, making it available to your application.

This node, however, provides only the interface for the external function code itself (the same as any other external function node). You must create the TAS code required by the Avaya IR system to actually use the external function. This can be done using IVR Designer's External Function Source Editor.

In this topic, you create an application that uses a custom external function. You also create the custom external function code.

Creating the External Function Definition

To create the custom external function definition:

1. Create a new TAS application, naming it extfunc.
2. Open the External Functions Manager.
3. Right-click the External Functions Manager and select New.
4. Name the new external function myexternalfunction.

Use all lowercase letters when naming external functions to avoid problems with case-sensitive UNIX systems.

The system displays the External Function Editor, as shown in the following figure.

Function Parameters

Function Name: myexternalfunction ☐ System Function

Description:

Source File Name: myexternalfunction.t

Function Arguments

Argument	Type	Direction	Help Message	Required
	character	To Func		True

Function Return Values

Return Code Description:

Branch	Operator	Value	Help Message
	=		

☐ Go to Fail branch when return code is less than zero.

Palette Appearance

Bitmap:  Group/Tab Name: Custom

OK Cancel

User-written external functions can specify various arguments and a Return code. For this demonstration, no arguments or Return codes are passed or received.

If the System Function check box in the upper-right corner of the External Function Editor is checked, the definition cannot be modified until or unless this check box is cleared. It does not actually make the external function a standard (or system) external function.

In the lower-left corner, locate the bitmap icon for external functions. This is a generic icon used for any custom external function unless another bitmap image is selected. For this demonstration, use the generic icon.

5. Create the external function definition by entering the following in the Description field:

This is my personal external function that can be shared. I am creating it for a class.

To make your custom external function available to other IVR Designer applications on your PC, you must share it out. Sharing it out on your PC, however, does not automatically make it available to other applications. You must specifically import it

into other applications. Also, you must copy the *.t file to the /vs/bin/ag/lib or /vs/bin/ag/eascripts directory.

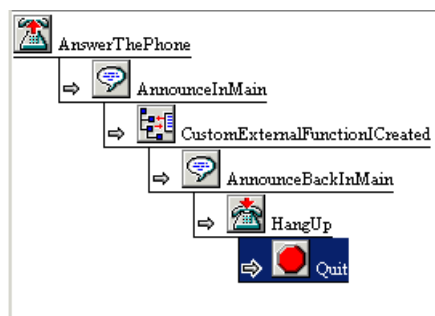
6. Right-click the External Functions Manager and select Sharing... When prompted, elect to save the application and then select: Shared. Click OK.
This makes the custom external function available to other IVR Designer applications and developers.

Notice that the new icon has been added to the External Function Palette. Pass your cursor over the node and notice that the pop-up hint shows this node to be myexternalfunction.

Creating the Call Flow

To create the application call flow:

1. Drag an Answer Call node to the design pad and name it AnswerThePhone.
2. Add an Announcement node, name it AnnounceInMain, and set the Initial Prompt to say:
You are now in the Main call flow.
3. Add a myexternalfunction node and name it CustomExternalFunctionICreated.
The only field available for editing is Put Return Code In. Since you are not checking the return code for anything, leave it blank. You may, however, want to use this field to track problems in the execution of custom external functions.
4. Add an Announcement node and name it AnnounceBackInMain, and set the Initial Prompt to say:
You are now back in Main. Thank you. Goodbye.
5. Complete the call with a Disconnect Call node and a Return node.
When finished the call flow should look like the following figure:

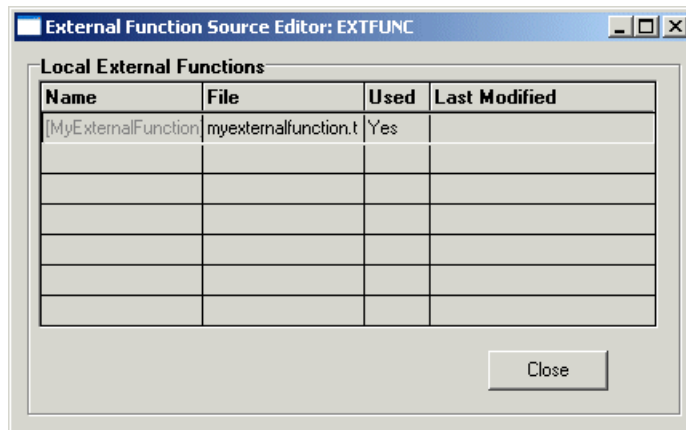


6. Save and verify the application.

Creating the External Function TAS Code

To create the TAS code to make the custom external function work:

1. Open the Code Generation tool window by clicking the Code Generation icon or by selecting Code Generation / Application Transfer from the Tools menu.
2. Click the External Function Source Code  icon.
The External Function Source Editor, shown in the following figure, is displayed.



- Right-click anywhere on the line that includes myexternalfunction and select Create File.

A notepad opens with the template code text for myexternalfunction. As with the Inline Code node example, you add a single line of TAS code so that you know that the custom external function is executed correctly.

- Edit the code and add an instruction similar to the one used for the [Using Inline Code](#) on page 475 example. Use the sample, shown in the following figure, to help you insert the code in the correct place. Insert the code that is shown in bold.

```

/*
*FUNCTION:
*   myexternalfunction
*DESCRIPTION:
*   This is my personal external function that can be shared.
*INPUT/OUTPUT:
*RETURN:
*   none
*SCCSID INFO:
*   EXTFILE PAS 21.1.2.1 of 16 Sep 1997
*   @(#)EXTFILE.PAS    21.1.2.1
*/

#ifdef MYEXTERNALFUNCTION_T
#define MYEXTERNALFUNCTION_T

DEFARG_COUNT(0)

L_myexternalfunction:
    trace    (im.0, im."@(#)EXTFILE.PAS    21.1.2.1")
    say      (im."You are now in the custom external function.")
    rts      0
#endif

```

- Save the changes to the file and exit Notepad. Note that the Last Modified entry for the file has been created.
- Click Close.
- If you have a development system, generate code and transfer and install the application.
- Connect to the Avaya IR system and log in.
- Change the directory to /lt/trans/sc/extfunc and list the files in the directory. Verify that the myexternalfunction.t file is listed.
- View the contents of this file, which should be identical to the one you just edited using the External Function Source Editor.

11. List the contents of /vs/trans.
Verify that the extfunc.T file is listed.
12. Test the application.
As part of the Text-to-Speech message, you should hear the message:
You are now in the custom external function.

To continue, see [Passing data between TAS applications](#) on page 480.

Passing data between TAS 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. If you want to call and use non-IVR Designer application scripts, you must use the Execute and Getarg external functions, covered in [Using the Execute and Getarg external functions](#) on page 468.

To continue, see [Using the Save and Exec node and Load Variables node](#) on page 480.

Related topics:

[Using the Save and Exec node and Load Variables node](#) on page 480

Using the Save and Exec node and Load Variables node

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, svexec and ldvar.

The primary application, svexec, does the following:

- Answers the telephone.
- Sets the value of a variable to be passed to the ldvar 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 ldvar application.

The secondary application, ldvar, does the following:

- Answers the telephone.
- Uses a Load Variables node to accept the variable values from the svexec application and places them in the new variables.
- Announces the values of the variables and quits.

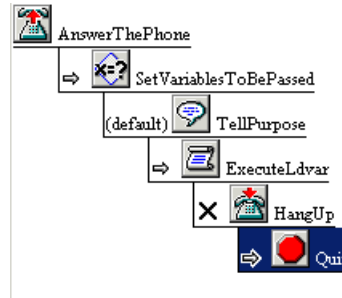
Create the svexec application

You create the svexec application to pass control and the values of the three variables to the ldvar 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 and length in both applications, or the values may be corrupted.

To create the svexec application:

1. Create a new TAS application and name it svexec.
2. Create the custom variables for this application:
 - [Variable1]. A character variable with a default value of Vasco da Gama
 - [Variable2]. A character variable with no default value (the value is set to 303-555-1212 using a Set and Test node)
 - [Variable3]. A number variable with a default value of 1497
3. Drag an Answer Call node onto the design pad and name it AnswerThePhone.
4. Add a Set and Test node and name it SetVariablesToBePassed. Use this node to assign a value of 303-555-1212 to the [Variable2] variable.
5. Add an Announcement node and name it AnnouncePurpose. Create an initial prompt to describe what this application is and what it does.
6. Add a Save and Exec  node, name it ExecuteLDVAR, and edit it as follows:
 - Script Name: ldvar
 - Return Control: False
 - Create Call Record: False
 - Three variables are passed. Highlight the Name field and click + to add three elements.
Element (1): [Variable1]
Element (2): [Variable2]
Element (3): [Variable3]
 - Trace Variables: False

When finished, the call flow should look like the following figure:

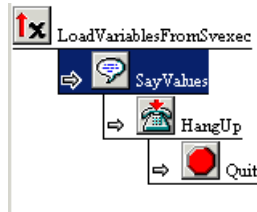


7. Simulate, verify, and save the application.
8. If you have a development system, generate code and transfer and install the application.

Create the Ldvar Application

You create an application named Ldvar to accept (load) the values of the variables sent from the svexec application. The Ldvar application does not return control to svexec. 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. However, the elements are received in the same order as in the Save and Exec node that passed them, so keep that in mind when naming the nodes in the new application.

1. Create the custom variables for this application.
 - [Name]. A character variable
 - [PhoneNumber]. A character variable, Spoken As AC
 - [NumberValue]. A number variable
2. Drag a Load Variables node  onto the design pad and name it LoadVariablesFromSVEEXEC. Edit it as follows:
 - Highlight the Name field and double-click the + to add three elements. Assign these elements to the following variables:
 Element (1): [Name]
 Element (2): [PhoneNumber]
 Element (3): [NumberValue]
 - Set Trace Variables to: False
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 Ldvar should look like the following figure:



5. Simulate and verify the application.
6. If you have a development system, generate code and transfer and install the application.
Do not assign the application to a voice channel. (You want svexec to be assigned to the channel.)

Test and trace the svexec application

To test and trace the svexec application:

1. Connect to the Avaya IR or CONVERSANT system and log in.
2. Start a trace with the following command (where ## is the assigned channel):
trace ## area SE | tee /tmp/svexec- ## .1out
3. Start a trace with the following command (where ## is the assigned channel):
trace ## | tee /tmp/svexec- ## .2out

Modify the applications to trace

To modify the applications so they will trace the variables:

1. Go back into both applications and set Create Call Record and Trace Variables to True.
2. Generate, transfer, and install the applications.
3. Test and trace the applications.
The trace output should now trace the variables.

To continue, see [Capturing voice from callers in TAS applications](#) on page 487.

Using stored procedures in TAS applications

You can use the StoredProc and StoredProcNext external functions 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 the StoredProc and StoredProcNext external functions](#) on page 484.

Related topics:

[Using the StoredProc and StoredProcNext external functions](#) on page 484

Using the StoredProc and StoredProcNext external functions

In this topic, you specify arguments for stored procedure external functions in a store locator application.

The locate application allows callers to find store locations near a specified zip code. It executes the LOCATE stored procedure, which exists on a database. The locate application does the following:

- Answers the telephone.
- Prompts the caller for a zip code.
- Creates a parameter string that includes the zip code.
- Uses the StoredProc external function to execute the LOCATE stored procedure on the 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 the StoredProcNext external function to retrieve the next record returned by the previous stored call.
- Announces the street, city, and phone number to the caller and continues to check for additional records.

To complete the locate application:

1. Open the slocate application and save it as locate.
The slocate 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.
 - Ensure that the external function palette is displayed.
Notice the StoredProc and StoredProcNext external functions on the palette. These are the functions you use to execute stored procedures and to retrieve data.
 - 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. The input is saved in both the [CollectedDigits] and [ZIPCODE] variables. This allows the zip code to be spoken as individual digits.
3. Locate the SetParamString node and specify its parameters:

- Assign the value of [Param1] to [Params].
You could assign a fixed string to the variable, instead of placing the string in a 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.

4. Locate the GetFirstStore node and specify its parameters:

The StoredProc external function executes a stored procedure and retrieves the results returned by the stored procedure.

- Put Return Code In: [ReturnCode]
- SPNAME: LOCATE
The stored procedure that is executed by this application is named LOCATE.
- PARAMS: [Params]
The [Params] variable holds the parameter string that you created in the SetParamString node and identifies the parameters to send to and receive from the stored procedure.
- DIPNAME: DBDIP2
[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.
- SPRETCODE: NO
The StoredProc external function 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.
- SPRESULTSET: YES

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.

- **RESULT: [Result]**

The stored procedure returns the result to the [Result] variable. If you specify multiple output parameters in the PARAMS attribute, the values are returned in a comma-delimited string. When SPRETCODE is set to YES, the return code is the first value returned in the variable.

5. Locate the ParseData node and specify its parameters.

- Parse the comma-delimited fields of the [Result] variable into the [STREET], [CITY], and [PHONE] variables.
- Branch to the (NoResult) branch if the value of the [ReturnCode] variable equals 0.
The return code will equal zero if no record is found for the zip code.

6. Locate the GetNextStore node and specify its parameters:

The StoredProcNext external function retrieves the next record returned by the GetFirstStore call.

- Put Return Code In: [ReturnCode]
- **RESULT: [Result]**
The stored procedure returns the results for the next record to the [Result] variable.

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 ParseData node and continues.

The remaining nodes are complete. This completes the locate application.

8. Simulate, verify, and save the application.

Before this application will work, the stored procedure must exist on a database on the target system.

To continue, see [Capturing voice from callers in TAS applications](#) on page 487.

Capturing voice from callers in TAS applications

This tour shows you how to specify and use the Voice Capture node to record a caller's spoken response. You also learn how to delete recorded messages, and you have the opportunity to write an application that plays back the recordings of callers' spoken responses.

To continue, see [Using the Voice capture node](#) on page 487.

Related topics:

[Using the Voice capture node](#) on page 487

[Using the Msg_Delete external function](#) on page 491

Using the Voice capture node

In this topic, you create an application that uses the Voice Capture node to record (capture) the caller's name, address, and telephone number. Callers are asked to speak the information after they hear a beep. The recorded information received from callers is then stored in a database table.

The captur application contains the following custom number-type variables:

- Duration variables: [Name_dur], [Address_dur], and [Phone_dur]. Store the time (in seconds) taken to record caller names, addresses, and telephone numbers.
- Combination variables: [Name_NX], [Address_NX], and [Phone_NX]. For each variable, two pieces of data reside in a single numeric field. These include a talkfile number (255) and a phrase tag number automatically assigned by the system. The two pieces of data are run through an algorithm to put them in a single variable.
- Return code variables: [Name_ret], [Address_ret], and [Phone_ret]. Store the return code that designates whether the voice capture succeeded for each of these data points.

The captur application also contains a database table named [VCTABLE], which contains three number fields. This table contains information that is used to store the recorded speech in the appropriate file location on the Avaya IR or CONVERSANT system. The following are the fields in this table:

- NX_NAME
- NX_ADDRESS
- NX_PHONE

Create the captur application

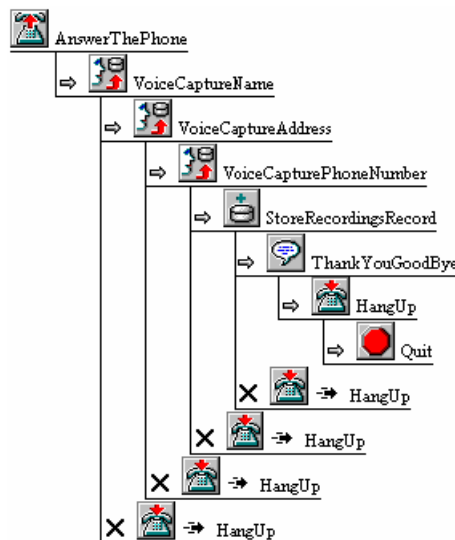
To create the captur application:

1. Create a new TAS application, naming it captur.
2. Drag an Answer Call node to the design pad and name it AnswerThePhone.

3. Add a Voice Capture  node and name it VoiceCaptureName.
4. Create the following three number-type variables needed for this node:
 - [Name_dur]
 - [Name_NX]. Set Speak Using to Recorded, Spoken As to NX, and Phrase Table to be blank.
If you do not set Phrase Table to be blank, you can not play back what the caller said.
 - [Name_ret]. Set Length to 3.
5. In the Node Inspector, edit the VoiceCaptureName node as follows:
 - The initial prompt should say:
Please say your name after the beep.
 - Set the Play Beep Before Recording field to True.
Callers hear a beep before they start talking.
 - Set the Max Length field to 45 (the default) for a maximum recording time of 45 seconds.
 - For the Rate field, select G.711.
The drop-down menu for this field displays the voice-encoding algorithms supported by the Avaya IR or CONVERSANT system.
 - For the Terminate On field, select Silence Detection (the default).
 - For the Initial Timeout and Completion fields, use the defaults.
 - For the Save In Variable field, select [Name_NX].
This is the variable that points to where the encoded speech is stored.
 - Set the Save On Hangup field to False.
All three pieces of data must be collected before writing a database record for future retrieval: [Name_NX], [Name_dur], and [Name_ret].
 - For the Actual Phrase Length field, select [Name_dur].
This holds the actual length in seconds of the recording.
 - For the Return Field field, select [Name_ret].
 - Do not use the Confirmation tab fields for this topic.
6. Add another Voice Capture node to the pass branch of VoiceCaptureName and name it VoiceCaptureAddress.
7. Create the following three number-type variables needed for this node:
 - [Address_dur]
 - [Address_NX]. Set Speak Using to Recorded, Spoken As to NX, and Phrase Table to be blank.
 - [Address_ret]. Set Length to 3.
8. Edit the VoiceCaptureAddress node as follows:
 - The initial prompt should say:
Please say your address after the beep.
 - Play Beep Before Recording: True

- Max Length: 45
 - Rate: G.711
 - Terminate On: Silence Detection
 - Initial Timeout: (default)
 - Completion: (default)
 - Save In Variable: [Address_NX]
 - Save On Hangup: False
 - Actual Phrase Length: [Address_dur]
 - Return Field: [Address_ret]
9. Add another Voice Capture node to the Pass Branch of VoiceCaptureAddress and name it VoiceCapturePhoneNumber.
 10. Create the following three number-type variables needed for this node:
 - [Phone_dur]
 - [Phone_NX]. Set Speak Using to Recorded, Spoken As to NX, and Phrase Table to be blank.
 - [Phone_ret]. Set Length to 3.
 11. Edit the VoiceCapturePhoneNumber node as follows:
 - The initial prompt should say:
Please say your phone number after the beep.
 - Play Beep Before Recording: True
 - Max Length: 45
 - Rate: G.711
 - Terminate On: Silence Detection
 - Initial Timeout: (default)
 - Completion: (default)
 - Save In Variable: [Phone_NX]
 - Save On Hangup: False
 - Actual Phrase Length: [Phone_dur]
 - Return Field: [Phone_ret]
 12. Create a database table to store the voice recordings so that you can play them back later. Name the database table VCTABLE.
 13. Create the three number-type fields:
 - [NX_NAME]
 - [NX_ADDRESS]
 - [NX_PHONE]
 14. Share out the [VCTABLE] database table.

15. Add an Add Record node to the pass branch of VoiceCapturePhoneNumber and name it StoreRecordingsRecord. Edit it as follows:
 - Table to add record: [VCTABLE]
 - Assignments to new record:
 - NX_NAME – [Name_NX]
 - NX_ADDRESS – [Address_NX]
 - NX_PHONE – [Phone_NX]
16. Add an Announcement node, name it ThankYouGoodBye, and edit the initial prompt to say the following:
The voice captures have been stored in the database table. Thank you. Good bye.
17. Add a Disconnect node and name it HangUp.
18. Add a Return node and name it Quit.
19. Make all the remaining open branches jump to the HangUp node.
When finished the call flow should look like the following figure:



20. Save and verify the application.
At this point, you would normally simulate the application. However, the Simulation tool does not accurately simulate the Voice Capture node, so skip that step. If you want to simulate the application, go ahead. It adds a record to the database table. The simulation script, however, contains some entries that may be confusing to you and do not accurately reflect what happens on the Avaya IR or CONVERSANT system.

Generate, Transfer, Install, Assign, and Test the Application

To generate, transfer, install, assign, and test the application:

1. If you have a development system, generate code and transfer and install the application.
2. Connect to the Avaya IR or CONVERSANT system and log in.

3. Use SQL*Plus to view the contents of the [VCTABLE] database table and verify that it is empty. If the table is not empty, delete all the records with the following command:
delete vctable;
4. Test the application as follows:
When prompted, say your name, address, and telephone number. Verify that you hear the message that says the record was written to the database table.
5. View the contents of the [VCTABLE] table with the following SQL*Plus command:
select * from vctable;
6. Trace the application with the following command (where ## is the assigned channel):
trace ## | tee /tmp/captur- ## .1out
7. Trace the application with the following command (where ## is the assigned channel):
trace ## area SE | tee /tmp/captur- ## .2out
8. Trace the application with the following command (where ## is the assigned channel):
trace dip1 | tee /tmp/captur- ## .3out
9. View the contents of the [VCTABLE] database table again.
Remember that recorded or encoded voice files and database table records are added while you are testing. While database records may not take up much disk space, encoded voice files can consume disk space rapidly. You usually want to delete messages after you have listened to them to save disk space. [Using the Msg_Delete external function](#) on page 491 explains how to delete messages.

To continue, see [Using the Msg_Delete external function](#) on page 491.

Using the Msg_Delete external function

In this topic, you create an application using the Msg_Delete external function node in conjunction with the Voice Capture node. The application you create uses the Msg_Delete external function to delete the recorded speech collected from callers, after the information is transcribed. Make a practice of deleting messages to free up valuable disk space when the information is no longer needed.

The capturpl application:

- Queries the existing database table [VCTABLE].
- Announces the information to the transcriber.
- Prompts the transcriber to delete or skip the current information.
- Takes action based on transcriber response:
 - If the transcriber chooses to delete the current information, deletes both the recorded speech and the database record pointing to the speech.
 - If the transcriber chooses to skip to the next record, retains the recorded speech and the database record.

The following four new number-type variables are created for use in capturpl:

- Combination variables: [Name_NX], [Address_NX], and [PhoneNumber_NX]. For each variable, two pieces of data reside in a single numeric field. These include a talkfile number (255) and a phrase tag number (automatically assigned by the system). The two pieces of data are run through an algorithm to put them in a single variable.
- [RecordCounter]. A number field used by a Set and Test node to check for the last record in a database table.

The existing database table [VCTABLE] is shared by this new application.

Create the application

To create the capturpl application:

1. Create a new TAS application and name it capturpl.
2. Import a private copy of the Msg_Delete external function.
3. Import a copy of the [VCTABLE] database table after you have shared it out from the captur application.
4. Create the following three new number-type variables:
 - [Name_NX]
 - [Address_NX]
 - [Phone_NX]

Set all these variables to Speak Using Recorded, Spoken As to NX, and the Phrase Table to blank.
5. Drag an Answer Call node to the design pad and name it AnswerThePhone.
6. Add a Query Table node and name it QueryDatabaseForAllRecordings. Edit it as follows:
 - Table to query: [VCTABLE]
 - Assignments from records:
 - NX_NAME: [Name_NX]
 - NX_ADDRESS: [Address_NX]
 - NX_PHONE: [Phone_NX]
7. Attach an Announcement node to the n branch of QueryDatabaseForAllRecordings and name it AnnounceRecordOfCaller. Edit it as follows:
 - The initial prompt should say:

[Name_NX] {sil.050} [Address_NX] {sil.050} [Phone_NX] {sil.050}

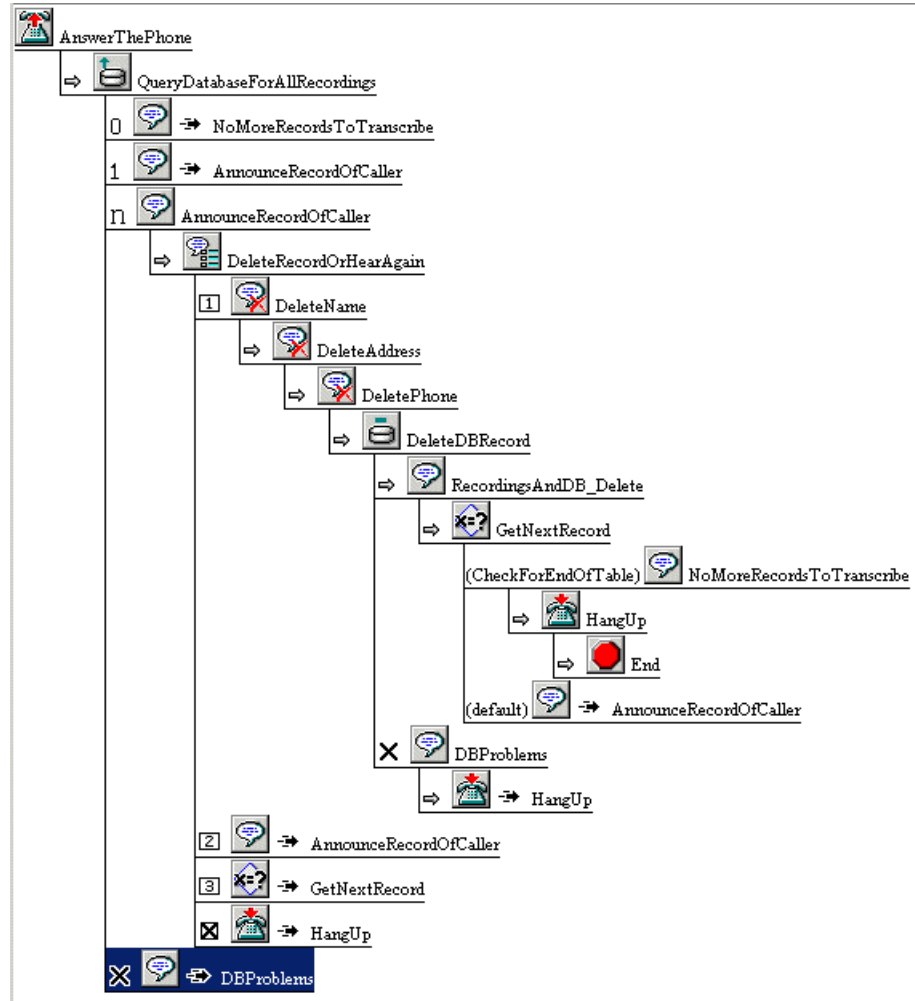
For the {sil.050} pauses, select 0.05 sec silence from the drop-down list of controls when { is typed.
 - Allow Interrupt: False.
8. Add a Menu node and name it DeleteRecordOrHearAgain. Edit it as follows:
 - The following four branches should be the defaults: 1, 2, 3, and X
 - The initial prompt should say:

To delete record, press one. {sil.050} To hear it again, press two. {sil.050}
To skip to the next record, press three.

9. Add a Msg_Delete  node to branch 1 and name it DeleteName. Edit it as follows:
 - Put Return Code In: leave empty
 - For Num NX?: NX Field (Talkfile & Phrase)
 - Phrase ID: Name_NX
 - Talkfile: 255
10. Add another Msg_Delete node to the pass branch of DeleteName and name it DeleteAddress. Edit it as follows:
 - Put Return Code In: leave empty
 - For Num NX?: NX Field (Talkfile & Phrase)
 - Phrase ID: Address_NX
 - Talkfile: 255
11. Add another Msg_Delete node to the pass branch of DeleteAddress and name it DeletePhone. Edit it as follows:
 - Put Return Code In: leave empty
 - For Num NX?: NX Field (Talkfile & Phrase)
 - Phrase ID: Phone_NX
 - Talkfile: 255
12. Add a Delete Record node to the pass branch of DeletePhone and name it DeleteDBRecord. Edit it as follows:
 - Table to search: [VCTABLE]
 - Selection criteria: NX_NAME = Name_NX
 - Delete multiple records: True
DeleteDBRecord removes the records that identify the locations on the Avaya IR or CONVERSANT system where the recorded speech was stored.
13. Add an Announcement node to the pass branch of DeleteDBRecord. Name the node RecordingsAndDB_Delete. Edit it so that the initial prompt says:
The recordings and database record have been successfully deleted.
14. Add a Set and Test node to the pass branch of RecordingsAndDB_Delete. Name the node GetNextRecord.
15. Insert a branch above the (default) branch and name it CheckForEndOfTable.
16. Edit the GetNextRecord node as follows:
 - Click the Expressions Tab and then add an element and set the following:
Expression Type: NextRecord
Next Record Number: [RecordCounter].
 - Click the Branch Conditions tab and add two elements.
Condition (1): Goto Branch CheckForEndOfTable if [RecordCounter] = 0

Condition (2): default

17. Add an Announcement node to the (CheckForEndOfTable) branch and name it NoMoreRecordsToTranscribe. Edit it as follows:
 - The initial prompt should say:
There are no more records to transcribe. Thank you. Goodbye.
 - Allow Interrupt: False
 18. Add a Disconnect Call node to the pass branch of the NoMoreRecordsToTranscribe branch and name it HangUp.
 19. Add a Return node to the pass branch of the HangUp node and name it End.
 20. For the (default) branch of GetNextRecord, jump to AnnounceRecordOfCaller.
 21. For the fail branch of DeleteDBRecord, add an Announcement node and name it DBProblems. Edit it as follows:
 - The initial prompt should say:
We are having database trouble. Please call back later.
 - Make this node jump to the HangUp node.
 22. For the remaining branches, set them as follows:
 - From the 0 branch of the QueryDatabaseForAllRecordings, jump to the NoMoreRecordsToTranscribe node.
 - From the 1 branch of the QueryDatabaseForAllRecordings node, jump to the AnnounceRecordOfCaller node.
 - From the fail branch of the QueryDatabaseForAllRecordings node, jump to the DBProblems node.
 - From menu choice [2] of the DeleteRecordOrHearAgain node, jump to the AnnounceRecordOfCaller node.
 - From menu choice [3] of the DeleteRecordOrHearAgain node, jump to the GetNextRecord node.
 - For the fail branch of the DeleteRecordOrHearAgain node, jump to the HangUp node.
- When finished, the call flow should look like the following figure:



Test the Application

To test the capturpl application:

1. If you have a development system, generate code and transfer and install the application.
2. Connect to the Avaya IR or CONVERSANT system and log in.
3. Start a trace on the application with the following command (where ## is the assigned channel):
`trace dip1 | tee /tmp/capturpl- ## .1out`
4. Start a trace on the application with the following command (where ## is the assigned channel):
`trace ## area SE | tee /tmp/capturpl- ## .2out`
5. Start a trace on the application with the following command (where ## is the assigned channel):
`trace ## | tee /tmp/capturpl- ## .3out`

6. Use SQL*Plus to check the database table [VCTABLE] and verify that the records do not exist.
7. View talkfile 255. Verify that the recorded voice phrase numbers are not listed.

To continue, see [Creating host applications](#) on page 496.

Creating host applications

Passing data between the Avaya IR or CONVERSANT system and a host computer is a common requirement. This tour introduces you to developing host applications in IVR Designer. Detailed development of host applications and their implementation on Avaya IR or CONVERSANT systems is beyond the scope of this tour.

IVR Designer applications can:

- Use 3270 IBM-type mainframe host screens and data.
- Send data to hosts and receive data from them.

Three host nodes are available for this purpose:

- Host Connection
- Host Send
- Host Send/Receive

This tour shows you how to set up the Host Send/Receive and Host Connection nodes to effectively communicate with 3270 host screens.

To continue, see [Host application development guidelines](#) on page 496.

Related topics:

[Host application development guidelines](#) on page 496

[Creating a host application](#) on page 498

Host application development guidelines

IVR Designer host applications that you develop need to mimic the behavior of the existing 3270 host screens and data. As a result, integrating existing 3270 host data with an IVR Designer application requires careful planning. If you plan to create IVR Designer applications that uses 3270 host data, follow the recommendations discussed here before you create the applications.

1. Make sure that you understand the current application as it executes on the 3270 host. If the application exists, view a demonstration to identify:
 - General screen appearance. Ask for printouts of the screens, annotated to identify data movement.
 - Error message screen information.

- The screen on which the current application starts. If the host session is reset, what screen shows first?
 - How to log in to the application.
 - Keys used to move backward and forward between screens.
 - How to log out of the application.
 - How to get to screens where data is input or received.
 - Any BLANK screens in the application.
 - Fields that uniquely identify each screen.
2. Create a graphical representation of the application that shows the flow of screens and data:
 - Create a flow of screens with sent and received data for scenarios where correct data is sent and the correct data is received.
 - Identify error conditions. For example, what screen is returned if the user inputs an invalid ID number or presses an invalid key?
 - Note the keys that are pressed to move backward and forward within the application.
 - If possible, get a test login ID, password, and test data from the host application administrator.
 3. Capture the screens with the IVR Designer Screen Mapping tool.
 The Screen Mapping tool lets you drag and drop the information into the fields from screens that you capture from the host machine. Gain access to this tool using the Screen Mapping  icon. See [Capturing host screens](#) on page 203 for more information on Screen Mapping.

The following two call flows are required, at a minimum, within an IVR Designer application that uses 3270 host data :

- The [Main] call flow
- The Host Flow

3270 host applications require that:

- The corresponding IVR Designer application is assigned to a voice channel and to a 3270 host session.
- The host session is in the logged in state. A logged in state indicates that the host session is at the Transaction Base Screen. The Host Connection node, included at the beginning of an IVR Designer [Main] call flow makes sure of this. If the host session is not logged in, the IVR Designer application does not start.

To continue, see [Creating a host application](#) on page 498.

Creating a host application

This topic introduces you to the requirements for creating an application that uses 3270 host data. You do not create a complete working application, but a partial application that:

- Provides 3270 host login information to an IVR Designer application.
- Uniquely identifies 3270 host screens to an IVR Designer application.
- Specifies fields on each of the uniquely identified screens. Fields on a screen are one of three types:
 - For data sent to the 3270 host
 - For data received from the 3270 host
 - For data both sent and received
- Specifies the sequence of screens used within the existing 3270 host application to login to the application, log out of the application, and recover from errors or timeouts.

The following three 3270 host screens are used for this application:

```

*****
*
*      CS01
*
*
*
*
*
*
*
*
*      LOGIN:
*
*      PASSWORD:
*
*
*
*
*
*
*
*
*
*
*****

```

```

*****
*
*   CS02
*
*
*           MAIN MENU
*
*   Account Balance      PF1
*
*   ...
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*****

```

```

*****
*
*   CS03
*
*
*           CUSTOMER INQUIRY SCREEN
*
*   Account Number:  _____
*
*   Customer Name:
*
*   Customer Balance: $
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*****

```

To continue, see [Creating the host definition](#) on page 499.

Related topics:

[Creating the host definition](#) on page 499

[Creating the Host and \[Main\] call flows](#) on page 505

Creating the host definition

The host application that you create in this topic:

- Starts at the LOGIN screen.
- Provides a login ID and password.
- Presses the Enter key.

- Moves to the MAIN_MENU screen after the login ID and password are validated.
- Presses the PF1 key to move to the CUSTOMER INQUIRY SCREEN (CUST_INQ).
- Enters the nine-digit customer ID on the screen and sends it to the 3270 host.
- Receives the customer's account balance and name from the 3270 host.

The host application needs:

- A screen to start the application (a screen to log in to), called the Login Base Screen. In this topic, the Login Base Screen is the LOGIN screen.
- A primary screen where the host transaction starts, called a Transaction Base Screen. This is normally not the same as the Login Base Screen. Typically, you do not want to log out and log back in for each successive call. It is better if you can keep coming back to a particular screen where you want to actually start the exchange of data in each call. That's the idea behind the Transaction Base Screen.
In this topic, the Transaction Base Screen is the CUSTOMER INQUIRY SCREEN (CUST_INQ). This is the screen at which the host application starts to execute. If the host application does not find the CUSTOMER INQUIRY SCREEN, it considers the 3270 host connection to be down.

Any screen that is not the Login Base Screen or the Transaction Base Screen is considered by IVR Designer simply to be other. In this topic, the MAIN MENU screen is considered to be an other screen by the host application (designated as None in the Base Type column).

To create the [Main] call flow and the host definition:

1. Create a new TAS application and name it host.
2. Right-click the Hosts Manager and select New.
3. Name the new host definition Host1.
The Host Definition Editor appears.
4. Identify the 3270 host screens to the IVR Designer application by setting the fields as follows:
First screen (row):
 - Screen Name: LOGIN
 - Base Type: LoginSecond screen (row):
 - Screen Name: MAIN_MENU
 - Base Type: NoneThird screen (row):
 - Screen Name: CUST_INQ
 - Base Type: TransactionWhen finished, the Host Definition Editor(Host Definition tab) should look like the following:

The image shows a Windows-style dialog box titled "Host Definition Editor - '[Host1]'". It has four tabs: "Host Definition", "Logins", "Screen Identification", and "Field Definition". The "Host Definition" tab is active, showing "Host Parameters" with three input fields: "Host Timeout:" (45), "Logical Unit Timeout:" (0), and "Unrecognized Screen Timeout:" (45). Below this is the "Screen Definition" section, which contains a table with columns: "Screen Name", "Base Type", "Blank Screen", "Rows", and "Columns". The table has three rows: "LOGIN" (Base Type: Login, Blank Screen: False, Rows: 24, Columns: 80), "MAIN_MENU" (Base Type: None, Blank Screen: False, Rows: 24, Columns: 80), and "CUST_INQ" (Base Type: Transaction, Blank Screen: False, Rows: 24, Columns: 80). At the bottom of the dialog are "OK", "Cancel", and "Apply" buttons.

Screen Name	Base Type	Blank Screen	Rows	Columns
LOGIN	Login	False	24	80
MAIN_MENU	None	False	24	80
CUST_INQ	Transaction	False	24	80

The Avaya IR or CONVERSANT system should be set up with a particular number of host sessions. Each host session is a line to the 3270 host and corresponds to a login on the 3270 host. The IVR Designer application being created logs in to the host with the same login and password each time. For this application, the number of logins available to the IVR Designer application is 12 sessions.

5. Click the Logins tab.
6. Right-click the first row in the Login/Password area of the screen and select the Fill Login/Password with default... option.
The Create Default Login dialog box appears.
7. Edit its fields as follows and click Apply.
 - Login name: ivr
 - Password: ivrpasswd
 - Number of Host Sessions: 12

If you accidentally click OK instead of Apply, right-click the Hosts Manager and select Edit Host from the popup menu to return to the Host Definition Editor.
8. Click the Screen Identification tab of the Host Definition Editor.
This is where you specify to the IVR Designer application how to uniquely identify each screen. If a screen is not identifiable to an IVR Designer application, the application considers the screen to be an Unrecognized Screen. Sometimes it is impossible to identify every screen that might be encountered by an IVR Designer application at the beginning of development. Some screens are not encountered during normal operation. Most 3270 host programs, however, have some means of recovering from error screens. For example, you might need to clear the screen and

then input LOGOUT to return to the login base screen and then log in and progress to the transaction base screen before the IVR Designer application can continue.

9. For Screen Name, select LOGIN.
The CS0# field is unique for each of the three screens, so you can use it to identify each screen. In this case, the screen identifier is CS01. This identifier is located in row 3, starting at column 5.
10. Complete the fields for this identifier as follows and click Apply.
 - Row: 3
 - Column: 5
 - Type: Text Present
 - Text: CS01
In this example, CS01 may be enough to uniquely identify this screen. However, as a precaution, you create two more identifiers for this screen. The word LOGIN appears on the screen beginning at row 10, column 5. Use this as the second identifier.
11. Move to the second row, complete the fields as follows and click Apply.
 - Row: 10
 - Column: 5
 - Type: Text Present
 - Text: LOGIN
The field where the login ID is entered begins at row 10, column 12. This is where you are likely to find a cursor as an indicator to send a string of text to the 3270 host. Use this as the third identifier.
12. Move to the third row, complete the fields as follows and click Apply.
 - Row: 10
 - Column: 12
 - Type: Cursor
13. For Screen Name, select MAIN_MENU.
The text string CS02 is a unique identifier.
14. Complete the fields for this identifier as follows and click Apply.
 - Row: 3
 - Column: 5
 - Type: Text Present
 - Text: CS02
15. For Screen Name, select CUST_INQ.
The text string CS03 is a unique identifier.
16. Complete the fields for this identifier as follows and click Apply.

- Row: 3
 - Column: 5
 - Type: Text Present
 - Text: CS03
17. Click the Field Definition tab.
This tab is used to define fields on screens sent to or received from the host. You start by defining fields on the LOGIN screen.
 18. For Screen Name, select LOGIN.
On this screen, the login ID and the password must be sent to the host along with an Enter command. The host should respond by going to the MAIN MENU screen, but return no data to the LOGIN screen. Therefore, you only send data to the host.

The login ID information is sent to the 3270 host starting at row 10, column 12. The maximum possible length for the login ID is 15 characters.
 19. Define the fields for the login ID as follows and click Apply.
 - Name: LOGIN_ID
 - Location: Regular
 - Row: 10
 - Column: 12
 - Length: 15
 - Direction: To Host
 - Class: Character
The password information is sent to the 3270 host starting at row 12, column 15. The maximum possible length for the login ID is 10 characters.
 20. Define the fields for the password (on the second line) as follows and click Apply.
 - Name: PASSWORD
 - Location: Regular
 - Row: 12
 - Column: 15
 - Length: 10
 - Direction: To Host
 - Class: Character
Next, you define the appropriate fields, if any, on the MAIN MENU screen.
 21. For Screen Name, select MAIN_MENU.
MAIN_MENU does not have any fields. It responds to specific keys. Later in this example, you use a PF1 key command to move to the CUST_INQ screen.

Next, you define the appropriate fields, if any, on the CUSTOMER INQUIRY SCREEN.

22. For Screen Name, select CUST_INQ.

On this screen there are three fields.

- The customer's account number is collected from a Prompt and Collect within the IVR Designer script. This field is sent to the host and is a character field.
 - The customer's name is received from the host. This field is a character field.
 - The customer's balance is received from the host. This field is a number field.
- The 3270 host defines the type of data it is expecting. You must also find out from the host administrator the data types of the specific fields on each screen.

23. Define the first field (for the customer's account number) as follows and click Apply.

- Name: CUST_ACCT_NO
- Location: Regular
- Row: 8
- Column: 33
- Length: 9
- Direction: To Host
- Class: Character

24. Define the second field (for the customer's name) as follows and click Apply.

- Name: CUST_NAME
- Location: Regular
- Row: 10
- Column: 32
- Length: 40
- Direction: From Host
- Class: Character

25. Define the third field (for the customer's account balance) as follows and click Apply.

- Name: CUST_BALANCE
- Location: Regular
- Row: 12
- Column: 37
- Length: 10

- Direction: From Host
- Class: Number

26. Click OK to exit the Host Definition Editor.

To continue, see [Creating the Host and \[Main\] call flows](#) on page 505.

Creating the Host and [Main] call flows

The next step in creating a host application is creating the Host Flow. The primary purpose of the Host Flow is to set up the specific interactions between the Avaya IR or CONVERSANT system and the host.

The [Main] call flow sends and receives data collected from the caller. The Host Flow simply manages how the host session (or connection) logs in, logs out, and recovers. The flow you create is active during the host session and tells the Avaya IR or CONVERSANT system how to:

- Login and get to the Transaction Base Screen, in this example the CUST_INQ screen.
- Log out and get to the Login Base Screen if the Transaction Base Screen is not found.
- Recover from a timeout or an encounter with an unrecognized screen.

Completing this flow requires a thorough understanding of the 3270 host screens that the IVR Designer application encounters. You must know how to move between screens to always have a way to return to a known screen (the Login Base Screen or the Transaction Base Screen).

The Login sequence

The object of the Login sequence is to get to the Transaction Base Screen and then stop there to await further instructions. Under the Login node, you must account for any screen the host session may encounter. In an ideal situation, the following events would occur:

- The session encounters the Login Base Screen (LOGIN), send the LOGIN screen the default login ID and password and then sends the Enter key command.
- The host displays the MAIN_MENU screen and a PF1 command is sent to it.
- The host displays the CUST_INQ screen.

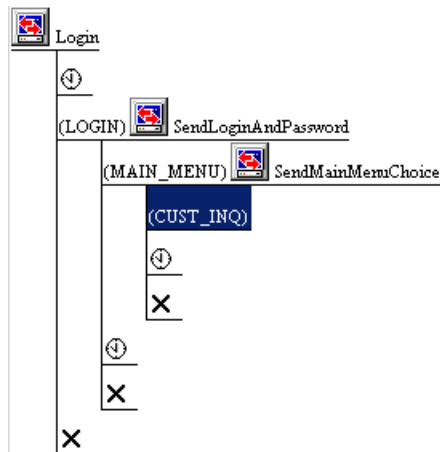
To set up the Login sequence:

1. Right-click the Host Manager. From the pop-up menu, select Edit Host Flow. IVR Designer displays a template call flow that is structured in three sections. The node that starts each of the sections is a Host Send/Receive node.

The clock symbols under the nodes indicate a time-out response.

2. Right-click the fail branch of the Login node and insert a branch. IVR Designer displays the Receive Screens For Host dialog box.
3. From the list of available screens, select LOGIN and click OK.
4. Drag a Host Send/Receive  node to the (LOGIN) branch you just created. Name the node SendLoginAndPassword.
5. Edit the node as follows:

- Host Name: [Host1]
 - Release Logical Unit: False
 - Send Screen Name: LOGIN
 - Send Field Mappings: Click
For LOGIN_ID, change Type to Special and Value to \$login.
For PASSWORD, change Type to Special and Value to \$password.
 - Aid Key: Enter
6. Insert a branch above the Timeout branch of the SendLoginAndPassword node.
 7. From the list of available screens, select MAIN_MENU and click OK.
 8. Attach a Host Send/Receive node to the (MAIN_MENU) branch you just created.
Name the node SendMainMenuChoice.
 9. Edit the node as follows:
 - Host Name: [Host1]
 - Release Logical Unit: False
 - Send Screen: MAIN_MENU
 - Send Field Mappings: Inactive and grayed out because this screen does not have any defined fields
 - Aid Key: PF1
 10. Insert a branch above the Timeout branch of the SendMainMenuChoice node.
 11. From the list of available screens, select CUST_INQ and click OK.
When completed to this point, the Login sequence should look like the following figure:



This is where this branch of the Login sequence ends (successfully) and waits for a nine-digit customer account number to be sent to it.

The sequence just defined is the ideal login sequence. Remember, to complete the [Host1] flow, you must fully understand the behavior of the 3270 host program and

its associated screens and data. Then you must try to plan for and proactively deal with any contingency.

The Logout sequence

The Logout sequence starts when the logout command is requested of a particular host session running on the Avaya IR or CONVERSANT system. The Logout sequence may also be executed if the Login or Recover sequence contains a jump to the Logout node.

Creating the [Main] call flow

As mentioned earlier, in order for the IVR Designer script to start and answer the call, the host session on the Avaya IR or CONVERSANT system must be logged in and waiting at the Transaction Base Screen. The CUST_INQ screen must be ready to send data to and receive data from the 3270 host system.

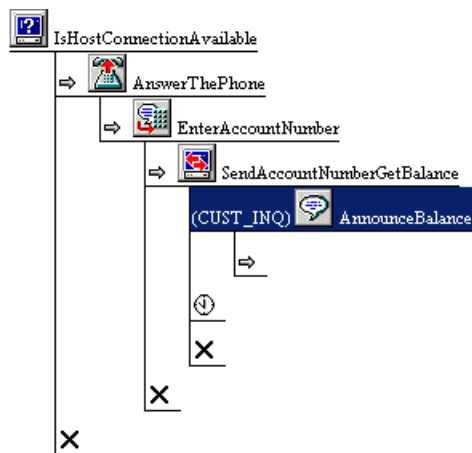
1. Drag a Host Connection  node to the [Main] call flow design pad. Name it IsHostConnectionAvailable.
The Avaya IR or CONVERSANT system looks for the Transaction Base Screen. If it is found on the session, the [Main] call flow takes the pass branch. If not, it takes the fail branch.

If the 3270 host session is not available, you might tell the caller to call back and then end the call. You could also write information to a database table that indicates the date and time of the failure before quitting from the script.
2. Add an Answer Call node and name it AnswerThePhone.
3. Create a new character-type variable and name it CI_AccountNumber. This is for a nine-digit account number to be sent to the host.
4. Add a Prompt and Collect node and name it EnterAccountNumber. Edit the node as follows:
 - Create the initial prompt to say:
Please enter your nine-digit account number.
 - On the Response tab, the maximum and minimum lengths should both be 9.
 - Save Formatted Response In: [CI_AccountNumber]
The script is now ready to send the caller's account number to the CUST_INQ screen.
5. Add a Host Send/Receive node to the pass branch of the EnterAccountNumber node and name it SendAccountNumberGetBalance. Edit the node as follows:
 - Host Name: [Host1]
 - Send Screen Name: CUST_INQ
 - For Send Field Mappings, click
Host Screen Field Mappings. Type: Variable, Value: CI_AccountNumber
 - Aid Key: Enter
When the application runs, the CUST_INQ screen should now return the customer name and balance data.
6. Insert a branch above the Timeout branch of the SendAccountNumberGetBalance node.

The application is expecting data to be returned from the CUST_INQ screen, so when the Receive Screens For Host dialog box appears, select CUST_INQ.

7. Add an Announcement node to the (CUST_INQ) branch and name it AnnounceBalance.
8. Move back to the Hosts section of the Globals Manager. Right-click and choose Edit Host.
9. Click the Field Definition tab and choose CUST_INQ as the Screen Name.
10. Right-click any of the three variables listed and select Create Variables Interactively. Select Yes for CUST_NAME and CUST_BALANCE to create CUST_NAME1 and CUST_BALANCE1 that can now be used in your [Main] call flow. Notice the variables list now includes CUST_NAME1 and CUST_BALANCE1.
11. Edit the AnnounceBalance initial prompt to say:
Your balance is, [CUST_BALANCE1]

When finished to this point, the [Main] call flow for this application should look like the following figure:



This is the end of the topic. You have not completed this application, but you should know enough to do that. You have learned how to set up a host application to send and receive data from a 3270 host system.

To continue, see [Testing and troubleshooting TAS applications](#) on page 521.

Avaya Interaction Center integration

The Avaya IR system can integrate with computer telephony services, such as those provided by Avaya Interaction Center (IC). Avaya IC is a comprehensive Multimedia Contact Center solution for customer relationship management. It routes, manages, records, and reports on

customer interactions across communication channels including Web, email, and telephony. Avaya Computer Telephony is the telephony component of Avaya IC.

With IC Integration, IVR Designer applications can perform actions such as:

- Retrieve values stored in a database through Avaya Computer Telephony
- Set values in a database through Avaya Computer Telephony
- Raise Avaya Computer Telephony alarms
- Invoke Avaya Computer Telephony transactions

IC integration is accessed through a single external function, named Vesp_dip. For more information on the Vesp_dip, see [Interaction Center external functions](#) on page 700.

To continue, see [Using the Vesp_dip external function](#) on page 509.

Related topics:

[Using the Vesp_dip external function](#) on page 509

Using the Vesp_dip external function

In this topic, you complete an application that uses the Vesp_dip external function to integrate with Avaya Computer Telephony. To simplify the process, a base application includes all the required nodes. Your role is to specify the nodes that use the Vesp_dip external function.

The payoff application allows loan company customers to call a payoff line to determine their loan payoff amount or to talk with a loan processor. The application uses the Vesp_dip to pass information to and retrieve information from Avaya Computer Telephony.

The payoff application does the following:

- Answers the telephone.
- Notifies Avaya Computer Telephony of the call.
- Retrieves the account number from an Avaya Computer Telephony database.
Prompts for the ID if it cannot be retrieved.
- Retrieves the associated PIN from an Avaya Computer Telephony database.
- Prompts for the PIN and matches it to the retrieved PIN.
- Presents a menu of choices:
 - 1 for loan payoff amount
 - 2 to transfer the call to a loan processor
 - 9 to end the call
- If the caller hangs up early, executes a call flow that notifies Avaya Computer Telephony of the end of the call.

To complete the payoff application:

1. Open the spayoff application and save it as payoff.
The spayoff application includes complete call flows. You will specify the arguments for the nodes that use the Vesp_dip external function.
2. Review the current application.
 - Review the custom variables in the application.
These variables are used to pass data to and retrieve data from an Avaya Computer Telephony database. Note the lengths of the variables. For more information on required variable lengths, see [Interaction Center external functions](#) on page 700.
 - Review the prompts and phrases for the application.
If you have proxy text-to-speech, edit the Phrase table to use text-to-speech. If you do not have proxy text-to-speech, record the phrases for the application.
 - Ensure that the external function palette is displayed.
Notice the Vesp_dip  external function on the palette. This is the function you use to integrate with Avaya Computer Telephony.
 - Open the node inspector so you can view the parameters for the nodes.
 - Review the TrapHangup Asynchronous Event node.
To effectively integrate with Avaya Computer Telephony, the application must notify Avaya Computer Telephony if the caller hangs up early.
 - Review the Set and Test nodes that check Return Codes.
Applications that integrate with Avaya Computer Telephony must do frequent error checking to ensure that data has been effectively passed between the Avaya IR system and Avaya Computer Telephony.
3. Locate the SetNewCall node and specify its parameters:
 - Put Return Code In: [ReturnCode]
 - VESP Function Name: newcall
The Vesp_dip node is used to perform a number of functions. In this case, the newcall function notifies Avaya Computer Telephony that a new call has arrived and requests an EDUID. The EDUID identifies the call within Avaya Computer Telephony. It is unique to the call and essential for all Avaya Computer Telephony requests. If available, Automatic Number Identification (ANI) and Dialed Number Identification Service (DNIS) information is stored with the EDUID in Avaya Computer Telephony.
 - Arg1: [ICID]
The EDUID retrieved from Avaya Computer Telephony is stored in the custom [ICID] variable.
 - Arg2: [Dummy]
All unused arguments must be set to blank with a dummy variable.
 - Arg3: [Dummy]
 - Arg4: [Dummy]

For more information on the Vesp_dip node functions, see [Interaction Center external functions](#) on page 700.

4. Locate the GetAcctNumber node and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: getvox
The getvox function retrieves a single value from the VOX server value cache in Avaya Computer Telephony. Use this function to retrieve stable values. Use the getvdu function to retrieve values that may change.
- Arg1: [ICID]
The EDUID is passed back to Avaya Computer Telephony to identify the call.
- Arg2: [DatabaseField]
The name of the database field to query is passed in this argument. The [DatabaseField] variable is set to account in the SetAcctDatabaseField node.
- Arg3: [Account]
The [Account] variable is used to receive the value from the account field in the Avaya Computer Telephony database.
- Arg4: [Dummy]
If Avaya Computer Telephony does not locate an account number associated with the ANI information for the call, the application must prompt for an account number.

5. Locate the SendAcctNumber node and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: tr2setacct
This is a custom transaction defined in Avaya Computer Telephony that receives the account number sent from the application, identifies the associated database record, and associates it with the call. For more information on creating custom transactions in Avaya Computer Telephony, see Avaya Interaction Center Workflow Designer.
- Arg1: [ICID]
The EDUID is passed back to Avaya Computer Telephony to identify the call.
- Arg2: [Account]
The [Account] variable is used to pass the account number entered by the caller to the account field in the Avaya Computer Telephony database.
- Arg3: [Dummy]
- Arg4: [Dummy]
If Avaya Computer Telephony cannot identify the account to associate with the call it returns an error code.

6. Locate the GetPIN node and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: getvdu
The getvdu function is used here to get the PIN value from the database record associated with the call. This value could be changed outside the application, so the cache that would be retrieved by the getvox function might not have the latest value.
- Arg1: [ICID]
The EDUID identifies the call.
- Arg2: [DatabaseField]
The name of the database field to query is passed in this argument. The [DatabaseField] variable is set to pin in the SetPINDatabaseField node.
- Arg3: [Pin]
The [Pin] variable is used to receive the value from the pin field in the Avaya Computer Telephony database.
- Arg4: [Dummy]
The PIN returned from Avaya Computer Telephony is matched to one entered by the caller. If the PINs match, the caller receives a menu of choices. The caller can hear the payoff amount, transfer to a loan processor, or end the call.

7. Locate the GetPayoff node and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: getvdu
The getvdu function is used here to get the payoff value from the database record associated with the call.
- Arg1: [ICID]
The EDUID identifies the call.
- Arg2: [DatabaseField]
The [DatabaseField] variable is set to payoff in the SetPayoffDatabaseField node.
- Arg3: [PayoffChar]
The [PayoffChar] variable is used to receive the value from the payoff field in the Avaya Computer Telephony database. The value is passed as a character variable.
- Arg4: [Dummy]
To speak the payoff value as currency, the value in [PayoffChar] is assigned to the variable [PayoffAmount] in the CheckPayoffReturnCode node before it is announced.

If the caller PIN does not match that returned from Avaya Computer Telephony, the application sends an alarm to Avaya Computer Telephony, transfers the call to a loan processor, and ends the call.

8. Locate the SendAlarm node and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: alarm
The alarm function raises an Avaya Computer Telephony alarm to indicate an exceptional condition. Using variable to pass the values of the arguments allows you to use a single SendAlarm node to send a variety of alarm conditions
- Arg1: [Arg1]
The [Arg1] variable is set to the alarm name in the previous Set and Test node.
- Arg2: [Arg2]
The [Arg2] variable is set to the alarm priority code.
- Arg3: [Arg3]
The [Arg3] variable is set to the text that describes the problem.
- Arg4: [Dummy]

9. Locate the TransferCall node and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: transfer
The transfer function transfers the call from the IR system to an agent at the specified extension.
- Arg1: [ICID]
The EDUID identifies the call.
- Arg2: [TransferExtension]
The [TransferExtension] variable is used to pass the extension to which to transfer the caller.
- Arg3: [Dummy]
- Arg4: [Dummy]

10. Locate the EndCall node in the [Main] call flow and specify its parameters:

- Put Return Code In: [ReturnCode]
- VESP Function Name: gone
The gone function notifies Avaya Computer Telephony that the call has ended and releases the EDUID.
- Arg1: [ICID]
The EDUID identifies the call.
- Arg2: [Reason]
The [Reason] variable, set in the previous Set and Test node, is used to identify the reason for ending the call.
- Arg3: [Dummy]
- Arg4: [Dummy]

After the call is ended, another Asynchronous Event node is used to stop trapping a caller hangup.

11. Locate the EndCall node in the [EarlyHangup] call flow and specify its parameters.

- Put Return Code In: [ReturnCode]
- VESP Function Name: gone
- Arg1: [ICID]
- Arg2: [Reason]
- Arg3: [Dummy]
- Arg4: [Dummy]

This completes the payoff application.

Before this application will work, the Avaya Computer Telephony VOX server must be programmed appropriately. For more information on programming the VOX server, see Avaya Interaction Center VOX Server Programmer's Guide.

To continue, see [Predictive Dialing System \(PDS\) integration](#) on page 514.

Predictive Dialing System (PDS) integration

The Avaya IR system can integrate with the Predictive Dialing System (PDS), also called the Dialer. The Dialer makes outbound calls and, upon caller contact, connects the called party to an available agent for job-related activities.

Integration of the IR system with PDS allows the PDS to make an outbound call and pass that voice and data to the IR system. The IR system processes the call as it would any inbound call. When the call is complete, information related to the call is passed back to the PDS so the record in the PDS calling list is updated.

PDS integration is accessed through a set of Dialer external functions. For more information on these external functions, see [Dialer external functions](#) on page 682.

To continue, see [Using Dialer external functions](#) on page 514.

Related topics:

[Using Dialer external functions](#) on page 514

Using Dialer external functions

In this topic, you complete an application that uses Dialer external functions to integrate with the Predictive Dialing System. To simplify the process, a base application includes many of the required nodes. Your role is to add and specify the Dialer nodes for the application.

The confirm application that you create is based upon the requirement to confirm appointments for a dentist office. The Dialer calls patients with appointments scheduled for the next day. Control is transferred to Avaya IR, which announces the scheduled appointment time and gives

the patient the opportunity to confirm the appointment, cancel the appointment without rescheduling, speak to a staff member about the appointment, or set a time for a staff member to call back about the appointment. After the patient choice is handled, the application gives the patient an option to be removed from the automated reminder list.

The confirm application:

- Gets the appointment time from Dialer.
- Announces the appointment time.
- Presents a menu of choices.
- Takes action based upon the choice:
 - If the patient presses 1, replays the message.
 - If the patient presses 2, sets the status for confirmation and acknowledges the confirmation.
 - If the patient presses 3, sets the status for cancellation and acknowledges the cancellation.
 - If the patient presses 4, sets the status for transfer, plays the transfer message, transfers the call to a live agent, and confirms the success of the transfer.
 - If the patient presses 5, sets the status for callback, prompts for a callback time, sets the call back, and confirms the success of the callback.
- Prompts for continuing with automated reminders.
 - If the patient presses 1, plays a thank you message, releases the line, and returns a completion code to the Dialer.
 - If the patient presses 2, sets the completion code for no call, sets DoNotCall with PDS, plays a thank you message, releases the line, and returns a completion code to the Dialer.

An initial script is required to set up the link between Avaya IR and PDS. This script prepares the IR channel to receive calls from PDS and sets PDS to pass the value of a patient name field to the IR system when it notifies the IR system of a call.

Develop the initial script

To develop the initial pdsis script:

1. Create a new TAS application and name it pdsis.
2. Create the following custom variables:
 - A number variable named [ReturnCode]
 - A character variable named [ErrorCode]
3. Import the following Dialer external functions:
 - End_softdisc
 - Handle_disc

- SetNotifyFld
 - Set_softdisc
4. Drag a Set_softdisc  node onto the design pad and name it SetSoftDisc. This node prepares the application for a soft disconnect. It should be the first external function in any application that integrates with PDS.
 5. Edit the SetSoftDisc node as follows:

Put Return Code In: [ReturnCode]
 6. Add a SetNotifyFld  node and name it SetNotifyField. This node sets the system to pass the value of a specified field from the Dialer calling list with the call notification message.
 7. Edit the SetNotifyField node as follows:
 - Put Return Code In: [ReturnCode]
 - FieldName: NAME1
 - ErrorCode: [ErrorCode]
 8. Add a Handle_disc  node and name it HandleDisc. This node sets the label for handling a soft disconnect and marks the beginning of any required event handling code to clean up the call before ending it.
 9. Edit the HandleDisc node as follows:

Put Return Code In: [ReturnCode]
 10. Add an End_softdisc  node and name it EndSoftDisc. This node ends handling of a soft disconnect and re-executes the application for the next call.
 11. Edit the EndSoftDisc node as follows:

Put Return Code In: [ReturnCode]

This completes the initial script.

Set up the confirm application to integrate with PDS

The sconfrm application includes a partial call flow for the application. You will complete the application with the appropriate nodes for the required Dialer functions.

To complete the confirm application:

1. Open the sconfrm application and save it as confirm.
2. Review the current application.
 - Review the custom variables in the application. These variables, indicated by false in the System column, are used to pass data to and retrieve data from PDS, to announce information to the patient, and to format the callback date.
 - Review the prompts and phrases for the application. If you have proxy text-to-speech, edit the Phrase table to use text-to-speech. If you do not have proxy text-to-speech, record the phrases for the application.

- Open the node inspector so you can view the parameters for the nodes.
 - Review the Set and Test nodes that set status.
The application passes a status that indicates the choice made by the patient to PDS as a completion code. Completion codes are specific to the application. They must be consistent with the completion codes on the Dialer.
 - Note the start of the application.
The application starts with an announcement node that welcomes the patient to the automated appointment reminder line. The application is missing a number of nodes required to allow it to accept a call from PDS. You will create these nodes and then connect them to the existing call flow.
3. Drag a Set_softdisc node onto the bottom of the design pad, name it SetSoftDisc, and edit it to put the return code in the [ReturnCode] variable.
 4. Add a Set_early_hu  node to the open branch of the SetSoftDisc node, name it SetEarlyHangup, and edit it to put the return code in the [ReturnCode] variable. This node prepares the application for an early hangup. It requires the application to define an L_EARLYHANGUP label. You will do this later.
 5. Add a Set and Test node to the open branch and name it InitializeStatus.
This node initializes the value of the status variable in case someone hangs up before responding to the menu.
 6. Edit the InitializeStatus node as follows:
Assign [PDSStatus] = 20
This sets the value of the status variable to 20 if the caller hangs up before making a choice from the menu.
 7. Add a Getarg  node to the open branch, name it GetName, and edit it as follows:
 - Put Return Code In: [ReturnCode]
 - Arg. Number: 1
 - Address: [Name]
 - Length: 24
This node gets the patient name passed with the call notification and places it in the [Name] variable. The call notification passes a single variable. You also need the appointment time, so you need to retrieve it separately.
 8. Add a ReadField  node to the open branch, name it GetTime, and edit it as follows:
 - Put Return Code In: [ReturnCode]
 - Field Name: APPOINTTIME
 - Field Value: [ApptTime]
 - Error Code: [ErrorCode]
This node reads the appointment time from the appointment field in the PDS customer record and places it in the [ApptTime] variable. This completes the setup required for PDS.
 9. Drag the Welcome node to the open branch under the GetTime node.

The setup nodes are now part of the larger callflow.

10. Drag the NoCallMenu node to the bottom of the design pad.
Now you will complete the open branches under the NoCallMenu node.

Complete the NoCallMenu and call completion nodes

The NoCallMenu gives the patient the opportunity to opt out of future automated reminders. After processing the choice, the application plays a thank you message and ends. You must complete the nodes to accomplish this.

1. Drag an Inline Code node to the open branch under the ThankYouMessage node and name it LabelHangup.
This node defines the label for SetEarlyHangup node at the beginning of the application.
2. Edit the LabelHangup node as follows:
Specify L_EARLYHANGUP: as the Text for the code.
3. Drag a FinishedItem  node to the open branch under the LabelHangup node, and name it CompleteCall.
4. Edit the CompleteCall node as follows:
 - Put Return Code In: [ReturnCode]
 - Completion Code: [PDSStatus]
 - Error Code: [ErrorCode]
This node releases the customer record and inserts a call completion code with the status value into the record.
5. Add a Handle_disc  node, name it HandleDisc, and edit it to put the Return Code in [ReturnCode].
6. Add an End_softdisc  node, name it EndSoftDisc, and edit it to put the Return Code in [ReturnCode].
This completes the [1] branch. You do not include a Return node in an application that integrates with PDS because you want to keep the channel open for the next call. You must now complete the branch that is taken if the patient chooses not to receive further automated reminders.
7. Add a DoNotCall node to the open [2] branch, name it DoNotCall, and edit it as follows:
 - Put Return Code In: [ReturnCode]
 - Error Code: [ErrorCode]
This node identifies the customer record as "do not call" (DNC). All matching records on other calling lists are also marked DNC.
8. Jump to the ThankYouMessage node from the open branch under the DoNotCall node.

This completes the call completion portion of the call flow.

Review the MainMenu options

Briefly review the menu options for the MainMenu node.

- If the caller chooses 1, the application jumps back to the AnnounceAppt node to replay the appointment information.
- If the caller chooses 2, the application sets the status for confirmation, acknowledges the confirmation, and jumps to the NoCallMenu node to complete the call.
- If the caller chooses 3, the application sets the status for cancellation, acknowledges the cancellation, and jumps to the NoCallMenu node to complete the call.
- If the caller fails to make a choice, the application sets an error status, announces the error, and jumps to the NoCallMenu node to complete the call.

These branches are complete. You must now complete the nodes for the transfer and callback branches.

Complete the transfer branch

The [4] branch of the MainMenu node currently includes nodes to set the transfer status and to announce the transfer. You must add the nodes to transfer the call to a PDS agent and to complete the call.

1. Drag a MoFlashBlind  node to the open branch under the AnnounceTransfer node, name it TransferCall, and edit it as follows:
 - Put Return Code In: [ReturnCode]
 - Error Code: [ErrorCode]
This node transfers the current call and data record to another job on the Dialer. The Dialer places the call in the queue for the new job or connects the call to an available agent.
2. Drag a Set and Test node to the open branch under the TransferCall node and name it TestTransfer.
You must test to verify that the transfer to PDS succeeded.
3. Add a branch before the (default) branch and name it IfSuccess.
4. Edit the TestTransfer node as follows:

If [ErrorCode] = "M00000", branch to (IfSuccess)
5. Complete the (IfSuccess) branch with a jump to the LabelHangup node.
The LabelHangup node and the following nodes complete handling of the call.
6. Add a Set and Test node to the (default) branch, name it SetTransferFailStatus, and edit it as follows:

Assign [PDStatus] = 54
This node sets the [PDStatus] variable to the code for a failed transfer.
7. Add an Announcement node to the open branch under the SetTransferStatus node, name it AnnounceTransferFailure, and edit it use the AnnounceTransferFailure initial prompt.
8. Jump to the NoCallMenu node from the open branch.

This allows the caller to opt out of future reminder calls and then ends the call. This completes the transfer call option.

Complete the callback branch

The [5] branch currently includes nodes to set the callback status, set and format the current date, and prompt for a callback time. You must add the nodes to set the callback time and determine whether it has been set successfully.

1. Review the SetDate and FormatDate nodes.
Notice how the [CurrentDate] character variable is assigned the value of the system [Date] variable. Then that date is broken apart, using substring expressions, and reassembled into the YYYY/MM/DD format using concat expressions.
2. Review the PromptForCallbackTime node and the SetTimeVariable node.
Notice that caller input is saved in both [CollectedDigits] and in the [CallbackTime] character variable. To announce the time, the value is also assigned to the [CallbackTimeVariable] time variable.
3. Drag a SetCallback  node to the (default) branch under the SetTimeVariable node, name it SetCallbackTime, and edit it as follows:
 - Put Return Code In: [ReturnCode]
 - Callback Date: [FormattedDate]
 - Callback Time: {CallbackTime}
 - Phone Index: 1
 - Error Code: [ErrorCode]
This node sets the time on the dialer to schedule a patient callback. Before it ends the call, the application needs to confirm that PDS accepted the callback instruction.
4. Add a Set and Test node to the open branch under the SetCallbackTime node and name it TestCallback.
5. Add a branch before the (default) branch and name it IfSuccess.
6. Edit the TestCallback node as follows:

If [ErrorCode] = "M00000", branch to (IfSuccess)
7. Add an Announcement node to the (IfSuccess) branch, name it AnnounceCallback, and edit it to use the AnnounceCallback prompt.
8. Complete the (IfSuccess) branch with a jump to the NoCallMenu node.
9. Add a Set and Test node to the (default) branch, name it SetCallbackFailStatus, and edit it as follows:

Assign [PDStatus] = 55
10. Add an Announcement node to the openbranch under the SetCallbackFailStatus node, name it AnnounceCallbackFailure, and edit it to use the AnnounceCallbackFailure prompt.
11. Complete the openbranch under the AnnounceCallbackFailure node with a jump to the NoCallMenu node.
12. Complete the failure branch of the PromptForCallbackTime node with a jump to the SetCallbackFailStatus node.

This completes the confirm application. Before you can run this application, you would need to install the Inbound and DoNotCall features on the Dialer and make sure that the completion codes and calling list fields referenced in the script are valid. You would also need to administer the IR system and set up the appropriate patient records in the Dialer. For more information, see *Dialer Administration* and *Avaya PDS User's Guide*.

Testing and troubleshooting TAS applications

The first steps in testing and troubleshooting an application are to verify and simulate the application. These steps identify potential problems before you generate code and install the application.

After you install 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 and view call data records to help you troubleshoot problems.

The topics in this section provide examples of how to trace applications and view call data records.

To continue, see [Tracing a TAS application](#) on page 521.

Related topics:

[Tracing a TAS application](#) on page 521

[Viewing results with SQL*Plus](#) on page 524

Tracing a TAS application

The Avaya IR or CONVERSANT system contains a trace facility that you can use to follow the steps of an IVR Designer application while it executes.

To trace an application:

1. Install the application on the Avaya IR or CONVERSANT system and assign it to a channel.
2. Connect to the Avaya IR or CONVERSANT system.
If necessary, click the Telnet icon  on the Application Install tab of the Code Generation / Application Transfer window.
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. Check that the application is assigned to the desired channel using the following command (where ## is the channel number to which the application is assigned):
display chan ##
6. Enter the following command (where ## is the channel number to which the application is assigned):

trace ## | tee /tmp/ application name - ## .out

The tee command sends 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.

7. Dial the number for the voice channel. When prompted, respond appropriately. To stop a trace, press the Delete key on the keyboard.

The following figure is an example of trace output:

```
> cat /tmp/demo1-##.out
TRIP: CH0##: event = TR_NEWCALL, TR line = 0, 30000 msec since the last ring
CH0##: AD: dispatch_call: ENTER: cid=08054d44 chan=## dispmode=IRD_AD_STARTUP
CH0##: AD: dispatch_call: Dispatching process=TSM service=demo1 type=PERMANENT
...
CH0##: TSM: demol: STEP: 0 VALUE: "Application Name: demo1"
CH0##: TSM: demol: STEP: 0 VALUE: "Application Version: 8/31/2001 12:23:05 PM"
CH0##: TSM: demol: STEP: 0 VALUE: "Avaya Inc. <app_dev_prod_name> Version:
4.5.12"
CH0##: TSM: demol: STEP: 0 VALUE: "Code Generator Version: 4.5.12"
CH0##: TSM: demol: STEP: 0 VALUE: "Platform: GBCS"
CH0##: TSM: demol: STEP: 0 VALUE: "@(#)CVISCMDS.PAS 21.1.2.1"
...
CH0##: TSM: demol: STEP: 2 VALUE: "EnterAcctNum"
CH0##: TSM: say(1 chars): ...
CH0##: TSM: say(89 chars): \!rnormal Please enter your 5, 6, or 7 digit acco...
TRIP: CH0##: event = TR_TTREC, TR line = 0, TT = "1", seq=0, 8630 msec since
the
last TT
...
TRIP: CH0##: event = TR_TTREC, TR line = 0, TT = "2", seq=0, 340 msec since the
last TT
...
TRIP: CH0##: event = TR_TTREC, TR line = 0, TT = "3", seq=0, 340 msec since the
last TT
...
TRIP: CH0##: event = TR_TTREC, TR line = 0, TT = "4", seq=0, 380 msec since the
last TT
...
TRIP: CH0##: event = TR_TTREC, TR line = 0, TT = "5", seq=0, 380 msec since the
last TT
...
TRIP: CH0##: event = TR_TTREC, TR line = 0, TT = "#", seq=0, 470 msec since the
last TT
...
CH0##: TSM: demol: STEP: 3 VALUE: "CheckLengthOfAcctNum"
...
CH0##: TSM: demol: STEP: 4 VALUE: "CustomerAcctNumHas5Digits"
CH0##: TSM: say(1 chars): ...
CH0##: TSM: say(53 chars): \!rnormal The customer account number has 5 digit...
...
CH0##: TSM: demol: STEP: 5 VALUE: "HangUp"
...
CH0##: TSM: demol: STEP: 6 VALUE: "Quit"
...
CH0##: TSM: demol: STEP: 0 VALUE: "end:"
...
CH0##: TSM: Script Terminates
CH0##: TSM: -- Program "demol" (level 0) Ends
...
CH0##: TSM: reset chan ## (state 0)
```

Notice the following on the trace shown above.

- The user-supplied names of the IVR Designer nodes are included in the trace output.
- The TRIP process is what is happening on the Tip Ring card. This does not apply to the Avaya IR system, which has no Tip Ring cards. Notice in this

example, the digits 1, 2, 3, 4, 5, and # were being collected as the caller entered them.

- The collected digits are then evaluated by the STEP named `CheckLengthOfAcctNum`, the `CustomerAcctNumHas5Digits` branch STEP is executed, the Text-to-Speech announcement associated with the branch STEP is spoken, the `HangUp` and `Quit` STEPs are executed, and the application ends.

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, shown in the following figure, are the only lines displayed in the trace.

```
> cat /tmp/demol-24SE.out
CH024: TSM: demol: STEP: 0 VALUE: "Application Name: demol"
CH024: TSM: demol: STEP: 0 VALUE: "Application Version: 8/31/2001 12:23:05 PM"
CH024: TSM: demol: STEP: 0 VALUE: "Avaya Inc. <app_dev_prod_name> Version:
4.5.12
"
CH024: TSM: demol: STEP: 0 VALUE: "Code Generator Version: 4.5.12"
CH024: TSM: demol: STEP: 0 VALUE: "Platform: GBCS"
CH024: TSM: demol: STEP: 0 VALUE: "@(#)CVISCMDS.PAS 21.1.2.1"
CH024: TSM: demol: STEP: 1 VALUE: "AnswerThePhone"
CH024: TSM: demol: STEP: 2 VALUE: "EnterAcctNum"
CH024: TSM: demol: STEP: -2 VALUE: "12345"
CH024: TSM: demol: STEP: 3 VALUE: "CheckLengthOfAcctNum"
CH024: TSM: demol: STEP: 4 VALUE: "CustomerAcctNumHas5Digits"
CH024: TSM: demol: STEP: 5 VALUE: "HangUp"
CH024: TSM: demol: STEP: 6 VALUE: "Quit"
CH024: TSM: demol: STEP: 0 VALUE: "end:"
>
```

To stop the trace, press the Delete key on the keyboard.

Trace Data Interface Process

With the trace utility you can also trace a specific DIP, or Data Interface Process. The local ORACLE database DIP is defined as `dip1`.

1. Start a trace with the following command:
`trace dip1`
2. Dial the number for the voice channel. When prompted, respond appropriately. The resulting output should look similar to the following figure:

```

> trace dip1
DBDIPl: Got message. mcont = INSERT, morig = 4, mchan = 24
DBDIPl: INSERT INTO "CALLSRECD" ("CALL_DATE","ID_IF_RECEIVED") VALUES
(TO_DATE('
20000301','YYYYMMDD'),'123456789')
DBDIPl: Reply code = 1, chan = 24, comp_time = 0.01 (seconds)
DBDIPl: Got message. mcont = SCRIPTTERM, morig = 4, mchan = 24

DBDIPl: Got message. mcont = INSERT, morig = 4, mchan = 24
DBDIPl: INSERT INTO "CALLSRECD" ("CALL_DATE","ID_IF_RECEIVED") VALUES
(TO_DATE('
20000301','YYYYMMDD'),'12345678')
DBDIPl: Reply code = 1, chan = 24, comp_time = 0.02 (seconds)
DBDIPl: Got message. mcont = SCRIPTTERM, morig = 4, mchan = 24

DBDIPl: Got message. mcont = INSERT, morig = 4, mchan = 24
DBDIPl: INSERT INTO "CALLSRECD" ("CALL_DATE") VALUES
(TO_DATE('20000301','YYYYMM
DD'))
DBDIPl: Reply code = 1, chan = 24, comp_time = 0.01 (seconds)
DBDIPl: Got message. mcont = SCRIPTTERM, morig = 4, mchan = 24

Stopped Tracing.
Filename: demo2-trace2.txt

```

Notice the INSERT line during the tracing of dip1. This message from dip1 indicates that a record has been added or inserted into the CALLSRECD table.

To continue, see [Viewing results with SQL*Plus](#) on page 524.

Viewing results with SQL*Plus

If you want to check the table schema and data on the Avaya IR or CONVERSANT system, you can use SQL*Plus, a query language—a set of commands and syntax used to gain access to the information in an ORACLE® database. SQL*Plus enables you to directly gain access to database tables local to the Avaya IR or CONVERSANT system. Although this is not a course on SQL*Plus, here's some basic information about using this query language to view database tables.

You gain access to SQL*Plus from the system prompt. You execute SQL*Plus commands from the SQL*Plus prompt (SQL>).

Two SQL*Plus symbols to remember are:

* Means all. For instance, it could be used to specify all the fields within a particular table or all the tables within a database.

; Indicates the end of a query command. End each command with a semi-colon.

To view table schema and data on the Avaya IR or CONVERSANT system:

1. At the Solaris prompt enter the following (sti is both the user name and password on your system):
sqlplus sti/sti

The SQL> prompt appears.

2. To have SQL*Plus show all the database tables in the system, enter:
select * from TAB;
3. Find the table in the list displayed.

4. To check the schema of the table, enter:
describe TABLENAME ;

SQL*Plus shows you much the same information as the IVR Designer Table Schema window: field name, field type, and column width (i.e., length). It also shows you the contents of the table. (The Null column of the list indicates whether or not a field may be blank, that is, whether it may contain no data.)

5. To view all the data in the table, enter:
select * from TABLENAME ;
6. To get the number of records in a table, enter:
select count(*) from TABLENAME ;
7. To return to the system prompt, enter:
exit

A sample session may look similar to the following:

```
> sqlplus sti/sti
SQL*Plus: Release 8.1.5.0.0 - Production on Wed Mar 1 14:29:42 2001

Copyright (c) Oracle Corporation 1999. All rights reserved.

Connected to:
Oracle8i Enterprise Edition Release 8.1.5.5.0 - Production
PL/SQL Release 2.3.2.3.0 - Production

SQL> describe callsrecd;
  Name                Null?   Type
-----
CALL_DATE              DATE
ID_IF_RECEIVED          VARCHAR2(9)

SQL> select * from callsrecd;

CALL_DATE ID_IF_REC
-----
01-MAR-00
01-MAR-00 123456789
01-MAR-00
01-MAR-00 12345678
01-MAR-00 987654321

5 rows selected.

SQL> exit
Oracle8i Enterprise Edition Release 8.1.5.5.0 - Production
PL/SQL Release 2.3.2.3.0 - Production

Filename: callsrecd-sqlplus.txt
```

SQL*Plus interprets lowercase table and field names as uppercase, unless you enclose the name in double quotes. For example, you could type select * from tab; and it would have the same effect as if you used uppercase select * from TAB;.

This completes the TAS application development tours.

Securing user data

You can secure personal details using the privacy property feature in IVR-Designer. This feature ensures data privacy by hiding user input data such as passwords or Personal Identification Numbers (PIN) in the application traces.

You can set the privacy property feature using the **Inline Code** node to hide user announcements.

Related topics:

[Setting up privacy property in TAS applications](#) on page 526

Setting up privacy property in TAS applications

1. Perform the following tasks to enable privacy property:
 - a. In your TAS application, identify the announcement that you want to hide.
 - b. Insert an **Inline Code** node before the announcement.
 - c. In the **Name the Inline Code node** dialog box, enter a unique name for the **Inline Code** node.
IVR-Designer opens the **Node Inspector** dialog box.
 - d. In the **Text** field, click .
 - IVR-Designer opens the **Inline Code** dialog box.
 - e. In the **Text of Code** field, enter:

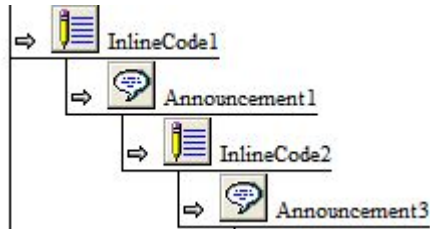
```
#include <irParams.h>
setIRAPIparam(IRP_INPUT_DISPLAY, imm.0)
setIRAPIparam(IRP_PRIVACY_ON, imm.1)
```

- f. Click **OK**.
2. Perform the following tasks to disable privacy property:
 - a. Insert an **Inline Code** node before the announcement.
 - b. In the **Name the Inline Code node** dialog box, enter a unique name for the **Inline Code** node.
IVR-Designer opens the **Node Inspector** dialog box.
 - c. In the **Text** field, click .
 - IVR-Designer opens the **Inline Code** dialog box.
 - d. In the **Text of Code** field, enter:

```
#include <irParams.h>
setIRAPIparam(IRP_INPUT_DISPLAY, imm.1)
setIRAPIparam(IRP_PRIVACY_ON, imm.0)
```

e. Click **OK**.

Example



In this example, **InlineCode1** enables data privacy for **Announcement1** and **InlineCode2** disables data privacy after **Announcement1**. In the trace logs, IVR-Designer hides the user input data for **Announcement1**. However, you can view user input data for **Announcement3**.

Chapter 12: Reference

Reference contains reference material including descriptions of menus, menu options, icons, palettes, nodes, system variables, and standard external functions.

Related topics:

[Avaya IVR Designer icons](#) on page 538

[Node Palette icons](#) on page 545

[External Function palette icons](#) on page 550

[Miscellaneous icons](#) on page 550

[Palettes](#) on page 554

[Node descriptions](#) on page 559

[System variable descriptions](#) on page 626

[Standard external functions](#) on page 645

Menus and menu options

File menu options

The following table describes the options on the File menu.

Option	Description
New	Creates a new application.
Open...	Opens an existing application.  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 on page 275.  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.

Option	Description
Undo	Undoes the last action or series of actions. Selecting Undo from the Edit menu or clicking the Undo icon on the Toolbar 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.  Note: 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.
Open...	Redoes any actions that were undone using the Undo command. Selecting Redo from the Edit menu or clicking the Redo icon on the Toolbar 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.

Cut	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.
Copy	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.
Paste	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.
Configure Toolbar...	Opens the Configure Toolbar window which lets you customize the Toolbar.
Configure Node Palette...	Opens the Configure Node Palette window which lets you customize the Node Palette.
Configure External Function Palette...	Opens the Configure External Functions Palette window which lets you customize the External Function Palette.

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.

Option	Description
Toolbar	Toggles between displaying and hiding the Toolbar. When checked, the Toolbar is displayed.
Node Palette	Toggles between displaying and hiding the Node Palette. When checked, the Node Palette is displayed.
External Functions Palette	Toggles between displaying and hiding the External Function Palette. When checked, the External Function Palette is displayed.

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.

Option	Description

Globals	Opens the Globals Manager.
Variables	Opens the Variables Manager.
Phrases	Opens the Phrases Manager.
Prompts	Opens the Prompts Manager.
Tables	Opens the Tables Manager.
External Functions	Opens the External Functions Manager.
FlexWords	Opens the FlexWords Manager.
Hosts	Opens the Hosts Manager.

Window menu options

The Window menu is available only when the Multiple Document Interface option has been selected in the Preference Editor.

When you select the Multiple Document Interface option, IVR Designer displays each call flow for an application in its own non-modal window, which means that you can have more than one call flow window open at one time and switch back and forth between them easily. The Window menu offers Windows-standard options for arranging and displaying the call flow windows.

The following table describes the options on the Window menu.

Option	Description
Cascade	Overlaps all open windows in a cascading pattern starting in the upper left corner of the Design Pad with the most recently activated windows on top.
Tile	Arranges all active windows to fill the available space on the Design Pad. Windows are tiled top-to-bottom and left-to-right, and progressing in order from the most recently active.  Tip: To arrange the active windows in a particular order, reactivate them by clicking in them one at a time in reverse order to the way you want them to appear on the Design Pad, and then click the Tile option.

Arrange Icons	Inactive in the current version.
Minimize All	Minimizes all open windows. Each window must be maximized to view the contents.
[Windows available]	Lists all currently open windows, even if minimized, in the order in which they were opened.

Tools menu options

The following table describes the options on the Tools menu.

Option	Description
Application Resources	The Application Resources tool lets you: • See what resources are in use with the current application • Import additional resources into your application as needed, either as Private Copies or Shared • See what resources in your application are being shared.
Call Flow Inventory	The Call Flow Inventory tool 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	The Code Generation tool generates the code necessary to transfer and

	install the application onto the target system. The Application Transfer tool transfers the application files to the target system.
Events String Editor	The Events String Editor tool lets you: • Determine which variables will be treated as Call Data Events on the target system. • Display both a list of all the variables defined for the application and a list of the variables selected to be treated as Call Data Events.
Node Inspector	The Node Inspector tools 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	The Preference Editor lets you: • Set application and toolwide preferences for using IVR Designer. • Control the working environment for IVR Designer
Screen Mapping Tool	The Screen Mapping Tool lets you: • Capture screens from an IBM Host. • Transfer the screen captures back to the PC. • Display the screens and map the field positions to the HostEdit display within IVR Designer.

Simulation	The Simulation tool lets you: • Run your application in a simulated environment. • Find problems within the application, thus reducing the amount of time required to test your application on the target system.
Verify Design	The Verify Design tool lets you: • Check your application design for possible errors or omissions. • View each call flow's results in a separate display. • Easily locate nodes with possible errors or omissions.

Sharing menu options

The following table describes the options on the Sharing menu.

Option	Description
Refresh From Shared Resources	Allows you to update local copies of resources imported to your application.
Update Shared Resources	Allows the owner of a shared resource to update the resource.

Help menu options

The following table describes the options on the Help menu.

Option	Description
Contents...	Opens the IVR Designer Help Topics. From there, you can navigate to topics of interest.

Search for Help on...	Opens the online Help index. To find a help topic, type the first few letters of the topic you are looking for, then click Display.
About...	Displays the About Avaya IVR Designer information box, which contains a copyright notice and the Avaya IVR Designer version number.

Avaya IVR Designer icons

Avaya IVR Designer icons provides a visual reference guide for icons used in IVR Designer, and 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 three 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.
- The External Functions Palette contains the icons for any external functions defined for the application.

In addition, several of the development tools and editors have their own sets of icons.

Related topics:

[Toolbar icons](#) on page 538

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.

Related topics:

[File Menu icons](#) on page 539

[Edit Menu icons](#) on page 539

[View Menu icons](#) on page 540

[Tools Menu icons](#) on page 541

[Sharing Menu icons](#) on page 544

[Help Menu icon](#) on page 545

File Menu icons

The following table describes the File menu icons.

Icon	Related menu option	Action
	New	Creates a new application.
	Open	Opens an existing application.
	Close	Closes the current application.
	Save	Saves the current application to disk.
	Delete	Deletes the current application.
	Print	Prints the application.
	Exit	Closes any open application and exits Avaya IVR Designer.

Edit Menu icons

The following table describes the Edit menu icons.

Icon	Related menu option	Action
	Undo	Reverses (undoes) the last action or series of actions since the last Save action.
	Redo	Reverses (redoes) any actions performed by the Undo option, one at a time.
	Cut	Removes selected node(s) from the call flow

		and places them on the clipboard.
	Copy	Places a copy of selected node(s) on the clipboard.
	Paste	Pastes the nodes from the clipboard onto any open branch.
	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 icons

The following table describes the View menu icons.

Icon	Related menu option	Action
	Globals	Opens the Globals Manager and all associated resource manager.
	Variables	Opens the Variables Manager and the Globals Manager, if it is docked there.
	Phrases	Opens the Phrases Manager and the Globals Manager, if it is docked there.
	Prompts	Opens the Prompts Manager and the Globals

		Manager, if it is docked there.
	Tables	Opens the Tables Manager and the Globals Manager, if it is docked there.
	External Functions	Opens the External Functions Manager and the Globals Manager, if it is docked there.
	FlexWords	Opens the FlexWords Manager and the Globals Manager, if it is docked there.
	Hosts	Opens the Hosts Manager and the Globals Manager, if it is docked there.

Tools Menu icons

The following table describes the Tools menu icons.

Icon	Related menu option	Action
	Application Resources Tool	Opens the Application Resources window, which lets you: • See what resources are in use with the current application • Import additional resources into your application as needed • See what resources in your application

		are being shared.
	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	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.
	Screen Mapping Tool	Opens the Screen Mapping Tool window, which lets you: • Telnet to a CONVERSANT

		or Avaya IR machine and capture screens from an IBM Host. • Open an FTP connection from IVR Designer to a CONVERSANT or Avaya IR machine and download the captured files. • Display the screens and map the field positions to the HostEdit display within IVR Designer.
	Events String Editor	Opens the Events String Editor, which lets you: • Determine which variables will be treated as Call Data Events on the target system • Display both a list of all the variables defined for the application and a list of the variables selected to be treated as Call Data Events.

	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	Opens the Preference Editor, which lets you set global preferences for using IVR Designer.
	Simulation Tool	Runs the Simulation tool, which tests your application in a simulated environment, which is often helpful in finding problems within the application.
	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.

Sharing Menu icons

The following table describes the Sharing menu icons.

Icon	Related menu option	Action

	Refresh from Shared Resources	Refreshes your shared-in resources, to ensure that you are working with the most recent version of the shared resource.
	Update Shared Resources	Updates your shared-out resources, to make the most recent version of the shared resource available to any developer who might be using it.

Help Menu icon

The following table describes the Help menu icon.

Icon	Related menu option	Action
	Help Contents...	Opens the online Help.

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.

Related topics:

[Control node icons](#) on page 545

[Telephony Node icons](#) on page 546

[Miscellaneous Node icons](#) on page 547

[IVR \(Interactive Voice-Response\) Node icons](#) on page 548

[Database Node icons](#) on page 549

[Host Node icons](#) on page 549

Control node icons

The following table describes the Control Node icons.

Icon	Node name	Action
	Call Flow node	Creates a new call flow or encapsulates a set of nodes.
	Return node	Stops the execution of the call flow that contains it. In the [Main] call flow, a Return node disconnects the call, if it has not already been disconnected. In a call flow other than the [Main] call flow, a Return node returns control to the branch of the call flow that has the same return value string.
	Set and Test node	Manipulates and evaluates a wide range of variable expressions and branch conditions.
	Time Branch node	Directs the call based on time of day, day of the week, or specific date.
	Asynchronous Event node	Defines the subroutine to be executed if an asynchronous event, such as a caller hanging up or an error, is detected.

Telephony Node icons

The following table describes the Telephony Node icons.

Icon	Node name	Action
	Answer Call node	Answers the incoming call; typically one of the first nodes of any application.

	Transfer Call node	Transfers a call to another number.
	Disconnect Call node	Disconnects the incoming call and executes the next node in the call flow (if one exists).

Miscellaneous Node icons

The following table describes the Miscellaneous Node icons.

Icon	Node name	Action
	Inline Code node	Lets you to write TAS code directly in your IVR Designer application.
	Trace Variables node	Traces values of selected variables on the target system; useful primarily as a debugging tool after installing your application on the target system.
	Save & Exec node	The Save and Exec node calls another application on the target system. By setting the attribute 'return control' in the Node Inspector to true, control will be returned to the calling application. If 'return control' is set to false, the pass branch will not appear. This node is typically used in conjunction with the Load Variables node. The Save and Exec node is similar but not identical to the Execute and Subprogram external functions. The differences lie in the way

		the two pass arguments and variables to the new application. When calling a Script Builder application, you must use the Execute or Subprogram external function because Script Builder does not have Load Variables node functionality or any equivalent action. For IVR Designer applications, you can use either the Save and Exec node, the Execute or Subprogram external functions.
	Load Variables node	Loads variable data passed to it from another IVR Designer application into the current application; typically used in conjunction with the Save & Exec node and is usually the first node in the current application.

IVR (Interactive Voice-Response) Node icons

The following table describes the IVR Node icons.

Icon	Node name	Action
	Announcement node	Plays a simple announcement to the caller.
	Menu node	Presents a list of options from which the caller can choose.
	Prompt and Collect node	Prompts a caller for input such as account or credit card numbers.

	Automenu node	Creates a dynamic menu of choices derived from a list of previously retrieved database records.
	Voice Capture node	Records a caller's spoken response.

Database Node icons

The following table describes the Database Node icons.

Icon	Node name	Action
	Query Table node	Retrieves records from a database table, based upon the specified selection criteria.
	Update Table node	Changes data within a specified database table.
	Add Record node	Adds a record to a database table during application execution.
	Delete Record node	Deletes a record from a database table during application execution.

Host Node icons

The following table describes the Host Node icons.

Icon	Node name	Action
	Host Connection node	Determines the status of the connection with a host machine in the network.
	Host Send node	Sends data to a remote host machine in the network.

	Host Send/Receive node	Sends data to and receives data from a remote host machine in the network.

External Function palette icons

The External Functions Palette contains the icons for any external functions you may use in your application. These external functions can be imported using the Application Resources tool or created within your IVR Designer application. The External Functions Palette is blank until you import or create an external function for your application.

Similar to the standard palette configuration options used for both the Toolbar and the Node Palette, external functions can be grouped together into user-defined categories and represented by customized bitmap images on the External Functions Palette.

IVR Designer provides an icon for each of the standard external functions packaged with the software.

IVR Designer also provides the following standard bitmap image that you can use with your own custom external functions.



To use other (customized) bitmaps, you can:

- Create your own, using any graphics software that supports the *.bmp file format.
- Use an existing bitmap image.

For information about creating and using customized bitmap images, see [The External Function Editor](#) on page 141. For more information about these capabilities and about External Functions in general, see [Working with external functions](#) on page 136.

Miscellaneous icons

In addition to the main palettes, several of the tools in IVR Designer have their own palettes and corresponding icons. The tools that have their own icon palettes are:

- The Code Generation/Application Transfer Tool
- The Events String Editor
- The Simulation Tool

Related topics:

[Code Generation/Application Transfer tool icons](#) on page 551

[Events String Editor icons](#) on page 552

[Simulation tool icons](#) on page 552

Code Generation/Application Transfer tool icons

The following table describes the Code Generation/Application Transfer Tool icons.

Icon	Icon/function name	Action
	Database Table Provisioning	(Available only on the Code Generation tab) Used to set table options for Code Generation and export during Application Transfer.
	Refresh from Shared Resources	(Available only on the Code Generation tab) Refreshes your shared-in resources, to ensure that you are working with the most recent version of the shared resource.
	Display Files on Remote System or Floppy	(Available only on the Application Transfer tab) Displays the files in the destination area before transferring your application.
	External Function Source Code	(Available only on the Code Generation tab) Lets you create an external function for the target system, by writing or editing TAS code from within IVR Designer.
	Telnet	(Available only on the Application Transfer tab) Establishes a telnet connection without exiting IVR Designer.

Events String Editor icons

The following table describes the Events String Editor icons.

Icon	Icon/function name	Action
	Append Variable	Adds a variable to the end of the Selected Variables list.
	Add Variable Before...	Inserts a variable before a variable on the Selected Variables list.
	Add Variable After...	Inserts a variable after a variable on the Selected Variables list.
	Delete Variable	Deselects a single variable from the Selected Variables list.
	Delete All Variables	Deselects (clears) all variables from the Selected Variables list.
	Move Variable Up	Moves a variable up on the Selected Variables list.
	Move Variable Down	Moves a variable down on the Selected Variables list.

Simulation tool icons

The following table describes the Simulation Tool icons.

Icon	Icon/function name	Action
	Stop Simulation	Stops a simulation.
	Pause Simulation	Pauses a simulation.

	Run Simulation	Runs a simulation from the currently active node or from where it was stopped or paused.
	Reset Variables	Resets the variables to their defaults; for example, to run the simulation again.
	Step-Through Simulation	Steps through a simulation one node at a time.
	Respond with Speech	Simulates spoken responses for the FlexWord speech recognition feature.
	Generate Asynchronous Event	Simulates an asynchronous event, such as a customer hanging up in the middle of an application.
	Force Timeout	Skips the countdown, or delay, that is programmed into many nodes, and go to the next node.
	Dial Pulse Recognition	Simulates a Dial Pulse Recognition (DPR) response from a caller.

Palettes

IVR Designer has three 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 External Function Palette – contains external function icons that you drag and drop onto the Design Pad to add special or customized functions to 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.

Related topics:

[Opening a Palette Configuration window](#) on page 554

[Rearranging icons on a palette](#) on page 555

[Using the Allow Scrolling option](#) on page 556

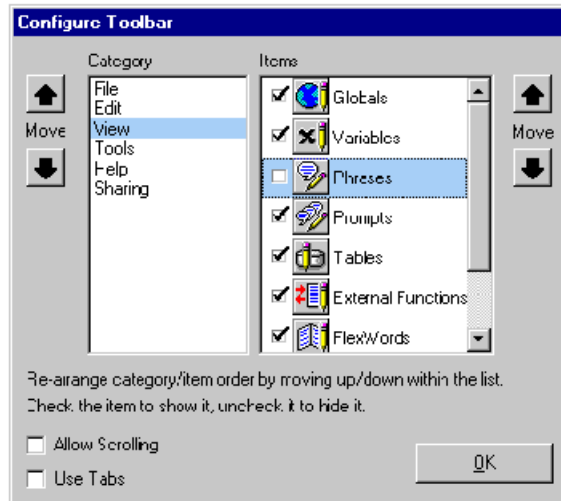
[Using the Tabs option](#) on page 556

[Docking/undocking palettes](#) on page 557

[Right-click menu options for palettes](#) on page 558

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 figure 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.



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.

Option	Description
Configure...	This option opens the palette configuration window for the selected palette.
Hide	This option hides the selected palette from view. To view the palette again, you must select the appropriate palette from the View menu.
Show Hints	This option toggles on and off. When checked (toggled on), it displays balloon-type help for the palette elements.
Allow Scrolling	This option toggles on and off. When checked (toggled on), it turns on the Allow Scrolling option.
Use Tabs	This option toggles on and off. When checked (toggled on), it turns on the Use Tabs option.

Node descriptions

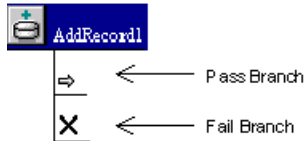
Nodes are the basic building blocks used to create an application. IVR Designer offers 25 different types of nodes, each with its own attributes and behaviors. You can use the Node Inspector to view and edit the attributes and elements of a node.

Related topics:

- [Add Record node](#) on page 559
- [Announcement node](#) on page 561
- [Answer Call node](#) on page 562
- [Asynchronous Event node](#) on page 562
- [Automenu node](#) on page 563
- [Call Flow node](#) on page 570
- [Delete Record node](#) on page 573
- [Disconnect Call node](#) on page 574
- [Host Connection node](#) on page 574
- [Host Send node](#) on page 576
- [Host Send/Receive node](#) on page 577
- [Inline Code node](#) on page 579
- [Load Variables node](#) on page 580
- [Menu node](#) on page 581
- [Prompt and Collect node](#) on page 588
- [Query Table node](#) on page 598
- [Return node](#) on page 601
- [Save and Exec node](#) on page 601
- [Set and Test node](#) on page 603
- [TAS2VXI node](#) on page 612
- [Time Branch node](#) on page 614
- [Trace Variables node](#) on page 615
- [Transfer Call node](#) on page 617
- [Update Table node](#) on page 621
- [Voice Capture node](#) on page 623

Add Record node

The Add Record node adds a record to a database table during application execution. The following figure shows the Add Record node.



Example

A customer places an order with ABC Automotive. Once the order information is obtained from the caller, the corresponding order information can be added to a local database table by using the Add Record node.

Behavior

- If a record is added to the database table, the Pass branch is taken.
- If a record cannot be added (for example, because it would create a duplicate record for a field defined as a key field), the Fail branch is taken.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Table to add record	From the drop-down menu, select the database table to which the record is to be added.
Assignments to new record	This attribute displays values to be assigned to the new record. When the  button is clicked, IVR Designer displays the Add Record Editor window. Use this editor to populate the fields of the record with values or variables.

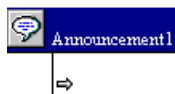
Add Record Editor

The following table describes the columns in the Add Record Editor.

Column	Description
FieldName	This column displays the names of all defined fields for the selected database table.
Add with value	From the drop-down menu, select the variable containing data to add to the field, or enter a literal value.

Announcement node

The Announcement node plays an announcement to the caller. The following figure shows the Announcement node.



Example

ABC Automotive has a customer hotline. The caller who reaches the hotline is greeted with the following announcement:

"Thank you for calling the ABC Automotive Customer Service hotline."

Behavior

- When executed, a simple announcement is played to the caller.
- No data is collected from the caller.

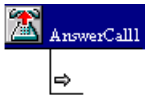
Attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Initial Prompt	From the drop-down menu, select the prompt you want to use as the announcement, or double-click to create a new one.
Allow Interrupt	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.
Discard Previous Touchtone	To ignore any touchtones the caller may have entered before the Initial Prompt is played, select True from the drop-down menu. To process any touchtones the caller may have entered before the Initial Prompt is played, select False from the drop-down menu. This allows you to take advantage of the Type Ahead feature, which works only with touchtones.

Answer Call node

The Answer Call node answers the incoming call. This is typically one of the first nodes of any application. The following figure shows the Answer Call node.



Example

A customer calls the ABC Automotive Inquiry Line. The system detects the incoming call and answers it.

Behavior

- The Answer Call node answers incoming calls and passes them along to the next node.
- The application will work in Simulation mode without an Answer Call node. When the application is downloaded to the target system, however, no incoming calls are answered without the code that is generated from the Answer Call node.



Note:

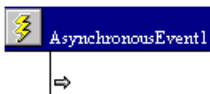
If you have not used an Answer Call node in your application, the Verify Design tool, when used, will notify you that it is missing.

Attributes

There are no user-configurable attributes for the Answer Call node.

Asynchronous Event node

The Asynchronous Event node defines the call flow to be executed if an asynchronous event, such as a caller hanging up, is detected. The following illustration shows the Asynchronous Event node.



For more information about the Asynchronous Event node, see [Using asynchronous events in TAS applications](#) on page 451.

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:

- Clear a host screen
- Reset the host to the transaction screen
- Increment a record counter variable
- Terminate the application so it is ready for a new caller.

Behavior

- The Asynchronous Event node specifies the call flow that is executed when an Asynchronous Event is detected.



Note:

The only Asynchronous Event currently supported is Inbound Call Termination.

- When the system detects an asynchronous event, the following sequence occurs. The system:
 - a. Drops the call
 - b. Jumps to the defined call flow
 - c. Executes the call flow
 - d. Terminates application processing of the call

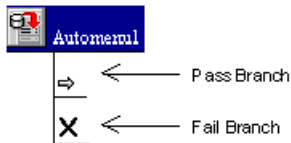
Attributes

Use the Node Inspector to assign and edit the following node attribute.

Attribute	Description
Inbound Call Terminated	From the drop-down menu, select the name of the call flow to be executed, or the [none] option. Note: Using the example above, once the caller has reached the point in the application where a hangup might be expected, then you would use the [none] option to negate a previously set asynchronous event routine.

Automenu node

The Automenu node dynamically creates a menu of choices derived from a list of previously retrieved database records. The following figure shows the Automenu node.



Example

A customer calling the ABC Automotive Inquiry Line can request to hear a list of this week's sale items. Since the items that are on sale change weekly, the information can be stored in a database table. Items in the database table can be marked as sale items and the sale price of the item can be updated. Only the data in the database table needs to be updated. The call flow information does not have to change.

Upon requesting the list of sale items, the customer hears the following:

"The following items are on sale during the week of December 12, 1997.

For information regarding Super Charger Batteries, press 1.

For information regarding Grason Spark Plugs, press 2.

For information regarding Generex Oil, press 3.

For information regarding APEX Windshield Wiper Fluid, press 4.

Please make your selection now."

Behavior

- The Automenu node plays the Initial Prompt as specified in the Node Inspector.
- The Automenu node dynamically creates a menu of choices for the caller, based upon a list of previously retrieved database records.
- The Automenu 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.
- If the caller enters an invalid response and the Tries Available field value has been exceeded, the Fail branch is executed.
- 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.
- If the caller does not respond within the time set in the First Digit Timeout field, the Timeout Prompt is played. If multiple digits are being collected from the caller and the caller does not enter a subsequent digit within the time set in the Inter-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 with enough digits, the Too Few Digits Prompt is played. If the Replay Initial Prompt field is set to True, the Initial Prompt also plays.
- Valid responses from the caller are saved in the appropriate variable.
- If a system failure occurs, for example if a recognition resource cannot be allocated, the Fail branch is executed immediately. A system failure could occur before the caller has an opportunity to respond to the initial prompt. A system failure can be distinguished from caller response errors by checking whether the Tries Available field

value has been met. If the Tries Available field value has been met, the failure is due to caller response errors. If the Tries Available field value has not been met, the failure is due to a system failure.

Inserting branches

When you insert an Automenu node, IVR Designer creates two default branches which cannot be renamed or deleted. You can, however, insert additional branches, which can be renamed later.

To insert an additional branch in an Automenu node:

1. Right-click where you want the new branch.
2. Select Insert Branch from the menu.
IVR Designer displays the Touchtones dialog box.
3. Do one of the following:
 - Enter the digit (or digits) you want to use to label the branch.
 - To define the branch as a timeout branch, click the  icon.
 - To define the branch using a FlexWord grammar, click the  icon. Clicking this icon opens a Grammar Word List window. This window displays the words defined for the FlexWord grammar specified in the Input ASR Mode attribute for this node.
 - To define the branch to use pattern matching, click the  icon. See [Pattern matching](#) on page 568 for more information.
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 use to label the branch.
 - To rename the branch as a timeout branch, click the  icon.
 - To rename the branch using a FlexWord grammar, click the  icon. Clicking this icon opens a Grammar Word List window. This window displays the words defined for the FlexWord grammar specified in the Input ASR Mode attribute for this node.
 - To rename the branch to use pattern matching, click the  icon. See [Pattern matching](#) on page 568 for information about pattern matching.
4. Click OK.

Related topics:

[Automenu node attributes](#) on page 566

[Pattern matching](#) on page 568

Automenu node attributes

Use the Node Inspector to assign and edit Automenu node attributes.

Prompts tab

Group: No group is associated with this attribute.

Attribute	Description
Initial Prompt	From the drop-down menu, select the prompt you want to play when the Automenu node begins to execute, or double-click to create a new prompt.

Group: Error Prompts

Attribute	Description
Bad Input Prompt	From the drop-down menu, select the prompt to play when the caller enters invalid input, such as an invalid account number.
Timeout Prompt	From the drop-down menu, select the prompt to play when the caller waits too long to respond to the Initial Prompt.
Too Few Digits Prompt	From the drop-down menu, select the prompt to play when the caller does not enter enough digits.

Response tab

Group: Input Parameters

Attribute	Description
Allow Interrupt	To allow the caller to interrupt while prompts are playing, select True from the drop-down menu. To play the prompts without interruption, select False from the drop-down menu.  Note: You can use this option only if the Input ASR Mode option is set to Touchtones or a speech recognition grammar.
Discard Previous Touchtone	To ignore any touchtones the caller may have entered before the Initial Prompt is played, select True from the drop-down menu. To process any touchtones the caller may have entered before the Initial Prompt is played, select False from the drop-down menu. This lets you to take advantage of the Type Ahead feature, which works only with touchtones.

Group: Input DPR Parameters

Attribute	Description
Input DPR Mode	From the drop-down menu, select the Dial Pulse Recognition (DPR) grammar you want to use to collect input from the caller.  Note: DPR grammars are not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Group: Input ASR Parameters

Attribute	Description
Input ASR Mode	From the drop-down menu, select whether you want to use Touchtones (default) or FlexWord speech recognition as input from the caller.
Mode Chosen	From the drop-down menu, select the variable that holds the value indicating how the caller responded to the Initial Prompt.  Note: If FlexWord speech recognition is chosen here, and the caller enters a response using Touchtones, the value returned to the variable is a negative number. Otherwise, the value returned is a positive number.
Reserve ASR/DPR Resource Before Playing Prompt	(Not available if Touchtone is selected as the Input Mode) To make sure that the ASR or DPR resource is available before obtaining spoken or dial pulse caller input, select True.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.
Free Reserved ASR/DPR Resource After Caller Response	(Not available if Touchtone is selected as the Input Mode) To free the ASR or DPR resource as soon as spoken or dial pulse input is retrieved from the caller, select True.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Group: Timeouts

Attribute	Description
First Digit Timeout	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.

Attribute	Description
Inter-Digit Timeout	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.

Group: Tries

Attribute	Description
Tries Available	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 3.
Tries Taken	From the drop-down menu, select a variable to store the number of tries it took a caller to enter a correct response.
Replay Initial Prompt	If you want the Initial Prompt to replay each time the caller enters an invalid response, select True.

Group: Caller's Response

Attribute	Description
Save Response In	From the drop-down menu, select the variable in which to save the caller's unformatted response.

Group: Data Collection

Attribute	Description
Track Caller's Response	(Not available if tracking for all Automenu nodes is turned on in the Events String Editor.) To track the caller's response for collection in Call Data Handling reports, select True from the drop-down menu.

Related topics:

[DPR Barge-in](#) on page 99

Pattern matching

Special Characters for Pattern Matching

Use the special characters when you want to distinguish between correct and incorrect caller input (in addition to between valid and invalid input). It also allows you to specify an action to take for each case in the list.

Creating Custom Cases

You can specify up to 12 custom cases of correct input. The system compares the input to each case and performs the action specified for the first true case, similar to an "If- Then- Else" statement. For example, your prompt could be:

"For your checking account balance, say or enter 1.

For your savings account balance, say or enter 2.

To speak with a customer service representative, say or enter 0."

For this example, list "1," "2," and "0" as three custom cases. For the digits 1 and 2, branch off to the appropriate nodes to provide the information requested. For the digit 0, specify a telephone number for an agent.

In many cases, you can describe a correct input but you cannot list all the correct values. For example, valid customer account numbers could be eight digits with the first digit always 4 and the last digit always 9, or a stock number might be six to nine digits in length and always begin with 76.

You can use the pattern symbols and special codes available in IVR Designer to describe a correct input case without specifying a value. Because the cases are compared in the order listed, include the most specific pattern first and more general patterns later.

Pattern Symbols

Character	Function
0, 1, 2,...9	These are the standard touchtone digits.
*, #	These are special touchtone keys, which are frequently used as touchtone control characters (terminator, erase, repeat, or cancel codes).

Custom Special Codes

Character	Function
A, B, C, D	These are extended digits. The touchtone standard includes a 16- digit keypad with a fourth column.
n	This represents any one of the 10 standard digits.
s	This represents either one of the two special digits
e	This represents any one of the four extended digits.
t	This represents any 1 of the 16 touchtone digits.
r	This is the repeat character. This special pattern symbol means "match one or more repetitions of the previous pattern symbol."

q	This is the quit character. This special pattern symbol means "stop comparing and accept any input string that has matched up to this point."

 Note:

If you use one of these characters or any other touchtones besides * and # as special codes, they will only have their special function and will not appear in the input. Therefore, checklist cases that include them will not be matched.

The literal characters (0 through 9, *, #, A through D) each matches one instance. The group characters (n, s, e, t) each matches one instance from its group.

The above example of an eight-digit account number beginning with 4 and ending with 9 matches the case "4nnnnnn9."

The two special pattern symbols "r" and "q" are useful when the input you want to match has a varying length (within a specified range). Use the "r" symbol when the input has a varying length portion that includes some repeated digits. Therefore, the symbol that precedes "r" in the pattern must be present one or more times. For example, the "5r2" would match any of the following inputs: "52," "555552," "552," and so on.

Use the "q" symbol when the first few digits of the input must follow a specified pattern, but the remainder of the input is more variable. For example, "76q" means that if the first part of the input matches the pattern, the remainder of the input can be ignored and the pattern is matched.

The "q" must be the last symbol in a pattern description. IVR Designer will ignore any characters after the "q" in a pattern.

You may use the special codes and pattern symbols together. For example, the "nr" would match either of the following inputs if # is a control character: "1#," "123#". However, the input "12* 45#" would be invalid for "nr" because the asterisk is not a standard digit. For "tr# 1q", the following inputs are valid: "124**# 1", 9# 1, "### 1". However, the input "123" or "# 1" is not valid. The first invalid case "123" does not provide the pound sign (#) or termination digit of 1 and the second invalid case "# 1" does not provide a touchtone digit as indicated by "t".

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.

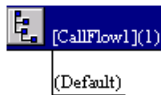
You can also use the Call Flow node to encapsulate a portion of a call flow and make a new, separate call flow out of it. The new call flow can then be viewed in its own call flow window on the design pad.

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.



 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 93.

For practice using the Call Flow node, see [Using embedded call flows in TAS applications](#) on page 400.

Behavior

- When this node is reached, the encapsulated call flow is executed.
- Like a function in a high-level language, the Call Flow node returns a value. Each return value is represented by a Return node in the call flow definition and by a branch on an instance of the call flow.

- 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.

Inserting and renaming branches

When you insert a Call Flow node in a call flow, it creates one branch named (by default) Default. This branch cannot be renamed or deleted.

When you select Insert Branch or Rename Branch on a Call Flow node, IVR Designer displays the Name Exit Condition dialog box.

Type the name you want for the branch, then click OK.

When naming branches, keep the following in mind:

- Names can be any length. You should, however, try to keep branch names as short as possible.
- Do not use spaces in names. IVR Designer ignores and deletes spaces when creating the branch name.



Note:

You may want to use an underscore (_) symbol in place of a space.

- Branch names are case sensitive. IVR Designer preserves cases as you assign them when naming the branch. 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_Branch as the branch name, IVR Designer converts it to New_Branch when creating the branch.



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. If, for example, you have a branch called New_Branch, you cannot also create a branch called NEW_BRANCH.

Attributes

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 appended with (2), and so on.

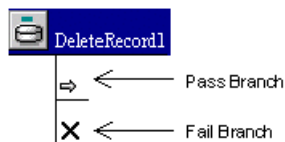
To locate and edit the call flow in the encapsulated node, click the node to highlight it, and then select Edit Flow from the menu. IVR Designer displays the encapsulated call flow on the design pad.

Use the Node Inspector to assign and edit the following node attribute.

Attribute	Description
Hide Execution	When set to True, this attribute hides the execution of the encapsulated node during Simulation.

Delete Record node

The Delete Record node deletes a record from a database table during application execution. The following figure shows the Delete Record node.



Example

A customer wants to cancel an existing order. Once the order information is obtained from the caller, the information is used as selection criteria. If a match is found, the order can be deleted from the local database table.

Behavior

- Records are selected from the database table based upon the selection criteria. If no selection criteria are specified, all records in the database table are selected.
- If one or more records are deleted from the database table, the Pass branch is taken.
- If multiple records are found and the Delete multiple records field is set to False, no records are deleted and the Fail branch is taken.
- If a record cannot be deleted (for example, because the database table resource is not available), the Fail branch is taken.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Table to search	From the drop-down menu, select the database table from which the record is to be deleted.
Selection criteria	Displays selection criteria, if any, to be used in deleting the records. When the  button is clicked, IVR Designer displays the Delete Record Editor window. Use this editor to define selection criteria for the fields of the record with values or variables.
Delete multiple records	When set to True, this option deletes all records that match the selection criteria.

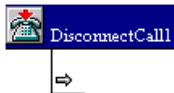
Delete Record Editor

The following table describes the columns in the Delete Record Editor.

Column	Description
FieldName	This column displays the names of all defined fields for the selected database table.
Relation	This column allows you to select and displays the relation operator for the selected variable field. Possible value: =
Variable/Literal	From the drop-down menu, select the variable containing data to delete from the database, or enter a literal value.

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.



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

The Disconnect node:

- Disconnects incoming calls and executes the next node
- Terminates the call flow at specific user-defined points
- Is not required to successfully terminate the dialog with the caller

Attributes

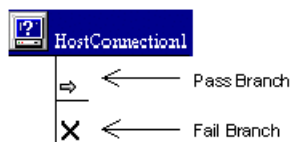
There are no user-configurable attributes for the Disconnect Call node.

Host Connection node

The Host Connection node determines the status of the connection with a host machine and allocates a logical unit (LU) for the call.

This node is typically one of the first ones in an application using remote hosts. The Host Connection node should be placed before the Answer Call node because host connectivity

should be checked before accepting incoming calls. Otherwise, the caller will hear silence while the connection is being checked. The following figure shows the Host Connection node.



Example

Your application needs to download account information from a remote host for a customer on the line. Before you can get account information from the host, however, you need to verify that the host is connected and that an LU has been allocated for the call. If the host is not connected, it may take several seconds to establish a connection. Meanwhile, the customer is on the line waiting to find out the status of the account. You can use the Host Connection node to verify that the host is connected before the call is answered.

Behavior

- If the connection is established within the allotted time, the Pass branch is taken.
- If the connection cannot be established within the allotted time, the Fail branch is executed.
- When a connection with the host has been established, an LU is assigned to the application. The LU remains assigned to the application until it is released or the call is terminated. If an LU is not available, the Fail branch is executed.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Host Name	Enter the name of the host, or select the name of the host from the drop-down menu.  Note: The host must be defined in the Host Definition Editor before you can use a Host Connection node.
Connection Timeout	Enter the number of seconds to allow for the connection to be established.  Note: The value of this field should be greater than or equal to the value defined for the Logical Unit Timeout on the Host Definition tab of the Host Definition Editor.

Host Send node

The Host Send node sends data to a host machine. The following figure shows the Host Send node.



Example

Your application needs to upload customer account information to a remote host. Before you can send the account information to the host, a login sequence must be established for security purposes. The Host Send node can be used to send the necessary login and password information, as well as the account information.

Behavior

- When taken, the Host Send node sends data to the designated host machine. The transmission of data is considered a blind send because the application does not know whether the host received the data. The host does not confirm the transmission.
- Once the data is transmitted, the default exit branch is taken.

Attributes

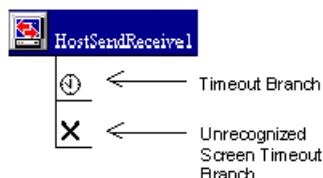
Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Host Name	Enter the name of the host, or select the name of the host from the drop-down menu.  Note: The host must be defined in the Host Definition Editor before you can use a Host Connection node.
Release Logical Unit	When set to True, this attribute releases the LU after the data is transmitted.
Send Screen Name	From the drop-down menu, select the name of the screen to send to the host. The screen names available are based on the host definition.
Send Field Mappings	Click the  button to open the Host Screen Field Mappings window. From this window, select which fields of the selected screen to send to the host.

	 Note: This option is active only if you have set the Direction field on the Field Definition tab of the Host Definition Editor to To Host or Both.
Aid Key	(Optional) From the drop-down menu, select an Aid Key to send to the host with the mapped fields.

Host Send/Receive node

The Host Send/Receive node both sends data to and receives data from a host machine in the network. The following figure shows the Host Send/Receive node.



Example

A customer wants account information that resides on a remote host. Your application sends the caller's account number and Personal Identification Number (PIN) to the host and gets the account balance from the host in return.

Behavior

- If no response from the host is received within the specified time, the Timeout branch is taken.
- The application executes the Unrecognized Screen Timeout branch when the Receive Screen data sent from the host is not the correct screen (that is, when the data sent from the host does not match the format defined on the Host Send/Receive screen), and the correct format was not sent within the time allotted in the host definition.

Inserting branches

When you insert a branch on a Host Send/Receive node, you select which screen you want to receive from the host.

To insert a branch on a Host Send/Receive node:

1. Click to select the desired node.
2. In Host Name field on the Node Inspector, use the dropdown menu to select the host which has the screen you want.

 Note:

Until you select a host, the other fields in the Node Inspector are inactive (grayed out), as are several right-click menu items for this node. As soon as you select a host, the Node Inspector fields become active, as do the menu items.

3. Right-click the branch where you want the new branch to be inserted, and select Insert Branch from the popup menu.

 Note:

When the new branch is inserted, it moves the selected branch down.

IVR Designer displays the Receive Screens for Host [HostName] window.

4. Click to select the host screen you want to receive and then click OK.
IVR Designer inserts a new branch and assigns it the name of the screen you selected.
5. If you want to receive field data from the host for this screen, map the desired fields to variables.

Renaming a branch

To rename a branch on a Host Send/Receive node:

1. Right-click the branch you want to rename.
IVR Designer displays the Receive Screens for Host [HostName] window.
2. Click to select the new host screen for which you want to receive field mappings and then click OK.
IVR Designer replaces the old branch name with the name of the screen you selected.

 Note:

When you rename the branch, you also undo any screen field mappings that may exist. You will need to redo the screen field mappings.

3. If you want to receive field data from the host for this screen, map the desired fields to variables.

Attributes

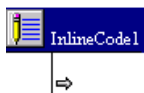
Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Host Name	Enter the name of the host, or select the name of the host from the drop-down menu.  Note: The host must be defined in the Host Definition Editor before you can use a Host Connection node.

Release Logical Unit	When set to True, this attribute releases the LU after the data is transmitted.
Send Screen Name	From the drop-down menu, select the name of the screen to send to the host. The screen names available are based on the host definition.
Send Field Mappings	Click the  button to open the Host Screen Field Mappings window. From this window, select which fields of the selected screen to send to the host.  Note: This option is active only if you have set the Direction field on the Field Definition tab of the Host Definition Editor to To Host or Both, and a new branch has been inserted in the appropriate node.
Aid Key	From the drop-down menu, select an Aid Key to send to the host with the mapped fields.

Inline Code node

The Inline Code node lets you write TAS code directly in your IVR Designer application. When the code is generated and the application is transferred, the TAS code is inserted at the point where this node is placed in the application. The following figure shows the Inline Code node.



Example

The developer writing the application to handle calls for the ABC Automotive Inquiry Line wants to include comments in the final source code to be installed on the target voice response system. Since the Code Generation tool does not automatically include comment blocks, the developer uses the Inline Code node to insert comments at the desired places in the application.

The same developer also uses the Inline Code node to call another application and return to the current application when the other is done executing. This is executed much like a subroutine.

Behavior

- This node takes TAS code as you have written it and inserts it into the generated application code.
- Since the TAS 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 lets you paste from the clipboard into the Inline Code Editor.
Note: Code is inserted as is into your application with no syntax checks performed. Any errors are not detected until you attempt to install the application on the target system.

Attributes

Use the Node Inspector to assign and edit the following node attribute.

Attribute	Description
Text	The field displays the first part of any code you have written. To write the TAS code, click the  button. IVR Designer displays the Inline Code window. Enter your TAS code in the Text of Code field. When you are finished, click OK.

Load Variables node

The Load Variables node loads variable data passed to it from another application into the current application. This node is typically used in conjunction with the Save and Exec node and is usually the first node in the current application. The following figure shows the Load Variables node.



For practice using the Load Variables node, see [Passing data between TAS applications](#) on page 480.

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 (Save and Exec node) and assigns the data to variables defined for the current application.
- The Load Variables node can receive 2 Kbytes of variable data as part of the node's operation.

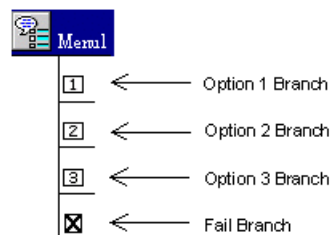
Attributes

Group: Variables

Attribute	Description
Elements	Click the  button to add variables to load 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 load. All variables defined for the current application are displayed on this drop-down menu.
Trace Variables	When set to True , this attribute causes the application to trace the values of all variables defined here on the target voice response system. This lets you use the trace command for these variable values on the target system. If the Trace Variables attribute is set to True , a one-second pause is inserted in the code before the trace function is executed on the target system. This is useful when you need to flush the buffer before executing the trace command.

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.



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 to callers in TAS applications](#) on page 359.

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 Prompts tab below).
- The Initial Prompt presents a predetermined menu of choices to the caller.
- This 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 with enough digits, the Too Few Digits Prompt is played. If the Replay Initial Prompt field is set to True, the Initial Prompt also plays after the Too Few Digits Prompt.
- If the caller enters an invalid response and the Tries Available field value has been met, the Fail branch is executed.
- 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.
- If a system failure occurs, for example if a recognition resource cannot be allocated, the Fail branch is executed immediately. A system failure could occur before the caller has an opportunity to respond to the initial prompt. A system failure can be distinguished from caller response errors by checking whether the Tries Available field value has been met. If the Tries Available field value has been met, the failure is due to caller response errors. If the Tries Available field value has not been met, the failure is due to a system failure.

Inserting branches

To insert a new branch on a Menu node:

1. Right-click the branch 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 use to label the branch.
- To define the branch as a timeout branch, click the  icon.
- To define the branch using a FlexWord grammar, click the  icon. Clicking this icon opens a Grammar Word List window. This window displays the words defined for the FlexWord grammar specified in the Input ASR Mode attribute for this node.
- To rename the branch to use pattern matching, click the  icon. See [Pattern matching](#) on page 568 for information about pattern matching.

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 FlexWord grammar, click the  icon. Clicking this icon opens a Grammar Word List window. This window displays the words defined for the FlexWord 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.

Related topics:

[Menu node attributes](#) on page 584

[Pattern matching](#) on page 586

Menu node attributes

Use the Node Inspector to assign and edit the Menu node attributes.

Prompts tab

Attribute	Description
Initial Prompt	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.

Group: Error Prompts

Attribute	Description
Bad Input Prompt	From the drop-down menu, select the prompt to play when the caller enters invalid input, such as an invalid account number.
Timeout Prompt	From the drop-down menu, select the prompt to play when the caller waits too long to respond to the Initial Prompt.
Too Few Digits Prompt	From the drop-down menu, select the prompt to play when the caller does not enter enough digits.

Response tab

Group: Input Parameters

Attribute	Description
Allow Interrupt	To allow the caller to interrupt while prompts are playing, select True from the drop-down menu. To play the prompts without interruption, select False from the drop-down menu.  Note: You can use this option only if the Input ASR Mode option is set to Touchtones or a speech recognition grammar.
Discard Previous Touchtone	To ignore any touchtones the caller may have entered before the Initial Prompt is played, select True from the drop-down menu. To process any touchtones the caller may have entered before the Initial Prompt is played, select False from the drop-down menu. This lets you to take advantage of the Type Ahead feature, which works only with touchtones.

Group: Input DPR Parameters

Attribute	Description
Input DPR Mode	From the drop-down menu, select the DPR grammar you want to use to collect input from the caller.

Attribute	Description
	 Note: DPR grammars are not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Group: Input ASR Parameters

Attribute	Description
Input Mode	From the drop-down menu, select whether you want to use Touchtones (default) or FlexWord speech recognition as input from the caller.
Mode Chosen	From the drop-down menu, select the variable that holds the value indicating how the caller responded to the Initial Prompt.  Note: If FlexWord speech recognition speech recognition is chosen here, and the caller enters a response using Touchtones, the value returned to the variable is a negative number. Otherwise, the value returned is a positive number.
Reserve ASR/DPR Resource Before Playing Prompt	(Not available if FlexWord speech recognition is selected as the Input Mode). To make sure that the ASR or DPR resource is available before obtaining spoken or dial pulse caller input, select True.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.
Free Reserved ASR/DPR Resource After Caller Response	(Not available if Touchtone is selected as the Input Mode) To free the ASR or DPR resource as soon as spoken or dial pulse input is retrieved from the caller, select True.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Group: Timeouts

Attribute	Description
First Digit Timeout	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.

Attribute	Description
Inter-Digit Timeout	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.

Group: Tries

Attribute	Description
Tries Available	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].
Tries Taken	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].
Replay Initial Prompt	If you want the Initial Prompt to replay each time the caller enters an invalid response (after any error messages when appropriate), select True .

Group: Caller's Response

Attribute	Description
Save Response In	From the drop-down menu, select the variable in which to save the caller's unformatted response. This must be a character variable.

Group: Data Collection

Attribute	Description
Track Caller's Response	(Not available if tracking for all Menu nodes is turned on in the Events String Editor .) To track the caller's response for collection in Call Data Handling reports, select True from the drop-down menu.

Related topics:

[DPR Barge-in](#) on page 99

Pattern matching

Special Characters for Pattern Matching

Use the special characters when you want to distinguish between correct and incorrect caller input (in addition to between valid and invalid input). It also allows you to specify an action to take for each case in the list.

Creating Custom Cases

You can specify up to 12 custom cases of correct input. The system compares the input to each case and performs the action specified for the first true case, similar to an "If- Then- Else" statement. For example, your prompt could be:

"For your checking account balance, say or enter 1.

For your savings account balance, say or enter 2.

To speak with a customer service representative, say or enter 0."

For this example, list "1," "2," and "0" as three custom cases. For the digits 1 and 2, branch off to the appropriate nodes to provide the information requested. For the digit 0, specify a telephone number for an agent.

In many cases, you can describe a correct input but you cannot list all the correct values. For example, valid customer account numbers could be eight digits with the first digit always 4 and the last digit always 9, or a stock number might be six to nine digits in length and always begin with 76.

You can use the pattern symbols and special codes available in IVR Designer to describe a correct input case without specifying a value. Because the cases are compared in the order listed, include the most specific pattern first and more general patterns later.

Pattern Symbols

Character	Function
0, 1, 2,...9	These are the standard touchtone digits.
*, #	These are special touchtone keys, which are frequently used as touchtone control characters (terminator, erase, repeat, or cancel codes).

Custom Special Codes

Character	Function
A, B, C, D	These are extended digits. The touchtone standard includes a 16- digit keypad with a fourth column.
n	This represents any one of the 10 standard digits.
s	This represents either one of the two special digits
e	This represents any one of the four extended digits.
t	This represents any 1 of the 16 touchtone digits.
r	This is the repeat character. This special pattern symbol means "match one or

	more repetitions of the previous pattern symbol."
q	This is the quit character. This special pattern symbol means "stop comparing and accept any input string that has matched up to this point."

 Note:

If you use one of these characters or any other touchtones besides * and # as special codes, they will only have their special function and will not appear in the input. Therefore, checklist cases that include them will not be matched.

The literal characters (0 through 9, *, #, A through D) each matches one instance. The group characters (n, s, e, t) each matches one instance from its group.

The above example of an eight-digit account number beginning with 4 and ending with 9 matches the case "4nnnnnn9."

The two special pattern symbols "r" and "q" are useful when the input you want to match has a varying length (within a specified range). Use the "r" symbol when the input has a varying length portion that includes some repeated digits. Therefore, the symbol that precedes "r" in the pattern must be present one or more times. For example, the "5r2" would match any of the following inputs: "52," "555552," "552," and so on.

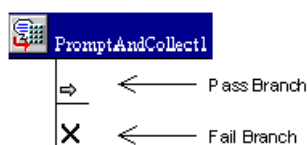
Use the "q" symbol when the first few digits of the input must follow a specified pattern, but the remainder of the input is more variable. For example, "76q" means that if the first part of the input matches the pattern, the remainder of the input can be ignored and the pattern is matched.

The "q" must be the last symbol in a pattern description. IVR Designer will ignore any characters after the "q" in a pattern.

You may use the special codes and pattern symbols together. For example, the "nr" would match either of the following inputs if # is a control character: "1#," "123#". However, the input "12* 45#" would be invalid for "nr" because the asterisk is not a standard digit. For "tr# 1q", the following inputs are valid: "124**# 1", "9# 1", "### 1". However, the input "123" or "# 1" is not valid. The first invalid case "123" does not provide the pound sign (#) or termination digit of 1 and the second invalid case "# 1" does not provide a touchtone digit as indicated by "t".

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 illustration shows the Prompt and Collect node.



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 TAS applications](#) on page 368.

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:

"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: 3159 8433 Is this correct? If so, say 'Yes' at the tone."

When the caller responds "Yes," the application validates the number against a database table field (Order Numbers) and moves on to the next prompt.

Behavior

- This node plays the Initial Prompt as specified in the Node Inspector (see Prompts tab below).
- The Initial Prompt requests caller input.
- This node directs the call based upon the caller's response.
- 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 with enough digits, the Too Few Digits Prompt is played. If the Replay Initial Prompt field is set to True, the Initial Prompt also plays after the Too Few Digits Prompt.
- If the caller enters an invalid response and the Tries Available field value has been met, the Fail branch is executed.
- 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.
- The caller may be asked to confirm the response entered.
- The caller's input can be validated against a variable, set, or table value.
- If a system failure occurs, for example if a recognition resource cannot be allocated, the Fail branch is executed immediately. A system failure could occur before the caller has an opportunity to respond to the initial prompt. A system failure can be distinguished from caller response errors by checking whether the Tries Available field value has been met. If the Tries Available field value has been met, the failure is due to

caller response errors. If the Tries Available field value has not been met, the failure is due to a system failure.

Inserting branches

When you insert a Prompt and Collect node, IVR Designer creates two default branches which cannot be renamed. You can, however, insert additional branches, which can be renamed later.

To insert an additional branch in a Prompt and Collect node:

1. Right-click where you want the new branch.
2. Select Insert Branch from the menu.
IVR Designer displays the Touchtones dialog box.
3. Do one of the following:
 - Enter the digit (or digits) you want to use to label the branch.
 - To define the branch as a timeout branch, click the  icon.
 - To define the branch using a FlexWord grammar, click the  icon. Clicking this icon opens a Grammar Word List window. This window displays the list words defined for the FlexWord grammar specified in the Input ASR Mode attribute for this node.
 - To define the branch to use pattern matching, click the  icon. See [Pattern matching](#) on page 568 for more information.
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 use to label the branch.
 - To rename the branch as a timeout branch, click the  icon.
 - To rename the branch using a list item defined for a FlexWord grammar, click the  icon. Clicking this icon opens a Grammar Word List window. This window displays the words defined for the FlexWord grammar specified in the Input ASR Mode attribute for this node.
 - To rename the branch to use pattern matching, click the  icon. See [Pattern matching](#) on page 568 for information about pattern matching.
4. Click OK.

Related topics:

[Prompt and Collect node attributes](#) on page 591

[Pattern matching](#) on page 596

[Tone masking](#) on page 598

Prompt and Collect node attributes

Use the Node Inspector to assign and edit the Prompt and Collect node attributes.

Attribute	Description
Initial Prompt	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.
Tone Masking	When set to True , this attribute causes the IR application to play intermediate tones while the caller responds to the Initial Prompt.

Error Prompts

Attribute	Description
Bad Input Prompt	From the drop-down menu, select the prompt to play when the caller enters invalid input, such as an invalid account number.
Timeout Prompt	From the drop-down menu, select the prompt to play when the caller waits too long to respond to the Initial Prompt.
Too Few Digits Prompt	From the drop-down menu, select the prompt to play when the caller does not enter enough digits.
Confirm Input Prompt	From the drop-down menu, select the prompt to play after the caller enters a response. This prompt repeats what the caller entered and asks the caller to confirm whether the entry is correct.

Input Parameters

Attribute	Description
Allow Interrupt	To allow the caller to interrupt while prompts are playing, select True from the drop-down menu. To play the prompts without interruption, select False from the drop-down menu.  Note: You can use this option only if the Input ASR Mode option is set to Touchtones or a speech recognition grammar.
Discard Previous Touchtone	To ignore any touchtones the caller may have entered before the Initial Prompt is played, select True from the drop-down menu. To process any touchtones the caller may have entered before the Initial Prompt is played, select False from the drop-down menu.

Attribute	Description
	This lets you to take advantage of the Type Ahead feature, which works only with touchtones.
Maximum Length	Enter the maximum number of digits to accept from the caller. This may be a variable from the drop-down menu.
Minimum Length	Enter the minimum number of digits to accept from the caller. This may be a variable from the drop-down menu.
Terminate Response With	Enter the characters to be entered by the caller to indicate the end of a response. This may be a variable from the drop-down menu.

Timeouts

Attribute	Description
First Digit Timeout	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.  Note: The First Digit Timeout attribute does not apply to applications that use Proxy Speech Recognition. For applications that use Proxy Speech Recognition, the timeout value must be set using the RequestPROXY external function, or on the speech recognition server. Refer to your speech recognition product documentation for information about setting a recognition timeout value.
Inter-Digit Timeout	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.

Tries

Attribute	Description
Tries Available	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 3.
Tries Taken	From the drop-down menu, select a variable to store the number of tries it took a caller to enter a correct response.
Replay Initial Prompt	If you want the Initial Prompt to replay each time the caller enters an invalid response (after any error messages when appropriate), select True .

Caller's Response

Attribute	Description
Save Response In	From the drop-down menu, select the variable in which to save the caller's unformatted response.

Attribute	Description
Save Formatted Response In	From the drop-down menu, select the variable in which to save the caller's formatted response.
Use Input to Select Active Record	When set to True , this causes the selected table record's values to be stored in the variables specified in the previous Query Table node.

Input DPR Parameters

Attribute	Description
Input DPR Mode	From the drop-down menu, select the DPR grammar you want to use to collect input from the caller.  Note: DPR grammars are not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Input ASR Parameters

Attribute	Description
Input Mode	From the drop-down menu, select whether you want to use Touchtones (default) or FlexWord speech recognition as input from the caller.
Mode Chosen	From the drop-down menu, select the variable that holds the value indicating how the caller responded to the Initial Prompt.  Note: If FlexWord speech recognition speech recognition is chosen here, and the caller enters a response using Touchtones, the value returned to the variable is a negative number. Otherwise, the value returned is a positive number.
Reserve ASR/ DPR Resource Before Playing Prompt	To make sure that the ASR or DPR resource is available before obtaining spoken or dial pulse caller input, select True.  Note: To reserve ASR or DPR resources, you must specify a node for the Fail branch. If you do not, you can set this value to True, but no code is generated to allocate ASR/DPR resources during the code generation process.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Attribute	Description
Free Reserved ASR/DPR Resource After Caller Response	To free the ASR or DPR resource as soon as spoken or dial pulse input is retrieved from the caller, select True.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Data Collection

Attribute	Description
Track Caller's Response	(Not available if tracking for all Menu nodes is turned on in the Events String Editor.) To track the caller's response for collection in Call Data Handling reports, select True from the drop-down menu.

Confirmation DPR Parameters

Attribute	Description
Confirmation Input DPR Mode	From the drop-down menu, select the dial pulse grammar you want to use as confirmation input from the caller.  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.

Confirmation ASR Parameters

Attribute	Description
Yes Digit	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].
Confirmation ASR Input Mode	From the drop-down menu, select whether you want to use Touchtones or FlexWord speech recognition as confirmation input from the caller.
Confirmation Mode Chosen	From the drop-down menu, select the variable that holds the value indicating how the caller responded to the Confirmation Prompt.
Confirmation Reserve ASR/DPR Resource Before Playing Prompt	To make sure that the ASR/DPR resource is available before obtaining spoken or dial pulse confirmation, select True .

Attribute	Description
	 Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.
Confirmation Free Reserved ASR/ DPR Resource After Caller Response	To free the ASR or DPR resource as soon as spoken or dial pulse confirmation is retrieved from the caller, select True .  Note: DPR is not currently supported on Avaya IR Release 1.0 through Release 1.2. DPR is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.
Validate Response Against	From the drop-down menu, select the entity to be used for validating the caller's input. There are four options: • Variable • Set • Table • (none)

Variable Validation

The attributes in this group are available only if **Variable** is selected in the **Validate Response Against** field.

Attribute	Description
Relation	Enter or from the drop-down menu select the relation symbol: • = • <> • < • <= • > • =>
Variable	Enter or from the drop-down menu select the variable containing the value to compare with the data collected from the caller.

Set Validation

The attributes in this group are available only if **Set** is selected in the **Validate Response Against** field.

Attribute	Description
Name of the Set	Enter the name of the set of values used to validate the caller's response. You must then enter a set of valid values, such as 1-999 or 1,2,7,0.

Table Validation

The attributes in this group are available only if **Table** is selected in the **Validate Response Against** field.

Attribute	Description
Table	Enter or from the drop-down menu select the table that contains the field value to be used to validate the caller's response.
Field	Enter or from the drop-down menu select the field within the selected table that contains the value to be used to validate the caller's response.

Related topics:

[DPR Barge-in](#) on page 99

Pattern matching

Special Characters for Pattern Matching

Use the special characters when you want to distinguish between correct and incorrect caller input (in addition to between valid and invalid input). It also allows you to specify an action to take for each case in the list.

Creating Custom Cases

You can specify up to 12 custom cases of correct input. The system compares the input to each case and performs the action specified for the first true case, similar to an "If- Then- Else" statement. For example, your prompt could be:

"For your checking account balance, say or enter 1.

For your savings account balance, say or enter 2.

To speak with a customer service representative, say or enter 0."

For this example, list "1," "2," and "0" as three custom cases. For the digits 1 and 2, branch off to the appropriate nodes to provide the information requested. For the digit 0, specify a telephone number for an agent.

In many cases, you can describe a correct input but you cannot list all the correct values. For example, valid customer account numbers could be eight digits with the first digit always 4 and the last digit always 9, or a stock number might be six to nine digits in length and always begin with 76.

You can use the pattern symbols and special codes available in IVR Designer to describe a correct input case without specifying a value. Because the cases are compared in the order listed, include the most specific pattern first and more general patterns later.

Pattern Symbols

Character	Function
0, 1, 2,...9	These are the standard touchtone digits.
*, #	These are special touchtone keys, which are frequently used as touchtone control characters (terminator, erase, repeat, or cancel codes).

Custom Special Codes

Character	Function
A, B, C, D	These are extended digits. The touchtone standard includes a 16- digit keypad with a fourth column.
n	This represents any one of the 10 standard digits.
s	This represents either one of the two special digits
e	This represents any one of the four extended digits.
t	This represents any 1 of the 16 touchtone digits.
r	This is the repeat character. This special pattern symbol means "match one or more repetitions of the previous pattern symbol."
q	This is the quit character. This special pattern symbol means "stop comparing and accept any input string that has matched up to this point."

Note:

If you use one of these characters or any other touchtones besides * and # as special codes, they will only have their special function and will not appear in the input. Therefore, checklist cases that include them will not be matched.

The literal characters (0 through 9, *, #, A through D) each matches one instance. The group characters (n, s, e, t) each matches one instance from its group.

The above example of an eight-digit account number beginning with 4 and ending with 9 matches the case "4nnnnnn9."

The two special pattern symbols "r" and "q" are useful when the input you want to match has a varying length (within a specified range). Use the "r" symbol when the input has a varying length portion that includes some repeated digits. Therefore, the symbol that precedes "r" in

the pattern must be present one or more times. For example, the "5r2" would match any of the following inputs: "52," "555552," "552," and so on.

Use the "q" symbol when the first few digits of the input must follow a specified pattern, but the remainder of the input is more variable. For example, "76q" means that if the first part of the input matches the pattern, the remainder of the input can be ignored and the pattern is matched.

The "q" must be the last symbol in a pattern description. IVR Designer will ignore any characters after the "q" in a pattern.

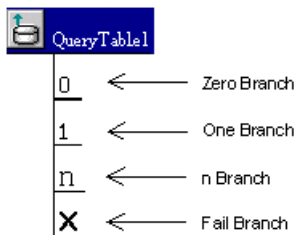
You may use the special codes and pattern symbols together. For example, the "nr" would match either of the following inputs if # is a control character: "1#," "123#". However, the input "12* 45#" would be invalid for "nr" because the asterisk is not a standard digit. For "tr# 1q", the following inputs are valid: "124**# 1", "9# 1", "### 1". However, the input "123" or "# 1" is not valid. The first invalid case "123" does not provide the pound sign (#) or termination digit of 1 and the second invalid case "# 1" does not provide a touchtone digit as indicated by "t".

Tone masking

To secure sensitive details you enter using DTMF inputs, such as PIN codes, credit card numbers, and bank account numbers, tone masking is used. This feature can be enabled or disabled from the Prompt and Collect node of IVR Designer. When you enable tone masking, and you enter a DTMF input, random digits are played to mask your input. Tone masking is available only for TAS applications.

Query Table node

The Query Table node retrieves records from a database table based upon the specified selection criteria. The following figure shows the Query Table node.



For more information about using the Query Table node, see [Locating information in tables](#) on page 375.

Example

A customer calls the ABC Automotive Inquiry Line about the status of an order. The order number is obtained from the customer and is used as the selection criteria to retrieve the order status from a local database table.

Behavior

- Records are selected from the specified database table based upon the defined selection criteria.
- When selection criteria are not specified, all the records in the database table are retrieved.
- The field values from the first record found that matches the selection criteria is placed in the Assignments from records variable (see Attributes).
- If no records matching the selection criteria are found, the Zero (0) branch is taken.
- If only one record matching the selection criteria is found, the One (1) branch is taken.
- If more than one record matching the selection criteria are found, the n branch is taken.
- If the specified database table could not be accessed for some reason, the Fail branch is taken.
- If multiple records matching the selection criteria are found, the values from the first record found are returned. The remaining records can be accessed by using the NextRecord or PrevRecord function of the Set and Test node, or by using the Automenu node to speak all the selected records back to the caller.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Table to query	Enter or from the drop-down menu select the table from which the records are to be selected.
Selection criteria	Displays the fields in the database table and the variables or literal values to which the fields are to be compared. When the  button is clicked, IVR Designer displays the Query Table Editor window. Use this editor to assign data from record fields to values or variables.
Assignments from records	Displays the fields in the database table and the variables that will be assigned the values found in each field. When the  button is clicked, IVR Designer displays the Query Table Editor window.

	Use this editor to assign data from record fields to values or variables.

Related topics:

[Query Table Editor](#) on page 600

Query Table Editor

The following table describes the columns in the Query Table Editor.

Column	Description
Field Name	Displays the names of all defined fields for the selected table.
Relation	Selects and displays the relation operator for the selected variable field. Possible value: =
Variable/Literal	Lets you select from a drop-down menu the variable, the value of which to compare with the database field, to set as the selection criteria.  Note: The variable type for this variable or literal value must match the type for the Assign value to field.
Assign value to	Lets you select from a drop-down menu the variable to assign the value from the first record that matches the selection criteria. For additional records, you can use the NextRecord and PrevRecord functions in the Set and Test node.  Note: The variable type for this variable (or literal value) must match the type for the Variable/Literal field.

Return node

The Return node stops the execution of the call flow that contains it. The following figure shows the Return node.



Behavior

- In the [Main] call flow, the Return node stops the execution of the application or returns control to the calling application.
- In any other call flow, the Return node results in the call flow returning and taking the branch with the same name as the Return node.
- In the [Main] call flow, the Return node terminates both the call (disconnects) and the application script.

Attributes

There are no user-configurable attributes for the Return node.

Save and Exec node

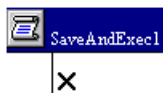
The Save and Exec node calls another application on the target system. The called application must be compiled and installed on the same target system on which the IVR Designer application that references the called application is installed.

By setting the attribute 'return control' in the Node Inspector to true, control will be returned to the calling application. If 'return control' is set to false, the pass branch will not appear. The Save and Exec node is typically used in conjunction with the Load Variables node.

Note:

The Save and Exec node is similar but not identical to the Execute and Subprogram external functions. The differences lie in the way the two pass arguments and variables to the new application. When calling a Script Builder application, you must use the Execute or Subprogram external function because Script Builder does not have Load Variables node functionality or any equivalent action. For IVR Designer applications, you can use either the Save and Exec node, the Execute or Subprogram external functions.

The following figure shows the Save and Exec node.



For practice using the Save and Exec node, see [Passing data between TAS applications](#) on page 480.

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 Save and Exec node to pass control from one part to the other.

Behavior

- This node calls another application (script) and executes it, passing control to the new application.
- Control can return to the current application if 'return control' is set to true.
- This node can pass 2 Kbytes of variable data as part of the node's operation.
- If for some reason the target system cannot execute the new application – if, for example, the application does not exist on the target system – the Fail branch is taken.

Attribute	Description
Script Name	Type a literal value, type the name of a variable containing the name of the application, or from the drop-down menu select the variable containing the name of the application or script to execute.
Return Control	When set to True , this attribute causes the called application to return control to the calling application after the called application is executed.
Put Return Code In	From the drop-down menu, select the variable in which to store the return code from the called application. The Put Return Code In attribute is available only if the Return Control attribute is set to True .
Create Call Record	When set to True , this attribute causes a call data record to be created.
Return Value	From the drop-down menu, select the variable in which to store the value returned from the called application. The Return Value attribute is available only if the Return Control attribute is set to True .

Group: Variables

Attribute	Description
Elements	Click the  button to add variables to pass to the other application, and then select the variable you want from the drop-down menu. The # elements field displays how many variables have been selected. From the corresponding drop-down menu, select the variable you want to pass to the other application. All variables defined for the

Attribute	Description
	current application are displayed on this drop-down menu. The Name column field displays the number assigned to each variable selected.
Trace Variables	When set to True , this attribute causes the application to trace the variables that are passed to the called application. You can use the trace command on the target system to display the values that are passed to the called application. If the Trace variable is set to True , a one-second pause is inserted in the code before the trace function is executed on the target system. This is useful when you need to flush the buffer before executing the trace command.

Set and Test node

The Set and Test node lets you manipulate and evaluate a wide range of variable expressions and branch conditions. This node has two basic uses: It lets you assign values to variables, and to compare a variable to another variable or value.

The kinds of variable assignments you can perform include the following:

- Arithmetic
- Assign
- Concat
- ItemCount
- ItemOf
- Length
- NextRecord
- Parse
- PrevRecord
- Substring



Note:

Before you can set test conditions for an expression, you must first add a branch to the node.

The following figure shows the Set and Test node.



For practice using the Set and Test node, see [Using calculations in TAS applications](#) on page 379 and [Handling strings in TAS applications](#) on page 436.

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.

Inserting and renaming branches

When you select Insert Branch or Rename Branch from the right-click menu on a Set and Test node, IVR Designer displays the Set Test Branch dialog box.

Type the name you want for the branch, and click OK.

Attributes

You can insert additional branches for a Set and Test node. Use the Node Inspector to assign and edit the following node attributes.

Tab	Attribute	Description
Expressions	Expressions	To add expressions, click the plus icon. To set an expression's attributes, select an element, and then click the  button. The Set Test Expression Editor displays.
Branch Conditions (Optional)	Conditions	To add a Branch Conditions statement, click the plus icon. Note: 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 Set Test Branch Expression Editor displays.

Related topics:

[Set Test Expression Editor](#) on page 605

[Set Test Branch Condition Editor](#) on page 611

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.

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.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.

Operand 1	Type a literal value or a variable name, or from the drop-down menu select the variable containing the value for the first operand.
Operator	From the drop-down menu, select the operator to use. Options include: +, -, *, /, and mod.
Operand 2	Type the literal value, or from the drop-down menu select the variable containing the value for the second operand.

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.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.
Source Value	Type the literal value, or from the drop-down menu select the variable containing the value, to be assigned to the variable.

Concat expression

The Concat expression concatenates two character strings into a single string. The following table describes the fields for the Concat expression type.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.
Source String 1	Type the character string, or from the drop-down menu select the variable containing the string to be assigned as the first string.
Source String 2	Type the character string, or from the drop-down menu select the variable containing the string to be assigned as the second string.
Maximum Length	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.

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.

The following table describes the fields for the ItemCount expression type.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.
Source List	Type or from the drop-down menu select the variable containing a list of items to be counted.

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.

The following table describes the fields for the ItemOf expression type.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.  Note: If the item number is out of range or otherwise invalid, or if the data source does not exist, a null string is returned.
Source List	Type or from the drop-down menu select the variable containing a list of items.
Item Number	Type a literal value or from the drop-down menu select the variable

	containing the item number of the item you want.

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.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.  Note: The variable must be a number variable.
Source String	Type the character string, or from the drop-down menu select the variable containing the string whose characters are to be counted.

NextRecord expression

The NextRecord expression returns the value of the index for the next record in a previously selected list of records and updates the variables populated by the most recent database query.

 Note:

The return value for this field is incremented before the record is retrieved and represents the number of the current record. In other words, the NextRecord number is actually the current record number after the node is executed.

The following table describes the fields for the NextRecord expression type.

Field	Description
Next Record Number	Type or from the drop-down menu select the name of the variable to assign the next record number to. If there is no next record, a value of zero is returned.

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 following table describes the fields for the Parse expression type.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.
Source String	Type the character string, or from the drop-down menu select the variable containing the string to be parsed.
Parsing Separator	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.

PrevRecord expression

The PrevRecord expression returns the value of the index for the previous record in a previously selected list of records and updates the variables populated by the most recent database query.

**Note:**

The return value for this field is decremented before the record is retrieved and represents the number of the current record. In other words, the PrevRecord number is actually the current record number after the node is executed.

The following table describes the fields for the PrevRecord expression type.

Field	Description
Previous Record Number	Type or from the drop-down menu select the name of the variable to assign the previous record number to. If there is no previous record, a value of zero is returned.

Substring expression

The Substring expression returns a selected portion of a character string. The following table describes the fields for the Substring expression type.

Field	Description
Destination Variable	Type or from the drop-down menu select the name of the variable to which you want to assign the resulting value.
Source Substring	Type the substring, or from the drop-down menu select the variable containing the substring to be returned.
Starting at Position	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. If the number specified here is a negative number, the resulting substring is defined by counting backwards from the end of the source string.
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.

Field	Description
Go To Branch	Type or from the drop-down menu select the name of the branch for which you want to set conditions.
Variable	Type a variable name or from the drop-down menu select the variable you want to use as the condition test variable.
Relation	From the drop-down menu select the relation operator to use in testing the variable. Options include: =, <>, <, <=, >, and >=.
Value	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.
Plus icon	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.
Minus icon	Collapses the Set Test Branch Condition Editor after it has been expanded.
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).

TAS2VXI node

IVR Designer supports TAS applications to call VoiceXML applications. The TAS2VXI node allows TAS application to instruct Avaya Voice Browser (AVB) to execute VoiceXML scripts.

The following figure shows the TAS2VXI node.



Note:

If **Return Control** is set to **False**, the pass branch does not appear.

Following are the processes that take place when the TAS2VXI node is used:

- The TAS2VXI node calls VXI application (AVAYAVXI0-5 scripts) and executes it, passing control to the new application.
- Control can return to the current application, if **Return Control** is set to **True**.
- The fail branch is taken, if the target system cannot execute the new application.
- On failure the control returns to the fail branch node set by the user.
- The total data sent from a TAS to VXI application must not exceed 512 bytes.

Related topics:

[TAS2VXI node attributes](#) on page 612

TAS2VXI node attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
URL	Specifies the path pointing to the VXML application played by the AVAYAVXI0-5 script. You can define a URL that enables you to access the called VXML application in one of the following ways: • Enter the URL or the complete path. It is recommended that you define the URL using the <code>http://</code> or <code>file://</code> protocol identifiers. • Select the required character variable that contains the URL from the drop-down list.
Return Control	When set to True , this attribute causes the called application to return control to the calling application after the called application is executed.

Attribute	Description
Put Return Code In	From the drop-down menu, select the variable in which the return code from the called application is to be stored. The Put Return Code In attribute is available only if the Return Control attribute is set to True .
Create Call Record	When set to True , this attribute causes a call data record to be created.
Input Variables	From the drop-down menu, select a variable containing the name of the parameter to be passed. This field contains the following dependencies: • You must click the plus symbol in the (<number> elements) box to add a new input variable. • You can add up to 10 input variables. • The number of input variables must be equal to the number of input value variables. • IVR Designer relates the value of an input variable to the value of an input value variable. For example, when you define <i>[NameVar1]</i> as <i>number</i> and <i>[ValueVar1]</i> as 1, IVR Designer relates <i>[NameVar1]=number</i> to <i>[ValueVar1]=1</i>.
Input Value Variables	From the drop-down menu, select a variable containing the value of the parameter to be passed. This field contains the following dependencies: • You must click the plus symbol in the (<number> elements) box to add a new input value variable. • You can add up to 10 input value variables. • The number of input value variables must be equal to the number of input variables. • IVR Designer relates the value of an input value variable to the value of an input variable. For example, when you define <i>[ValueVar1]</i> as 1 and <i>[NameVar1]</i> as <i>number</i>, IVR Designer relates <i>[ValueVar1]=1</i> to <i>[NameVar1]=number</i>.
Output Variables	From the drop-down menu, select a variable that stores the value returned from the called VXML application. This field contains the following dependencies: • You must click the plus symbol in the (<number> elements) box to add a new output variable. • You can add up to 10 output variables. • Active only if the Return Control field is set to True.
Trace Variables	The application traces the Put Return Code In variable when this is set to True . The default is False . The trace values can be displayed by using the trace command on the target system. If the Trace

Attribute	Description
	Variables attribute is set to True , a one-second pause is inserted in the code before the trace function is executed on the target system. This is useful when you need to flush the buffer before executing the trace command.

**Note:**

Total data length including the **URL** , **Input variables**, and **Input value** variables passed from TAS to VXI must not exceed 512 bytes.

Time Branch node

The Time Branch node is used to direct a call based on time of day, day of the week, or specified date. The following figure shows the Time Branch node.



For more information about and practice using the Time Branch node, see [Branching based on time, day, or date in TAS applications](#) on page 348.

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 a branch or branches that have the time conditions you want to set.
- 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

Use the Node Inspector to assign and edit the following node attribute.

Attribute	Description
Trace Variables	Embeds a command telling the target system to use the trace function on this node when running the application. This is useful primarily as a debugging tool after installing your application on the target system.

Trace Variables node

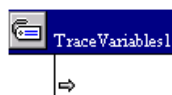
The Trace Variables node lets you print out the values of the selected variables on the target voice response system. This is useful primarily as a debugging tool after installing your application on the target voice response system.



Note:

For the Trace Variables node to be useful, you have to run the trace command on the target system.

The following figure shows the Trace Variables node.



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 Trace Variables node in the call flow both before and after the variable is supposed to be updated. The output from the

trace command on the CONVERSANT or Avaya IR system displays the value of the variable both before and after it is updated.

Behavior

- The selected variable is tagged to be traced when installed on the target voice response system.
- When the trace function is invoked on the CONVERSANT or Avaya IR system, the value for the selected variable is printed to the output.

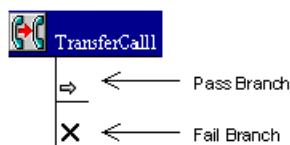
Attributes

Use the Node Inspector to assign and edit the following node attributes.

Group	Attribute	Description
	Sleep Before Trace	When set to True, this attribute inserts a one-second pause in the code before the trace function is executed on the target system. This can be useful when you need to flush the buffer before executing the trace command.
Variables	Elements	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.

Transfer Call node

The Transfer Call node is used to transfer or bridge a call to another number. The following figure shows the Transfer Call node.



For more information about using the Transfer Call node, see [Using intelligent transfer](#) on page 383.

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 this node is executed, the call is transferred or bridged to the specified telephone number.
- If the transfer or bridge is completed successfully, the Pass branch is taken.
- If the transfer or bridge is not completed within the time specified, the Fail branch is taken.
- Depending on the Transfer Type you specify, 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 Action and Transfer Type you specify (see Action and Transfer Type below).

Transfer Options tab

Group: Transfer Options

Attribute	Description
Phone Number	Enter the telephone number or from the drop-down menu select the variable containing the telephone number to transfer the caller to.  Note: Use alphanumeric characters to enter the telephone number.
Action	For Avaya Interactive Response Release 1.2.1 or later, select one of the following actions: • Transfer (default) • Bridge The transfer action transfers the call according to the Transfer Type option that is selected. The bridge action places an outbound call to a third party and maintains the connection while the caller interacts with the third party.

Attribute	Description
	When the third party hangs up, the application continues with the next node. For CONVERSANT and Avaya Interactive Response Release 1.2 or earlier, use the default value for this attribute.
Transfer Type	From the drop-down menu select the type of transfer to make. There are three options: • Blind (default): The application dials the third-party number to start the transfer, and then relinquishes all call-handling responsibilities. In other words, in a blind transfer, the transfer call is placed, and the call is released. The caller is then left to deal with a busy signal or a no-answer. • Full CCA: (Call Classification Analysis) This is a more advanced form of transfer. Full CCA provides more complete interpretation of network progress tones than does intelligent CCA. It includes detection of busy, answered, ring no-answer, and reorder. • Intelligent (CCA): The application dials the third-party number to start the transfer. It then monitors call progress signals to determine if the line is busy, is ringing, or has been answered. The application continues to monitor the transfer until it is successfully completed. If the call cannot be completed successfully, control of the call returns to the application and the Fail branch of the node is taken.
Transfer Status	From the drop-down menu select the variable in which to store the transfer status result. The default value is [TRANSFER_RESULT].
Equipment Group Number	For Avaya Interactive Response Release 1.2.1 or later, type or select the variable that specifies the Equipment Group Number associated with the incoming call. The variable type must be number. For CONVERSANT and Avaya Interactive Response Release 1.2 or earlier, use the default value for this attribute. The range is 0 through 31. The default value is 0.
Maximum Rings	(Not available for Transfer Type Blind.) Enter a number, or from the drop-down menu select the variable that specifies the maximum number of rings to allow before directing the call to the Fail branch. The range is 0 through 20. The default value is 7.
ANI	Type up to 30 characters or select the variable that specifies the Automatic Number Identification (ANI) for the call, if available. The variable type must be character. For the Transfer Call node, ANI is supported on Avaya Interactive Response Release 1.2.1 or later.
UUI	Type up to 120 characters or select the variable that specifies the User-to-User Information (UUI). The variable type must be character. If you enter text for this attribute, do not use double quotes ("). For the Transfer Call node, UUI is supported on Avaya Interactive Response Release 1.2.1 or later.

Attribute	Description
	 Note: This is applicable only if the protocol is ISDN PRI.

Group: Answer Options

These options are not available for Transfer Type **Blind**.

Attribute	Description
Answer Message	From the drop-down menu select the prompt to play to the third party when the call is answered.  Note: The Answer Message attribute is not currently supported on Avaya IR.
Allow Answer Message Interrupt	When set to True , , this attribute allows the third party to interrupt the Answer Message and connect with the caller more quickly.
Answer Connection	From the drop-down menu select one of the following options, which determines how the call is handled after an answer is detected: • Complete: (Default) Connects the caller to a third party when the third party answers. • Hold: Leaves the caller on hold when the third party answers.  Note: The Hold option is not currently supported on Avaya IR.

Group: Caller Options

Attribute	Description
Message to Caller	From the drop-down menu, select the prompt to play to the caller prior to transferring the call.

Failure Treatment tab**Group:** Transfer Failure Options

These options are not available for Transfer Type **Blind**.

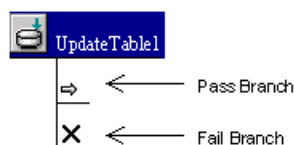
Attribute	Description
Failure Message	From the drop-down menu, select the prompt or message to play if the call cannot be completed successfully.

Attribute	Description
Allow Failure Message Interrupt	When set to True , this attribute allows the caller to interrupt the Failure Message by hanging up or selecting another option.
Busy Connection	From the drop-down menu, select one of the following options, which determines how the call is handled when a busy signal is detected: • Reconnect: (Default) Drops the third party and reconnects the caller to the application so that other action can be taken when the third party is busy. • Complete: Connects the caller to a third party when the third party is busy. • Hold: Leaves the caller on hold when the third party is busy.  Note: The Complete and Hold options are not currently supported on Avaya IR.
No Answer Connection	From the drop-down menu, select one of the following options, which determines how the call is handled when no answer is detected: • Reconnect: (Default) Drops the third party and reconnects the caller to the application so that other action can be taken when the third party does not answer. • Complete: Connects the caller to a third party when the third party does not answer. • Hold: Leaves the caller on hold when the third party does not answer.  Note: The Complete and Hold options are not currently supported on Avaya IR.
Error Connection	From the drop-down menu select one of the following options, which determines how the call is handled when an erroneous connection is detected: • Reconnect: (Default) Drops the third party and reconnects the caller to the application so that other action can be taken when there is an erroneous connection. • Complete: Connects the caller to a third party when there is an erroneous connection. • Hold: Leaves the caller on hold when there is an erroneous connection.

Attribute	Description
	 Note: The Complete and Hold options are not currently supported on Avaya IR.

Update Table node

The Update Table node is used to change data within a specified database table. The following figure shows the Update Table node.



Example

A customer calls ABC Automotive wanting to change the quantity of a particular item ordered earlier. The order information is obtained from the caller and used as selection criteria to locate the order information records in the database. If a match is found, the quantity field can be updated in the database table.

Behavior

- When a record is found that matches the selection criteria, the record is changed in the database table.
- If no selection criteria are specified, all records are selected from the specified database table.
- If Update multiple records is set to True and at least one record matching the selection criteria is found, all records found are updated.
- If Update multiple records is set to False and more than one record matching the selection criteria is found, no records are updated.
- If one or more records in the database table are updated, the Pass branch is taken.
- If no records in the database table are updated, the Fail branch is taken.
- If for some reason the database table cannot be accessed, the Fail branch is taken.

Attributes

Use the Node Inspector to assign and edit the following node attributes.

Attribute	Description
Table to update	Enter or from the drop-down menu select the table in which the records are to be changed.

Selection criteria	Displays the fields in the database table and the variables or literal values to which the fields are to be compared. When the  button is clicked, IVR Designer displays the Update Table Editor window. Use this editor to assign data from record fields to values or variables.
Assignments from records	Displays the fields in the database table and the variables that will be assigned the values found in each field. When the  button is clicked, use this editor to assign data from record fields to values or variables.
Update multiple records	When set to True, this option lets you update more than one record at a time.

Related topics:

[Update Table Editor](#) on page 622

Update Table Editor

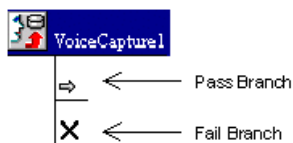
The following table describes the columns in the Update Table Editor.

Column	Description
Field Name	Displays the names of all defined fields for the selected database table.
Relation	Lets you select and displays the relation operator for the selected variable field. Possible values: =
Variable/Literal	From the drop-down menu, select the variable to set as the selection criteria for updating the variable.  Note: The variable type for this variable or literal value must match the type for the Update with value field.
Update with value	From the drop-down menu select the variable containing the data to update the field. For additional records, you can use the NextRecord and PrevRecord functions in the Set and Test node.

	* Note: The variable type for this variable or literal value must match the type for the Variable/Literal field.

Voice Capture node

The Voice Capture node is used primarily to record a caller's spoken response. Use this node to gather information that cannot be entered with touchtones, for example, from callers with rotary telephones. You can use it also to collect alphanumeric input from callers, for example, street addresses or city names. The following figure shows the Voice Capture node.



For practice using the Voice Capture node, see [Capturing voice from callers in TAS applications](#) on page 487.

Example

ABC Automotive wants to gather names and addresses from customers calling the Inquiry Line. Callers are prompted:

"If you would like to be on our mailing list, please say your name, street address, city, state, and zip code.

Please begin recording at the sound of the tone."

When the caller responds, the information is stored in a variable for later transcription to a mailing database.

Behavior

- When a caller speaks a valid response, the Pass branch is taken.
- If a caller does not speak a response and has exceeded the number of tries, the Fail branch is taken.

Related topics:

[Voice Capture node attributes](#) on page 624

Voice Capture node attributes

Use the Node Inspector to assign and edit the Voice Capture attributes.

Prompts tab

Attribute	Description
Initial Prompt	From the drop-down menu, select the prompt you want to play when the Voice Capture node begins to execute, or double-click to create a new prompt.
Confirm Prompt	From the drop-down menu, select the prompt to play after the caller responds. This prompt is used to get feedback as to whether the caller's initial response was correct.
Confirm Error Prompt	(Inactive until a Confirm Prompt has been selected) From the drop-down menu, select the prompt to play when the caller does not enter a valid Yes or No digit.

Recording tab

Group: Recording Options

Attribute	Description
Play Beep Before Recording	When set to True , (default), this option plays a beeping tone at the end of the Initial Prompt before beginning the recording.
Max Length	Enter or from the drop-down menu select the value or variable that contains the value for the maximum length of time (in seconds) to allow for each caller's response. The range is 1 through 999. The default value is 45.
Rate	From the drop-down menu, select the voice-encoding algorithm to use when recording the caller's voice. Options include: • ADPCM16: (Default) Adaptive Differential Pulse Code Modulation at 16 Kbps (not supported on Avaya IR) • ADPCM32: Adaptive Differential Pulse Code Modulation at 32 Kbps (not supported on Avaya IR) • PCM64: Pulse Code Modulation at 64 Kbps in Mu-law encoding format (not supported on Avaya IR) • CELP16: Code Excited Linear Prediction at 16 Kbps (not supported on Avaya IR) • G.711: Pulse Code Modulation at 64 Kbps (not supported on CONVERSANT R9 and earlier releases)
Initial Timeout	Enter or from the drop-down menu select the number or the variable containing the number of seconds to wait for the caller to respond before replaying the Initial Prompt. The default value is 5.

Attribute	Description
Terminate On	From the drop-down menu, select the means by which the end of the recording is detected and the recording is terminated. Options include: • Maximum Duration: Terminates the response only when the value in the Max Length field is reached. • Silence Detection: (Default) Terminates the response when a user-determined amount of silence is detected. The number of seconds of silence is set in the Completion field. • Touchtone Detection: Terminates the response when a touchtone is detected.
Completion	(Used only with the Silence Detection option above) Enter the number or from the drop-down menu select the variable containing the number of seconds of silence to use in detecting the end of the recording. The default value is 5.

Group: Recorded Speech Options

Attribute	Description
Save in Variable	Enter the variable name or from the drop-down menu select the variable to be used to store the caller's response (must be a number variable with a Spoken As setting of NX and a Speak Using setting of recorded in the Variables Manager).
Save On Hangup	When set to True (default), this option saves the caller's response even if the caller hangs up.
Actual Phrase Length	Type or select from the drop-down menu the name of the variable to assign the number representing the length of the recorded phrase in seconds.
Return Field	Type or select from the drop-down menu the name of the variable to assign the return code value from the voice capture operation. The following return codes are available for the Voice Capture node: 1 – Coding completed normally 2 – Touchtone termination 3 – Completion timeout -1 – Insufficient space -2 – Code failure -3 – Initial timeout -4 – Caller hangup

Group: Tries

Attribute	Description
Tried Available	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 3.

Attribute	Description
Tries Taken	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].

Confirmation tab

The options on this tab are only available if a Confirm Prompt is selected on the Prompts tab.

Attribute	Description
Yes Digit	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].
No Digit	Enter the digit, using a variable name or a literal value, to be used to negate a caller's response. The default value is [NoDigit].
Confirm Tries	Enter the number or from the drop-down menu select the variable representing the number of tries the caller has to verify a correct recording. The default value is 2.

Group: Confirmation ASR Parameters

Attribute	Description
Confirmation Input Mode	From the drop-down menu, select whether you want to use Touchtones (default) or FlexWord speech recognition as confirmation input from the caller.
Confirmation Mode Chosen	From the drop-down menu, select the variable that holds the value indicating how the caller responded to the Confirmation Prompt.
Confirmation Reserve ASR/DPR Resource Before Playing Prompt	To make sure that the ASR or DPR resource is available before obtaining spoken or dial pulse confirmation, select True . The default value is False .
Confirmation Free Reserved ASR/DPR Resource After Caller Response	To free the ASR or DPR resource as soon as spoken or dial pulse confirmation is retrieved from the caller, select True . The default value is False .

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. The use of variables in prompts is typically defined using the Prompts tab on the Node Inspector.

The following table describes the system variables included with IVR Designer. For more information about a particular variable, see the topic for that variable.

Variable Name	Description
ANI (Automatic Number Identification) on page 629	Stores the calling party ID number, if available.
Attempts on page 630	Stores the number of times the caller tries to respond to a prompt.
Channel on page 630	Channel number on which the current call is being handled.
CI_MODE on page 631	Stores the SSP card number.
CI_NO_DIGS_GOT on page 632	Is the default field for the number of input digits entered by the caller on the last input attempt for a Prompt and Collect node.
CI_RECOG on page 632	Number, set by the system, representing the recognizer used to identify a caller response.
CollectedDigits on page 633	Value of the digit or digits pressed or spoken by the caller in responding to a prompt.
Date on page 634	Current date in yyyyymmdd format.
Default on page 634	24-character string variable; available for general use by application developers.
DNIS (Dialed Number Identification Service) on page 635	Stores the calling party ID number dialed, if available.
Errno on page 635	Number variable used for return values.
F_HOST_ERROR on page 636	Stores the error code if the Get Screen action did not return a recognized screen.

F_HOST_SCREEN on page 637	Stores the screen name last recognized by the Get Host Screen action.
FINDBEST on page 637	Character variable used to help improve recognition accuracy in applications that use DPR or speech recognition.
HOURS_CLOSED on page 638	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 during business hours, this variable is set to zero, which is the default value.
LANGUAGE on page 638	A character string describing the language of the application.
NoDigit on page 639	Value of the number for a caller to press or speak signifying a "no" response.
PivotYear on page 640	Number used to help convert character variables to date variables; used to assign the date to the correct century.
RecordCounter on page 641	Used as a counter to track the number of records that match selected criteria. Each number in the final set represents a different record of all those found.
RecordsFound on page 641	Increments by 1 each time a record is found. The final value of the number is the total number of records found.
Time on page 642	Current time in hh24mmss format.
TRANSFER_RESULT on page 642	A character code representing the result of the last transfer attempt.
Tries on page 643	Represents the number of times to allow a caller to attempt a response to a prompt.
UNIX_TIME on page 644	Used to determine the current system date and time through the use of the external function u_datetime.
YesDigit on page 644	Value of the number for a caller to press or speak signifying a "yes" response.

Related topics:

[ANI \(Automatic Number Identification\)](#) on page 629
[Attempts](#) on page 630
[Channel](#) on page 630
[CI_MODE](#) on page 631
[CI_NO_DIGS_GOT](#) on page 632
[CI_RECOG](#) on page 632
[CollectedDigits](#) on page 633
[Date](#) on page 634
[Default](#) on page 634
[DNIS \(Dialed Number Identification Service\)](#) on page 635
[Errno](#) on page 635
[F_HOST_ERROR](#) on page 636
[F_HOST_SCREEN](#) on page 637
[FINDBEST](#) on page 637
[HOURS_CLOSED](#) on page 638
[LANGUAGE](#) on page 638
[NoDigit](#) on page 639
[PivotYear](#) on page 640
[RecordCounter](#) on page 641
[RecordsFound](#) on page 641
[Time](#) on page 642
[TRANSFER_RESULT](#) on page 642
[Tries](#) on page 643
[UNIX_TIME](#) on page 644
[YesDigit](#) on page 644

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.

Attribute	Default Value
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*

Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	30
Default	none*

* 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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* 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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	1*

* Indicates a value that you can edit or reset.

CI_MODE

As a variable, CI_MODE stores the SSP card number.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* Indicates a value that you can edit or reset.

CI_NO_DIGS_GOT

As a variable, CI_NO_DIGS_GOT is the default field for the number of input digits entered by the caller on the last input attempt for a Prompt and Collect node.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* Indicates a value that you can edit or reset.

CI_RECOG

As a variable, CI_RECOG is a number representing the recognizer used to identify a caller response. Possible values include the following:

- 0 – Touch Tone
- 1 – FlexWord
- 2 – WholeWord
- 3 – Dial Pulse

**Note:**

: This variable is set by the system and does not need to be changed by the user. This release of IVR Designer does not support WholeWord.

Default attributes are listed in the following table.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* 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.

Attribute	Default Value
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	68
Default	none*

* Indicates a value that you can edit or reset.

Date

As a variable, Date stores the date on which the call was received, in yyymmdd format.

When spoken by the CONVERSANT system, the collected value is spoken as M d, y. For example, 19980102 would be spoken as "January second (pause) nineteen ninety eight" unless you change the Spoken As field value.

Default attributes are listed in the following table.

Attribute	Default Value
Type	Date
Spoken as options for date variables on page 111	tts: ADMDYY* recorded: D*
Speak Using	current default*
Phrase Table	not applicable
Input As	mmddyy
Length	8
Default	date the application was created*

* Indicates a value that you can edit or reset.

Default

As a variable, Default stores caller input as a character string.

When spoken by the CONVERSANT or Avaya IR system, the collected values are spoken as character-based strings.

Default attributes are listed in the following table.

Attribute	Default Value
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*

Phrase Table	not applicable
Input As	char
Length	24
Default	none*

* 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
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	30
Default	none*

* Indicates a value that you can edit or reset.

Errno

As a variable, Errno stores a number. Errno can be assigned to return error codes or any other number value.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* Indicates a value that you can edit or reset.

F_HOST_ERROR

As a variable, F_HOST_ERROR stores the error code if the Get Screen action did not return a recognized screen.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10

Default	0*

* Indicates a value that you can edit or reset.

F_HOST_SCREEN

As a variable, F_HOST_SCREEN stores the screen name last recognized by the Get Host Screen action.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* Indicates a value that you can edit or reset.

FINDBEST

As a variable, FINDBEST is used to help improve recognition accuracy in applications that use DPR or speech recognition.

Default attributes are listed in the following table.

Attribute	Default Value
Type	character

Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	68
Default	none*

* 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 during business hours, this variable is set to zero, which is the default value.

Default attributes are listed in the following table.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* Indicates a value that you can edit or reset.

LANGUAGE

As a variable, LANGUAGE is a character string describing the language of the application.

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
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	15
Default	none*

* 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.

Attribute	Default Value
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	1
Default	2*

* Indicates a value that you can edit or reset.

PivotYear

PivotYear is used to designate a year which the CONVERSANT 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.

If the PivotYear value is set to "0" (the default), then the century is determined by the current date (century) setting for the CONVERSANT. In other words, before January 1, 2000, dates are assigned to the 20th century (for example, 1956). After January 1, 2000, dates are assigned to the 21st century (for example, 2056).

For example, suppose you received the character string, 012156, from a host screen, and you want to store and use it as a date variable. You have set the value of PivotYear to "60". Using a Set and Test node, you assign the character variable from the host to a date variable. Based upon the PivotYear setting, the date is thus interpreted as 20560121 (January 21, 2056).

Default attributes are listed in the following table.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	Not applicable
Input As	#
Length	10
Default	50*

* Indicates a value that you can edit or reset.

RecordCounter

As a variable, RecordCounter is used as a counter to track the number of records that match selected criteria. Each number in the final set represents a different record of all those found.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* Indicates a value that you can edit or reset.

RecordsFound

As a variable, RecordsFound increments by 1 each time a record is found. The final value of the number is the total number of records found.

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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*

Phrase Table	not applicable
Input As	#
Length	10
Default	0*

* 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 hhmm(AP)M. 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.

Attribute	Default Value
Type	Time
Spoken as options for time variables on page 115	tts: ATHMAM* recorded: T*
Speak Using	Current default*
Phrase Table	not applicable
Input As	hhmm(AP)M
Length	6
Default	Time the application was created*

* 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.

Attribute	Default Value
Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	3
Default	none*

* 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.

Attribute	Default Value
Type	number
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	#
Length	10

Default	3*

* Indicates a value that you can edit or reset.

UNIX_TIME

As a variable, UNIX_TIME can be used to determine the current system date and time through the use of the external function `u_datetime`.

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
Type	character
Spoken as options for number variables on page 112	tts: AN* recorded: N*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	10
Default	time the variable was encountered in the application

* 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 or Avaya IR system, the value is spoken as a digit.

Default attributes are listed in the following table.

Attribute	Default Value

Type	character
Spoken as options for character variables on page 108	tts: A* recorded: C*
Speak Using	current default*
Phrase Table	not applicable
Input As	char
Length	1
Default	1*

* Indicates a value that you can edit or reset.

Standard external functions

IVR Designer includes, as part of the standard package, a variety of external functions that you can use in developing your applications. These are known as standard external functions (or system external functions). For more information about external functions, refer to the Application Development with Script Builder document for your voice response system, and [Using external functions in TAS applications](#) on page 456.

External functions are grouped in the help system and on the External Functions Palette according to general functionality. The following table lists the external functions in alphabetical order, the tab or group name for each external function, and the software package name and type. For package type, base means that the external function is included with the base software that comes with the voice response system, licensed means that you have to pay a separate licensing fee to use the software package.

External function	Tab name	Package name	Package type
A_Callinfo	ASAI external functions on page 658	CTIDIP, AVctidip*	Licensed
A_Event	ASAI external functions on page 658	CTIDIP, AVctidip*	Licensed
A_RouteSel	ASAI external functions on page 658	CTIDIP, AVctidip*	Licensed

A_Trn	ASAI external functions on page 658	CTIDIP, AVctidip*	Licensed
AllocPROXY	NLSR external functions on page 726	SRproxy, AVsrproxy*	Licensed
Attr_ANI	PRI external functions on page 725	PRI, AVnms*	Licensed
Attr_busy	PRI external functions on page 725	PRI, AVnms*	Licensed
Attr_ring	PRI external functions on page 725	PRI, AVnms*	Licensed
Attr_silence	PRI external functions on page 725	PRI, AVnms*	Licensed
Background	Phrase external functions on page 724	vs	Base
Background_Tag	Phrase external functions on page 724	vs	Base
BT_Concat5	FAX external functions on page 696	BrookTrout	Licensed
BT_Exec_UNIX	FAX external functions on page 696	BrookTrout	Licensed
BT_FAX_CNG	FAX external functions on page 696	BrookTrout	Licensed
BT_FAX_Combine	FAX external functions on page 696	BrookTrout	Licensed
BT_FAX_CovrPage	FAX external functions on page 696	BrookTrout	Licensed

BT_FAX_Current	FAX external functions on page 696	BrookTrout	Licensed
BT_FAX_Get	FAX external functions on page 696	BrookTrout	Licensed
BT_FAX_Send	FAX external functions on page 696	BrookTrout	Licensed
Call_Bridge	Telephony external functions on page 732	xferdip	Base
Callprg	Control external functions on page 665	sc, AVsc*	Base
CINFO	PRI external functions on page 725	PRI, AVnms*	Licensed
Clear_ucid	UCID external functions on page 699	ASAI and UCID, AVasai* and AVctidip*	Base
Complete	Telephony external functions on page 732	vs, AVir*	Base
Concat5	FAX external functions on page 696	sbcscfax, AVnmsfax*	Licensed
Conv_data	Telephony external functions on page 732	vs, AVir*	Base
CtiCallInfo	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiCallstate	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiConfer	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed

CtiDial	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiDisconn	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiHold	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiNotify	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiPrivData	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiRetrieve	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
CtiTransfer	CTI DIP external functions on page 671	CTIDIP, AVctidip*	Licensed
Datetime_u	Date/Time external functions on page 681	vs, AVir*	Base
DeallocPROXY	NLSR external functions on page 726	SRproxy, AVsrproxy*	Licensed
DecodeURL	Vonetix external functions on page 733	Vonetix	Base
DoNotCall	Dialer external functions on page 682	Dialer	Licensed
DPR_Disable	Speech Features external functions on page 729	dpr, AVdpr*	Base
DPR_Reset	Speech Features external functions on page 729	AVdpr*	Base

EMPAC	NICE external functions on page 714	NICE	Base
EMStart	NICE external functions on page 714	NICE	Base
EMStop	NICE external functions on page 714	NICE	Base
EncodeURL	Vonetix external functions on page 733	Vonetix	Base
End_softdisc	Dialer external functions on page 682	Dialer	Licensed
Exec_UNIX	FAX external functions on page 696	sbcscfax, AVnmsfax*	Licensed
Execute	Control external functions on page 665	vs, AVir*	Licensed
FAX_CNG	FAX external functions on page 696	sbcscfax	Licensed
FAX_CovrPage	FAX external functions on page 696	sbcscfax	Licensed
FAX_Get	FAX external functions on page 696	sbcscfax, AVnmsfax*	Licensed
FAX_Queue	FAX external functions on page 696	sbcscfax, AVnmsfax*	Licensed
FAX_Send	FAX external functions on page 696	sbcscfax, AVnmsfax*	Licensed
FF_Code	Form Filler external functions on page 699	Form Filler	Licensed

FF_Store	Form Filler external functions on page 699	Form Filler	Licensed
FinishedItem	Dialer external functions on page 682	Dialer	Licensed
Get_ucid	UCID external functions on page 699	ASAI and UCID, AVasai* and AVctidip*	Licensed
Get_uui	UCID external functions on page 699	ASAI and UCID, AVasai* and AVctidip*	Licensed
Getarg	Control external functions on page 665	vs, AVir*	Licensed
Getday	Date/Time external functions on page 681	sc, AVsc*	Base
GetdcPROXY	NLSR external functions on page 726	SRproxy	Licensed
Handle_disc	Dialer external functions on page 682	Dialer	Licensed
HookflashLine	Dialer external functions on page 682	Dialer	Licensed
IcrTrans	GeoTel external functions on page 700	Avaya	Licensed
ISDN_billing	PRI external functions on page 725	PRI, AVnms*	Licensed
ISDN_service	PRI external functions on page 725	PRI, AVnms*	Licensed
Ixfer	Telephony external functions on page 732	vs, AVir*	Licensed

ListCbackFmt	Dialer external functions on page 682	Dialer	Licensed
LogloStart	Dialer external functions on page 682	Dialer	Licensed
LogloStop	Dialer external functions on page 682	Dialer	Licensed
LreqPROXY	NLSR external functions on page 726	SRproxy, AVsrproxy*	Licensed
NMSPlayGain	Control external functions on page 665	vs, AVir*	Licensed
Make_Call	Telephony external functions on page 732	vs, AVir*	Licensed
MoFlashBlind	Dialer external functions on page 682	Dialer	Licensed
Mon_year	Date/Time external functions on page 681	sc, AVsc*	Base
MrcpSetParam	NLSR external functions on page 726	SRproxy, AVsrproxy*	Licensed
Msg_Code	Phrase external functions on page 724	vs, AVir*	Base
Msg_Code_Tag	Phrase external functions on page 724	vs	Base
Msg_Delete	Phrase external functions on page 724	vs, AVir*	Base
Msg_Delete_Tag	Phrase external functions on page 724	vs	Base

MX_GetData	CRM Central external functions on page 671	CRM	Base
MX_SendData	CRM Central external functions on page 671	CRM	Base
Pack_phrNX	Phrase external functions on page 724	vs, AVir*	Base
Proxylang	Speech Features external functions on page 729	TTSproxy, AVttsprxy*	Licensed
Proxytext	Speech Features external functions on page 729	TTSproxy, AVttsprxy*	Licensed
Proxytts_on	Speech Features external functions on page 729	TTSproxy, AVttsprxy*	Licensed
Proxyvoice	Speech Features external functions on page 729	TTSproxy, AVttsprxy*	Licensed
ReadField	Dialer external functions on page 682	Dialer	Licensed
Reconnect	Telephony external functions on page 732	vs	Licensed
ReleaseLine	Dialer external functions on page 682	Dialer	Licensed
ReleaseLU	CLEO external functions on page 658	CLEO	Licensed
RequestPROXY	NLSR external functions on page 726	SRproxy, AVsrproxy*	Licensed
ReserveLU	CLEO external functions on page 658	CLEO	Licensed

Ret2vxml	Control external functions on page 665	AVvoicxml*	Licensed
RmdcPROXY	NLSR external functions on page 726	SRproxy	Licensed
RunTransaction	CLEO external functions on page 658	CLEO	Licensed
SendMessage	Dialer external functions on page 682	Dialer	Licensed
SetCallback	Dialer external functions on page 682	Dialer	Licensed
Set_early_hu	Dialer external functions on page 682	Dialer	Licensed
SetNotifyFld	Dialer external functions on page 682	Dialer	Licensed
Set_softdisc	Dialer external functions on page 682	Dialer	Licensed
Set_uui	UCID external functions on page 699	ASAI and UCID, AVasai* and AVctidip*	Licensed
SetCallback	Dialer external functions on page 682	Dialer	Licensed
Sleep	Control external functions on page 665	sc, AVsc*	Base
SP_Allocate	Speech Features external functions on page 729	Speech Features	Base
SR_Allocate	Speech Features external functions on page 729	Speech Features	Base

SR_Prompt	Speech Features external functions on page 729	Speech Features	Base
StartRecording	NICE external functions on page 714	NICE	Base
StopRecording	NICE external functions on page 714	NICE	Base
StoredProc	Control external functions on page 665	sc, AVsc*	Base
StoredProcNext	Control external functions on page 665	sc, AVsc*	Base
Subprogram	Control external functions on page 665	sc, AVsc*	Base
SWIttsSetup	Speech Features external functions on page 729	TTSproxy, AVttsprxy*	Licensed
Tbct_getinfo	PRI external functions on page 725	tbct, AVtbct*	Licensed
Tbct_Xfer	PRI external functions on page 725	tbct, AVtbct*	Licensed
Text2fax	FAX external functions on page 696	sbcxfax, AVnmsfax*	Licensed
Transfera	Telephony external functions on page 732	vs	Base
Transferb	Telephony external functions on page 732	vs, AVir*	Base
Tts_file	Speech Features external functions on page 729	TTSproxy	Licensed

U_datetime	Date/Time external functions on page 681	vs, AVir*	Base
Unpack_phrNX	Phrase external functions on page 724	vs, AVir*	Base
UpdateField	Dialer external functions on page 682	Dialer	Licensed
Vesp_dip	Interaction Center external functions on page 700	Interaction Center	Licensed
VntxAddAttr	Vonetix external functions on page 733	Vonetix	Base
VntxAddChild	Vonetix external functions on page 733	Vonetix	Base
VntxAddSibl	Vonetix external functions on page 733	Vonetix	Base
VntxClearDoc	Vonetix external functions on page 733	Vonetix	Base
VntxClearFD	Vonetix external functions on page 733	Vonetix	Base
VntxClearSes	Vonetix external functions on page 733	Vonetix	Base
VntxDBExec	Vonetix external functions on page 733	Vonetix	Base
VntxDBNxtRow	Vonetix external functions on page 733	Vonetix	Base
VntxDBQuery	Vonetix external functions on page 733	Vonetix	Base

VntxDBXCols	Vonetix external functions on page 733	Vonetix	Base
VntxDelTag	Vonetix external functions on page 733	Vonetix	Base
VntxFeature	Vonetix external functions on page 733	Vonetix	Base
VntxGetAttr	Vonetix external functions on page 733	Vonetix	Base
VntxGetDoc	Vonetix external functions on page 733	Vonetix	Base
VntxGetNumAI	Vonetix external functions on page 733	Vonetix	Base
VntxGetNumTg	Vonetix external functions on page 733	Vonetix	Base
VntxGetNumTk	Vonetix external functions on page 733	Vonetix	Base
VntxGetTag	Vonetix external functions on page 733	Vonetix	Base
VntxGetToken	Vonetix external functions on page 733	Vonetix	Base
VntxInit	Vonetix external functions on page 733	Vonetix	Base
VntxLoadDoc	Vonetix external functions on page 733	Vonetix	Base
VntxNewDoc	Vonetix external functions on page 733	Vonetix	Base

VntxParse1	Vonetix external functions on page 733	Vonetix	Base
VntxPostDoc	Vonetix external functions on page 733	Vonetix	Base
VntxScanDoc	Vonetix external functions on page 733	Vonetix	Base
VntxSetAttr	Vonetix external functions on page 733	Vonetix	Base
VntxSetDocFD	Vonetix external functions on page 733	Vonetix	Base
VntxSetTag	Vonetix external functions on page 733	Vonetix	Base
VntxStoreDoc	Vonetix external functions on page 733	Vonetix	Base

* Avaya Interactive Response package name

Related topics:

[CLEO external functions](#) on page 658
[ASAI external functions](#) on page 658
[Control external functions](#) on page 665
[CRM Central external functions](#) on page 671
[CTI DIP external functions](#) on page 671
[Date/Time external functions](#) on page 681
[Dialer external functions](#) on page 682
[FAX external functions](#) on page 696
[Form Filler external functions](#) on page 699
[UCID external functions](#) on page 699
[GeoTel external functions](#) on page 700
[Interaction Center external functions](#) on page 700
[NICE external functions](#) on page 714
[Phrase external functions](#) on page 724
[PRI external functions](#) on page 725
[NLSR external functions](#) on page 726

[Speech Features external functions](#) on page 729

[Telephony external functions](#) on page 732

[Vonetix external functions](#) on page 733

CLEO external functions

The following table describes the CLEO external functions.

Name	Description
RunTransaction	Invokes a CLEO 3270 Plugin transaction.
ReserveLU	Acquires a parked CLEO 3270 Logical Unit (LU) for a host application. A successful call to this function results in the LU being exclusively associated with the call channel and call script until a ReleaseLU function is invoked. The CLEO 3270 Plugin remembers which LU is assigned, therefore, no "handle" needs to be returned.
ReleaseLU	Releases a previously reserved CLEO 3270 LU from a host application. The CLEO 3270 Plugin remembers which LU is assigned to the call channel and call script, therefore no argument is required.

ASAI external functions

The following table describes the external functions that can be used in conjunction with the ASAI feature on the target system.

**Note:**

: The external functions in this group will work on the target system only if you have purchased the ASAI feature.

For more information about a particular external function, refer to that topic.

Name	Description

A_Callinfo on page 659	Used to access call information, such as ANI and DNIS, obtained from ASAI for a call on a telephony channel on the target system.
A_Event on page 661	Used to retrieve information related to a call being monitored by an ASAI domain.
A_RouteSel on page 662	Used in conjunction with the A_Event external function to send an ASAI route message to the PBX.
A_Tran on page 663	Used by voice applications running on an ASAI interface to transfer a call.

For more information about using ASAI external functions in voice applications, see your CONVERSANT product documentation or the Avaya IR System Help.

Related topics:

[A_Callinfo](#) on page 659

[A_Event](#) on page 661

[A_RouteSel](#) on page 662

[A_Tran](#) on page 663

A_Callinfo

The A_Callinfo function accesses call information, such as ANI and DNIS, obtained from ASAI for a call on a telephony channel on the target system.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code.
Calling Number	Select the variable in which to store the Calling Party Number (up to 30 characters). The variable type must be character. For incoming ISDN calls, this is the Automatic Number Identification (ANI) for the call, if available.
Called Number	Select the variable in which to store the Called Party Number (up to 30 characters). The variable type must be character. For incoming ISDN calls, this

	is the Dialed Number Identification Service (DNIS) for the call, if available.
Switch Data	Select the variable in which to store switch-collected touch-tone data. The variable type must be character.
Trunk Group Id	Select the variable in which to store the trunk group identification. The variable type must be number.
Call Id	Select the variable in which to store the call identification. The variable type must be number.
Cause Value	Select the variable in which to store the cause value if an error occurs. The variable type must be number.
Ucid	Select the variable in which to store the Universal Call Identification (UCID). The variable type must be character. UCID is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
Uui	Select the variable in which to store the User-to-User Information (UUI). The variable type must be character. UUI is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
Ani II Digits	Select the variable in which to store the ANI Information Indicator (ANI II) digits. The variable type must be number. ANI II Digits is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
Redirecting Number	Select the variable in which to store the network (ISDN) Redirecting Number. The variable type must be character. Redirecting Number is supported on CONVERSANT R9, and on Avaya Interactive Response.

A_Event

The A_Event function retrieves information related to a call being monitored by an ASAI domain.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code.
Connected Num	Select the variable in which to store the connected party number. The variable type must be character.
Calling Num	Select the variable in which to store the Calling Party Number (up to 30 characters). The variable type must be character. For incoming ISDN calls, this is the Automatic Number Identification (ANI) for the call, if available.
Called Num	Select the variable in which to store the Called Party Number (up to 30 characters). The variable type must be character. For incoming ISDN calls, this is the Dialed Number Identification Service (DNIS) for the call, if available.
Switch Data	Select the variable in which to store switch-collected touch-tone data. The variable type must be character.
Trunk Group Id	Select the variable in which to store the trunk group identification.
Call Id	Select the variable in which to store the call identification.
Other Call Id	Select the variable in which to store the call identification from a conference or transfer.
LAI Display Info	Select the variable in which to store the Look-Ahead Interflow (LAI) information for the call. The variable type must be character.

VIS Data	Select the variable in which to store data saved by the application that previously handled the call. The variable type must be character.
Routing ID	Select the variable in which to store the unique tag for a routing request. The variable type must be number.
Cause Value	Select the variable in which to store the cause value if an error occurs. The variable type must be number.
Ucid	Select the variable in which to store the Universal Call Identification (UCID). The variable type must be character. UCID is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
Uui	Select the variable in which to store the User-to-User Information (UUI). The variable type must be character. UUI is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
ANI II Digits	Select the variable in which to store the ANI II digits. The variable type must be number. ANI II Digits is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
Redirecting Number	Select the variable in which to store the network (ISDN) Redirecting Number. The variable type must be character. Redirecting Number is supported on CONVERSANT R9, and on Avaya Interactive Response.

A_RouteSel

The A_RouteSel function is used in conjunction with the A_Event function to send an ASAI route message to the PBX.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code.
Destination No	Type or select the variable in which to store the number (up to 20 characters) to which to transfer the call. The variable type must be character.
Split Ext	Type or select the variable in which to store the number (up to 5 characters) for a Direct Agent call. The variable type must be character.
Priority	Enter Yes for a priority call, No for a regular call. The variable type must be character.
Routing Id	Type or select the variable in which to store the unique tag for a routing request. The variable type must be number.
Cause Value	Specify the variable in which to store the cause value if an error occurs. The variable type must be number.
Uui	Type or select the variable in which to store the User-to-User Information (UUI). The variable type must be character. If you enter text for this attribute, do not use double quotes ("). UUI is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.

A_Tran

The A_Tran function is used by voice applications running on telephony channels on the target system to transfer a call.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description

Put Return Code In	Select the variable in which to store the return code.
Destination No	Type or select the variable in which to store the number (up to 20 characters) to which to transfer the call. For Avaya IR Release 1.2 or later, the destination number can be a Voice over IP (VoIP) station.
Split Ext	Type or select the variable in which to store the number (up to 5 characters) for doing a Direct Agent call.
Priority	Enter Yes for a priority call, No for a regular call.
VIS Data	Type or select the variable in which to store up to 20 characters of data that is saved and used later by the A_Event external function.
Call State	Select the variable in which to store the call state. The variable type must be number.
Cause Value	Select the variable in which to store the cause value if an error occurs. The variable type must be number.
Ucid	Select the variable in which to store the Universal Call Identification (UCID). The variable type must be character. UCID is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.
Uui	Type or select the variable in which to store the User-to-User Information (UUI). The variable type must be character. If you enter text for this attribute, do not use double quotes ("). UUI is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.

Control external functions

The following table describes the Control external functions that can be used in IVR Designer applications.

Name	Description
Callprg	Calls an external program written in C, Java, or UNIX.  Note: For more information about Callprg, see Callprg on page 667.
Execute	Executes another application on the current channel and returns control. For more information about the Execute external function, see Using the Execute external function on page 465.  Note: The Execute external function is similar to the Save and Exec node. The differences lie in the way the two pass arguments and variables to the new application. When calling a Script Builder application, you must use the Execute external function because Script Builder does not have Load Variables node functionality or any equivalent action. For IVR Designer applications, you can use either the Save and Exec node or the Execute external function.
NMSPlayGain	Adjusts the play back volume control within a TAS application. This feature is available in IVR Designer Release 5.3 and later.
Getarg	When a TAS script is called from a Script Builder application, Getarg can be used in conjunction with the Execute and Subprogram external functions to extract an argument after the Execute function completes. When a TAS script is called from a VoiceXML application, Getarg can be

	used to collect variable data passed to it by a CallScript node in a VoiceXML application. For more information about the Getarg external function, see Using the Execute and Getarg external functions on page 468.
Ret2vxml	Ret2vxml can be used to return variable data and control from a TAS script to a CallScript node in a VoiceXML application.
Sleep	Sleeps or pauses the execution of the application for the specified number of seconds.
StoredProc	StoredProc can be used to execute a stored procedure created on a database and retrieve the result set (values) returned by the stored procedure. For more information about StoredProc, see StoredProc on page 668.  Note: This external function is supported only on Avaya IR.
StoredProcNext	StoredProcNext can be used to retrieve the next (second or greater) record of a multiple-record result set returned by a stored procedure. StoredProcNext can be used only after a StoredProc external function. For more information about StoredProcNext, see StoredProcNext on page 670.  Note: This external function is supported only on Avaya IR.
Subprogram	References an external subprogram on the target CONVERSANT or Avaya Interactive Response system. The referenced subprogram must be compiled and installed on the same target system on which the IVR

	Designer application that references the subprogram is installed. The Subprogram external function may return the following error messages: • -1 – Subprogram not installed (the specified subprogram could not be found) • -2 – No space error (the application may be trying to pass too much data)  Note: The Subprogram external function is similar to the Save and Exec node. The differences lie in the way the two pass arguments and variables to the new application. When calling a Script Builder application, you must use the Subprogram external function because Script Builder does not have Load Variables node functionality or any equivalent action. For IVR Designer applications, you can use either the Save and Exec node or the Subprogram external function.

Related topics:

[Callprg](#) on page 667

[NMSPlayGain](#) on page 668

[StoredProc](#) on page 668

[StoredProcNext](#) on page 670

Callprg

Calling a C or UNIX program

To call a C or UNIX program, the parameters should be as follows:

- ProgramName, use: /path/ProgramName
- (optional) Arguments, use: Argument1 Argument2 ... ArgumentN
- (optional) RetValue should be a number to hold the exit or return code from the program. The maximum value the program can return is 255.

- (optional) RetString should be a character string to hold the standard output data from the program.
- (optional) RetLength should be the maximum length of the RetString if used (maximum value 2000 characters).

Calling a Java program

The Java runtime environment is an executable program. To call a Java program, the parameters should be as follows:

- ProgramName, use: /path/to/java
- Arguments, use: -classpath classpath/to/the/class/files JavaClass Argument1 Argument2 ... ArgumentN
- RetValue is not applicable for Java programs.
- (optional) RetString should be a character string to hold the standard output data from the Java program.
- (optional) RetLength should be the maximum length of the RetString (maximum value 2000 characters).

NMSPlayGain

NMSPlayGain adjusts the play back volume control within a TAS application.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Volume Level	Use the variable to set the volume level for playback of a recorded voice message. The variable can take a minimum value of -54 to a maximum value of +24. The units of the values are decibels (db). The variable type must be number. The return codes are: 0 = success lesser than 2 = wrong values for volume level -3 = IRAPI error

StoredProc

StoredProc can be used to execute a stored procedure created on a database and retrieve the result set (values) returned by the stored procedure.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code. The return codes are: -3 = invalid argument -2 = the database DIP is not running -1 = an exception occurred 0 = no records found - greater than 0 = success
SPNAME	Type or select a variable that contains the name of the stored procedure to execute. The variable type must be character.
PARMS	Type or select a variable that contains a comma-separated string of parameters to send to and receive from the stored procedure. The variable type must be character. The parameter string contains arguments that specify input parameters, output parameters, or both. Each argument has three parts: parameter type, value, and argument type. The parameter type can be INT (integer), STRING (character string), DATE, TIME, or TIMESTAMP. The argument type can be IN (input only), OUT (output only), or INOUT (input and output). The following is an example of a parameter string with two arguments. The first argument sends the integer 1234. The second argument receives the character string ABCD. Example: INT, 1234, IN, STRING, ABCD, OUT
DIPNAME	Type or select a variable that contains the name of the database DIP on the target platform that is configured to access the database with the stored procedure specified in the SPNAME attribute. The variable type must be character.

	The values for DIPNAME are: dip1, DBDIP2, DBDIP3, DBDIP4, DBDIP5.
SPRETCODE	If the stored procedure returns a numeric return code, type or select a variable that contains the value YES. If the stored procedure does not return a numeric return code, type or select a variable that contains the value NO. The variable type must be character.
SPRESULTSET	If the stored procedure returns a result set, type or select a variable that contains the value YES. If the stored procedure does not return a result set, type or select a variable that contains the value NO. The variable type must be character.
RESULT	Select the variable in which to store the returned result set from the stored procedure. The variable type must be character.

StoredProcNext

StoredProcNext can be used to retrieve the next (second or greater) record of a multiple-record result set returned by a stored procedure. StoredProcNext can be used only after a StoredProc external function.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code. The return codes are: -3 = invalid argument -2 = the database DIP is not running -1 = an exception occurred 0 = no records found greater than 0 = success
RESULT	Select the variable in which to store the returned result set from the stored procedure. The variable type must be character.

CRM Central external functions

The following table describes the CRM Central external functions that can be used in IVR Designer applications.

Name	Description
MX_GetData	Used to get character parameter data from CRM Central.  Note: This external function is currently not supported on Avaya IR.
MX_SendData	Used to send character parameter data to CRM Central after the custom application on the CONVERSANT has finished. This external function also returns a code ranging from 201 through 999 indicating a user-defined status.  Note: This external function is currently not supported on Avaya IR.

CTI DIP external functions

The following table describes the external functions that can be used in conjunction with the CTI DIP. For more information about a particular external function, refer to that topic.

Name	Description
CtiCallInfo on page 674	Identifies the call ID of a call that has reached the CONVERSANT or Avaya IR. Identifies the number that was dialed by the caller: if the call was inbound from an ISDN trunk group, it also identifies the ANI and DNIS associated with the call.
CtiCallState on page 675	Used to determine the call ID and associated state (held, established, etc.) of the calls present at the port. The

	states are represented by numeric values.
CtiConfer on page 676	Requests that two calls present at a CONVERSANT or Avaya IR port be conferenced together. It requires that one of the calls involved in the transfer be on hold at the CONVERSANT or Avaya IR port prior to execution.
CtiDial on page 677	Requests a call be placed from the CONVERSANT or Avaya IR port to a destination. One of the calls involved in the transfer be on hold at the CONVERSANT or Avaya IR port prior to execution.
CtiDisconn on page 678	Requests a call be placed from the CONVERSANT or Avaya IR port to disconnect a call. This function is best used in conjunction with the CtiCallState function to determine the state of a call before it is disconnected. Also, the CONVERSANT or Avaya IR port will not detect any dial tone on the line after the call is disconnected, so if the application needs to terminate beforehand, it must explicitly do so after a successful return.
CtiHold on page 678	Requests that a call that is active at the CONVERSANT or Avaya IR port be placed on hold. This function is best used in conjunction with the CtiCallState function to determine the state of a call before it is placed on hold. If an error is encountered, the CTI DIP will populate the Error Code field.
CtiNotify	Provides answer notification to a voice application that has launched a call. The CtiNotify function is most useful in outbound calling when used in conjunction with the CtiDial function. The function depends on receiving answer supervision on a timely basis from the network. Therefore, if the call is sent over a non-ISDN trunk to a destination, the function may return

	several seconds after the call is actually connected. When the CtiNotify request is made a message is sent to the CTI DIP. The CTI DIP will check the status of the calls currently at the CONVERSANT or Avaya IR port. If a call is found to be either in the DELIVERED or ORIGINATED state, the CTI DIP will tag this call to receive answer notification. If no call is currently in either the DELIVERED or ORIGINATED state, the CTI DIP will return a "-1" to the CtiNotify external function. Otherwise, the CTI DIP will track the call that has been tagged. When the tracked call is answered, the CTI DIP will send the call ID of the answered call to the application. If the CtiNotify function receives a "-1", it will try up to 3 times to find a call at that CONVERSANT or Avaya IR port that is in either the DELIVERED or ORIGINATED state. There will be a 500-millisecond pause between attempts.
CtiPrivData on page 679	Requests that any private data associated with a call at the CONVERSANT or Avaya IR port be provided. Currently this function supports User-to-User Information (UUI), and Universal Call ID (UCID).
CtiRetrieve on page 680	Requests that a held call at the CONVERSANT or Avaya IR port be retrieved. This function is typically used only to recover from an error, such as dialing a busy number, and the application needs to reconnect to the held party to inform them of the busy party.
CtiTransfer on page 681	Requests that two calls present at a CONVERSANT or Avaya IR port be transferred together. This function requires that one of the calls involved in the transfer be on hold at the CONVERSANT or Avaya IR port prior to the execution of this function.

Related topics:

[CtiCallInfo](#) on page 674
[CtiCallState](#) on page 675
[CtiConfer](#) on page 676
[CtiDial](#) on page 677
[CtiDisconn](#) on page 678
[CtiHold](#) on page 678
[CtiPrivData](#) on page 679
[CtiRetrieve](#) on page 680
[CtiTransfer](#) on page 681

CtiCallInfo

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the CTI DIP found a call at the CONVERSANT or Avaya IR port • a value less than or equal to zero indicates that the CTI DIP did not find a call at the CONVERSANT or Avaya IR port
Call ID	Select the variable in which to store the call ID. The variable type must be number.
ANI	Select the variable in which to store the ANI for the call (up to 30 characters). The variable type must be character.
Called Number	Select the variable in which to store the Called Number or DNIS (up to 30 characters). The variable type must be character.

Station ID	Select the variable in which to store the CONVERSANT or Avaya IR port extension. (up to 7 characters). The variable type must be character.
Skill	Select the variable in which to store the skill/split hunt-group number. (up to 7 characters). The variable type must be character.

CtiCallState

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: - the call ID of call #1 indicates that the CTI DIP found a call at the CONVERSANT or Avaya IR port - a value of -1 indicates that the CTI DIP did not find a call at the CONVERSANT or Avaya IR port
Call ID 2	Select the variable in which to store the call ID of call #2. The variable type must be number.
Call ID 3	Select the variable in which to store the call ID of call #3. The variable type must be number.
State 1	Select the variable in which to store the state of call #1 (see Call state values). The variable type must be number.
State 2	Select the variable in which to store the state of call #2 (see Call state values). The variable type must be number.

State 3	Select the variable in which to store the state of call #3 (see Call state values). The variable type must be number.

Call state values

A CONVERSANT or Avaya IR port can support up to 3 simultaneous calls. This external function can be used to determine the call ID and associated state of the calls present at the port. The states are represented by the following values:

Value	State
0	None
1	Delivered
2	Established
3	Held
4	Conferenced
5	Transferred
6	Initiated
7	Originated



Note:

The ordering of the calls is determined by the arrival order. Call #1 is most likely the first call received by the CONVERSANT or Avaya IR.

CtiConfer

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description

Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the call conference was successfully • a value less than or equal to zero indicates that the call conference failed
Held Call ID	Select the variable in which to store the call ID of the held call. The variable type must be number.
Active Call ID	Select the variable in which to store the call ID of the active call. The variable type must be number.

CtiDial

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the call was dialed successfully • a value less than or equal to zero indicates that the dialing attempt failed
Phone Number	Type or select the variable that contains the destination number to be dialed (up to 64 bytes). The variable type must be character.

User-User Info	Type or select the variable that contains the UUI (up to 96 bytes). The variable type must be character. If you enter text for this attribute, do not use double quotes ("). UUI is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.

CtiDisconn

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the call was successfully disconnected • a value less than or equal to zero indicates that the call disconnect failed
Call ID	Select the variable in which to store the call ID. The variable type must be number.

CtiHold

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description

Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the call was successfully placed on hold • a value less than or equal to zero indicates that the call hold attempt failed
Call ID	Select the variable in which to store the call ID. The variable type must be number.
Error Code	Select the variable in which to store the error code (CSTA_Universal_Failure_t) if the call hold attempt fails. The variable type must be number.

CtiPrivData

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • the value 1 indicates that the CTI DIP successfully retrieved the private data • a value less than or equal to zero indicates that no private data exists
Private Data	Select the variable in which to store private data (up to 96 characters) associated with the call. The variable type must be character.

Type	Select the variable in which to store a value that represents the type of private data retrieved. The values are: • 0 = UUI • 1 = II DIGITS (not currently implemented) • 2 = UCID The variable type must be number.
Call ID	Select the variable in which to store the call ID. The variable type must be number.
Vendor	Select the variable in which to store the name of the PBX vendor (up to 32 characters). The variable type must be character.

CtiRetrieve

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the call was successfully retrieved, and that the CTI DIP received a CSTA_RETRIEVE_CALL_CONFIRM event • a value less than or equal to zero indicates that the call retrieval failed
Call ID	Select the variable in which to store the call ID. The variable type must be number.

Error Code	Select the variable in which to store the error code (CSTA_UNIVERSAL_FAILURE_CONF) if the call retrieval fails. The variable type must be number.

CtiTransfer

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select the variable in which to store the return code from CONVERSANT or Avaya IR. The return code can have one of the following values: • a value greater than zero indicates that the call transfer was successfully, and that the CTI DIP received a CSTA_TRANSFERRED_CALL_CONF event • a value less than or equal to zero indicates that the call transfer failed
Held Call ID	Select the variable in which to store the call ID of the held call. The variable type must be number.
Active Call ID	Select the variable in which to store the call ID of the active call. The variable type must be number.

Date/Time external functions

The following table describes the external functions that can be used in conjunction with the Date and Time features on the target system.

Name	Description

Datetime_u	Converts a date field and a time field to UNIX date&time, computing the internal UNIX time. For more information about the Mon_year external function, see Using the Datetime_u and Getday external functions on page 461.
U_datetime	Converts UNIX date&time to a date field and a time field by converting the internal UNIX clock value.
Getday	Converts UNIX time to a day of the week and returns a numeric field which corresponds to the appropriate day (1 is Sunday, 2 is Monday, and so on).
Mon_year	Speaks the date as a month and, optionally, the year as a two-digit or four-digit year. For more information about the Mon_year external function, see Using the Mon_year external function on page 464.

Dialer external functions

The following table describes the Avaya Predictive Dialing System (PDS), also called Dialer, external functions that can be used in IVR Designer applications.

The Set_Softdisc, Handle_Disc, and End_Softdisc external functions are required to handle caller disconnects, to provide continuity within job execution, and to disconnect the call and end processing. All other Dialer external functions provide optional functionality.

 Note:

Note: The IVR Designer application should never hang up the line or perform a quit instruction. The system will recover, but this leads to great inefficiencies in the processing of jobs.

For more information about a particular external function, refer to that topic.

Name	Description
DoNotCall external function on page 685	Identifies a customer record as "do not call" (DNC). All matching records on other calling lists are also marked DNC. The DNC feature ensures that records

	appearing in multiple calling lists will not be recalled.
End_softdisc external function on page 686	Ends handling a soft disconnect and re-executes the agt_chl application for the next call. See the descriptions for the Handle_disc and Set_Softdisc external functions.
FinishedItem external function on page 686	Releases a customer record and inserts a call completion code into the record. This must be the last activity performed by the script, since the Dialer sends another call after receiving it.
Handle_disc external function on page 687	Sets the label for handling a soft disconnect. This function marks the beginning of the event-handling code. Between Handle_disc and End_softdisc, place whatever code you think is necessary to clean up the call, for example, updating the completing code on the Dialer system. See the descriptions for the End_softdisc and Set_Softdisc external functions.
HookflashLine external function on page 687	Transfers the current call by way of a switch that accepts hookflash-initiated transfers. This external function disconnects the transferring application before transferring the call.
ListCbackFmt external function on page 688	Returns the date format from the caller record. It also lists the number of telephone numbers in the customer record.
LogloStart external function on page 689	Turns on message-logging on the dialer to trace call transactions. Creates a file on the Dialer named /tmp/ AgentName _API.trans. It contains all the messages between the agent application and the Dialer. Use this file to verify message content and to track the agent application session. See the description for the LogloStop external function.
LogloStop external function on page 689	Turns off message-logging on the dialer. Stops the agent binary from writing to the /tmp/ AgentName _API.trans. Use

	this external function to terminate the log file. See the description for the LogloStart external function.
MoFlashBlind external function on page 690	Transfers the current call and data record to another job on the Dialer. This external function disconnects the transferring application before transferring the call and customer record. The Dialer places the customer call in the queue for the new job or connects the customer to an available agent. If the Dialer uses ANI/DNIS information, the transferred call carries the outbound number the Dialer dials as the ANI and the original outbound job name as the DNIS. The inbound or blend job receiving the call sees the call as an inbound call.
ReadField external function on page 691	Reads a field from the Dialer customer record.
ReleaseLine external function on page 691	Releases the telephone line but allows further updating of the customer record.
SendMessage external function on page 692	Sends a message to the Dialer supervisor's screen.
SetCallback external function on page 693	Sets the time on the Dialer to schedule a customer callback.
Set_early_hu external function on page 694	Prepares the application for an early hangup. The sim_agt DIP posts a hangup event to the application when the Dialer detects a hangup. The application must define an L_EARLYHANGUP label. The code associated with the L_EARLYHANGUP label should perform the processing that is required when an early hangup event is received.
SetNotifyFld external function on page 694	Sets the system to pass the value of a specified key field from the Dialer calling list with the CallNotification message.

	 Note: The SetNotifyFld function should be used only in an initial application to be run when an agent joins a job.
Set_Softdisc external function on page 695	Prepares the application for a soft disconnect. This function catches the dipterm event and runs the handler. This should be the first external function in the application. See the descriptions for the Handle_disc and End_Softdisc external functions.
UpdateField external function on page 695	Updates a data field on the Dialer customer record.

Related topics:

[DoNotCall external function](#) on page 685
[End_softdisc external function](#) on page 686
[FinishedItem external function](#) on page 686
[Handle_disc external function](#) on page 687
[HookflashLine external function](#) on page 687
[ListCbackFmt external function](#) on page 688
[LogloStart external function](#) on page 689
[LogloStop external function](#) on page 689
[MoFlashBlind external function](#) on page 690
[ReadField external function](#) on page 691
[ReleaseLine external function](#) on page 691
[SendMessage external function](#) on page 692
[SetCallback external function](#) on page 693
[Set_early_hu external function](#) on page 694
[SetNotifyFld external function](#) on page 694
[Set_Softdisc external function](#) on page 695
[UpdateField external function](#) on page 695

DoNotCall external function

The DoNotCall external function identifies a customer record as "do not call" (DNC). All matching records on other calling lists are also marked DNC. The DNC feature ensures that records appearing in multiple calling lists will not be recalled.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see "AGTDoNotCall" in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

End_softdisc external function

The End_softdisc external function ends handling a soft disconnect and re-executes the agt_chl application for the next call. See the descriptions for the Handle_disc and Set_Softdisc external functions.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if this external function does not get executed.

This external function uses the Avaya IR soft disconnect (soft_disc) command. For information about soft disconnect, see the Avaya IR System Help.

FinishedItem external function

The FinishedItem external function releases a customer record and inserts a call completion code into the record. This must be the last activity performed by the script, since the Dialer sends another call after receiving it.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description

Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Completion Code	The two-digit numeric code to place in the customer record that indicates the results of the telephone call. This completion code is customer-specific, but you must be consistent with the completion codes on the Dialer.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTFinishedItem in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

Handle_disc external function

The Handle_disc external function sets the label for handling a soft disconnect. This function marks the beginning of the event-handling code. Between Handle_disc and End_softdisc, place whatever code you think is necessary to clean up the call, for example, updating the completing code on the Dialer system. See the descriptions for the End_softdisc and Set_Softdisc external functions.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function.

This external function uses the Avaya IR soft disconnect (soft_disc) command. For information about soft disconnect, see the Avaya IR System Help.

HookflashLine external function

The HookflashLine external function transfers the current call by way of a switch that accepts hookflash-initiated transfers. This external function disconnects the transferring application before transferring the call.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
smsg	The number to transfer the call to.
Error Code	Select or name the variable in which to store the 6-digit error code from Dialer for this external function.

For more information, see AGTHookflashLine in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

ListCbackFmt external function

The ListCbackFmt external function returns the date format from the caller record. It also lists the number of telephone numbers in the customer record.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Date Format	Select or name the variable in which to store the date format that Dialer will return. The date format returned will be one of the following: • CCYY/MM/DD • MM/DD/CCYY • DD/MM/CCYY

Number of Phones	Select or name the variable in which to store the number of phones listed in the customer record.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTListCallbackFmt in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

LogloStart external function

The LogloStart external function turns on message-logging on the dialer to trace call transactions. Creates a file on the Dialer named /tmp/ AgentName _API.trans. It contains all the messages between the agent application and the Dialer. Use this file to verify message content and to track the agent application session. See the description for the LogloStop external function.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Error Code	Select or name the variable in which to store the 6-digit error code from Dialer for this external function. The error code is: M00000 - Complete

For more information, see AGTLogloStart in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

LogloStop external function

The LogloStop external function turns off message-logging on the dialer. Stops the agent binary from writing to the /tmp/ AgentName _API.trans. Use this external function to terminate the log file. See the description for the LogloStart external function.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Error Code	Select or name the variable in which to store the 6-digit error code from Dialer for this external function. The error code is: M00000 - Complete

For more information, see AGTLogloStop in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

MoFlashBlind external function

The MoFlashBlind external function transfers the current call and data record to another job on the Dialer. This external function disconnects the transferring application before transferring the call and customer record. The Dialer places the customer call in the queue for the new job or connects the customer to an available agent. If the Dialer uses ANI/DNIS information, the transferred call carries the outbound number the Dialer dials as the ANI and the original outbound job name as the DNIS. The inbound or blend job receiving the call sees the call as an inbound call.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Error Code	Select or name the variable in which to store the 6-digit error code from Dialer for this external function.

For more information, see AGTMoFlashBlind in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

ReadField external function

The ReadField external function reads a field from the Dialer customer record.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Field Name	Enter the name of the field in the Dialer calling list from which you want to receive the field format and the field value. Get the field name from the Dialer administrator.
Field Value	Select or name the variable in which to store the field value returned by the Dialer.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTReadField in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

ReleaseLine external function

The ReleaseLine external function releases the telephone line but allows further updating of the customer record.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function.

	The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Script Label	This is an optional attribute. This attribute refers to a label in the /opt/avaya/pds/scripts/telephony.spt file.
Message Number	This is an optional attribute. This attribute refers to a message number configured in the /opt/avaya/pds/config/voicemsg.cfg file.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTReleaseLine in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

SendMessage external function

The SendMessage external function sends a message to the Dialer supervisor's screen.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Message	Enter the message (or select the variable that contains the message) to send to the Dialer supervisor. The maximum message length is 79 characters.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function. The error code is: M00000 - Complete

For more information, see AGTSendMessage in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

SetCallback external function

The SetCallback external function sets the time on the Dialer to schedule a customer callback.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Callback Date	Enter the callback date or select the variable in which you have stored it. The format must match the format returned by the ListCbackFmt external function. The date can be in one of the following formats: • CCYY/MM/DD • MM/DD/CCYY • DD/MM/CCYY
Callback Time	Enter the callback time or select the variable in which you have stored it. The time can be in any of the following formats: • HHMM (24-hour clock) • HHMM[A P] (12-hour clock) • HHMM+ (incremental time from current time)
Phone Index	Index of the Dialer phone field to use for the recall. This information can be obtained from the Dialer administrator and must fall in the range returned by the ListCbackFmt external function.

Optional Fields	Specify a customer name and telephone number (separated by a comma) to contact during the recall.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTSetCallback in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

Set_early_hu external function

The Set_early_hu external function prepares the application for an early hangup. The sim_agt DIP posts a hangup event to the application when the Dialer detects a hangup. The application must define an L_EARLYHANGUP label. The code associated with the L_EARLYHANGUP label should perform the processing that is required when an early hangup event is received.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function.

This external function uses the Avaya IR soft disconnect (soft_disc) command. For information about soft disconnect, see the Avaya IR System Help.

SetNotifyFld external function

The SetNotifyFld external function sets the system to pass the value of a specified key field from the Dialer calling list with the CallNotification message.



Note:

The SetNotifyFld function should be used only in an initial application to be run when an agent joins a job.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description

Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Field Name	Enter the name of the field in the Dialer calling list from which you want to receive the field value as part of each CallNotify message. Get the field name from the Dialer administrator.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTSetNotifyKeyField in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

Set_Softdisc external function

The Set_Softdisc external function prepares the application for a soft disconnect. This function catches the dipterm event and runs the handler. This should be the first external function in the application. See the descriptions for the Handle_disc and End_Softdisc external functions.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function.

This external function uses the Avaya IR soft disconnect (soft_disc) command. For information about soft disconnect, see the Avaya IR System Help.

UpdateField external function

The UpdateField external function updates a data field on the Dialer customer record.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Description
Put Return Code In	Select or name the variable in which to store the return code if Avaya IR does not execute this external function. The return codes are: -1 = a failure of the DIP -2 = a timeout condition
Field Name	Enter the name of the field in the Dialer calling list to update, or select the variable in which you have stored the field name.
Field Value	Enter the new value to insert into the field.
Error Code	Select or name the variable in which to store the 6-character error code from Dialer for this external function.

For more information, see AGTUpdateField in Chapter 6, "Commands and Notification Events," in the Avaya PDS Agent API Reference Guide.

FAX external functions

The following table describes the external functions that can be used in conjunction with the FAX features on the target system. For more information about the FAX external functions, see [Using the FAX external functions](#) on page 471.

Name	Description
FAX_Queue	Queues a fax for transmission.
FAX_CovrPage	Combines two faxes into a single fax, with the second fax starting on the same page as the first (same as BT_FAX_CovrPage).  Note: This external function is currently not supported on Avaya IR.
FAX_Send	Directs the system to transmit the specified graphic image or text file to a designated fax telephone number. Can

	be used to schedule a fax transmission (same as BT_FAX_Send).
Exec_UNIX	Directs the system to execute a UnixWare command or shell script (same as BT_Exec_UNIX).
Concat5	Concatenates (joins together) up to five character strings to make a resulting single string (same as BT_Concat5).
FAX_Get	Directs the system to receive a fax on the current call and to place it into a designated directory and file (same as BT_FAX_Get).
FAX_CNG	Turns FAX CNG tone detection off or on (same as BT_FAX_CNG).  Note: This external function is currently not supported on Avaya IR.
Text2fax	Converts text files to fax files in TIFF format. This external function can convert text files that contain up to 66 lines per page and have either of the following line lengths: • 80 characters per line • 120 characters per line
BT_FAX_Get	Directs the system to receive a fax on the current call and to place it into a designated directory and file (same as FAX_Get).  Note: This external function is currently not supported on Avaya IR.
BT_FAX_CovrPage	Combines two faxes into a single fax, with the second fax starting on the same page as the first (same as FAX_CovrPage).  Note: This external function is currently not supported on Avaya IR.

BT_FAX_Current	Directs the system to transmit on the current call the specified graphical image or text files to a designated fax telephone number.  Note: This external function is currently not supported on Avaya IR.
BT_FAX_Combine	Combines up to three files into a single fax, with each file starting on a separate (new) page.  Note: This external function is currently not supported on Avaya IR.
BT_FAX_CNG	Turns FAX CNG tone detection off or on (same as FAX_CNG).  Note: This external function is currently not supported on Avaya IR.
BT_Concat5	Concatenates, or joins together, up to five character strings to make a resulting single string (same as Concat5).  Note: This external function is currently not supported on Avaya IR.
BT_Exec_UNIX	Directs the system to execute a UnixWare command or shell script (same as Exec_UNIX).  Note: This external function is currently not supported on Avaya IR.
BT_FAX_Send	Directs the system to transmit the specified graphic image or text file to a designated fax telephone number. Can be used to schedule a fax transmission (same as FAX_Send).

	 Note: This external function is currently not supported on Avaya IR.

Form Filler external functions

The following table describes the external functions that can be used in conjunction with Form Filler features on the target system.

Name	Description
FF_Code	Codes a message for the Form Filler application.
FF_Store	Stores the phrases for the Form Filler application.

UCID external functions

The following table describes the external functions that can be used in conjunction with Universal Caller Identification(UCID) features on the target system.

Name	Description
set_uui	Sets User-to-User Information (UUI) for an outbound call. If you enter text for this attribute, do not use double quotes ("). UUI is supported on CONVERSANT V8 and later, and on Avaya Interactive Response.  Note: The function get_uui() must be used before any call to set_uui(), because the two functions access a common space. Common space is used to store any binary remainder of an inbound UUI field, and to reuse it in an outbound call.
get_uui	Extracts the UUI from an incoming call.

get_ucid	Extracts the Universal Call Identification (UCID) from a call.
clear_ucid	Clears the UCID from a call.

GeoTel external functions

The following table describes the external functions that can be used in conjunction with GeoTel features on the target system.

Name	Description
IcrTrans	Obtains an extension from a GeoTel Intelligent Call Router (ICR) in order to transfer a call.  Note: This external function is currently not supported on Avaya IR.

Interaction Center external functions

The following table describes the external functions that can be used in conjunction with Interaction Center features on the target system.

Name	Description
Vesp_dip	Accesses the Voice Enhanced Services Platform (VESP) dip. For information about the functions that can be called from the Vesp_dip, see Vesp_dip on page 700.

Related topics:

[Vesp_dip](#) on page 700

Vesp_dip

This topic describes the functions that can be called from the Vesp_dip external function.

**Note:**

Any function that can accept an EDUID can also accept a string of the form #channelnum, for example, #17. For more information, see the VOX Server Programmer's Guide.

Related topics:

[+create](#) on page 701
[+getarg](#) on page 702
[+send](#) on page 703
[+setarg](#) on page 704
[newcall](#) on page 705
[pseudo_ani](#) on page 706
[getvdu](#) on page 707
[getvox](#) on page 708
[setvdu](#) on page 710
[tr<n><op>](#) on page 711
[transfer](#) on page 712
[gone](#) on page 712
[alarm](#) on page 713

+create

The +create function creates a request.

**Note:**

The +create function does not take effect until it is sent to the VOX Server by a +send function.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	+create	Creates a request.
Arg 1		Select the variable in which to store an allowed VOX Server command name ("VOX.setvdu", "VOX.getvdu", "VOX.getvox", "VOX.tr1XXX", or "VOX.tr2XXX"). The variable must have

		Length set to 34 and Type set to character.
Arg 2		Select the variable in which to store a string containing the total number of arguments. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

+getarg

The +getarg function gets the desired argument.

The string value "0" (or empty) obtains any error string (exception) the VOX Server returns on the request. This will be an empty string if the Vesp_dip request did not return an error. The string is useful for debugging and logging; it is not intended that an Avaya IR application attempt to analyze it. An exception string from any Avaya Interaction Engine Server may be returned (especially the EDU, Telephony, Workflow, and VOX servers).

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	+getarg	Gets the desired argument.
Arg 1		Select the variable in which to store a string representing the argument to be fetched ("0", "1", ..., "62"). The variable must have

		Length set to 34 and Type set to character.
Arg 2		Select the variable in which to store the result. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

+send

The +send function sends the assembled request.

This function does not return until the VOX Server has responded, or the request has timed out. The return code represents the VOX Server's reply.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	+send	Sends the assembled request.
Arg 1		Select a Glossary on page 737 variable. The variable must have Length set to 34 and Type set to character.
Arg 2		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

Arg 3		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

+setarg

The +setarg function sets the value of the specified argument. Any arguments that are not specified are left blank.

**Note:**

The +setarg function does not take effect until it is sent to the VOX Server by a +send function.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	+setarg	Sets the value of the specified argument.
Arg 1		Select the variable in which to store a string indicating the argument you are setting ("1", "2", ..., "62"). The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable in which to store a string containing the value of the argument. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The

		variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

Example

The following example shows how to fetch two values (A and B) from the EDU.

+create "VOX.getvdu" "5"	Create a 5 parameter request.
+setarg "1" "2df5d9e6003b0000780002a1f430002"	Set the value of the EDUID.
+setarg "2" "A"	Set the value of the second argument to A.
+setarg "4" "B"	Skip argument 3, which is where A's result will go, and set the value of the fourth argument to B.
+send	Send the request and wait for a response.
+getarg "3" valueA	Get A's value.
+getarg "5" valueB	Get B's value.

newcall

The newcall function notifies Avaya Computer Telephony when a call arrives, and asks Avaya Computer Telephony to pass back an EDUID for that call. The newcall function must be placed immediately after an Answer node.



Note:

The EDUID identifies the call within Avaya Computer Telephony. It is unique to a particular call, and it is essential for all Avaya Computer Telephony requests.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.

VESP Function Name	newcall	Notifies Avaya Computer Telephony that a new call arrived on the IVR.
Arg 1		Select the variable in which to store the EDUID received from Avaya Computer Telephony, which identifies the new call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

pseudo_ani

The pseudo_ani function should be used only with a network IVR, an IVR not associated with an Avaya Computer Telephony Server, to transfer a call to an Avaya Computer Telephony-enabled site.

This function takes a pseudo-ANI from the list provided by the configuration parameter pseudo_ani, associates it with the given EDUID, and returns the pseudo-ANI in the response.

For information about the pseudo-ANI and the VOX Server's interaction with a network IVR, see the VOX Server Programmer's Guide.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.

VESP Function Name	pseudo_ani	Select the variable in which to store the returned pseudo-ANI.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the pseudo-ANI associated with the call. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

getvdu

The getvdu function retrieves a single value from the call's EDU. For example, a customer's account information can be looked up in the database and stored in an EDU on the EDU Server. The EDU Server passes the requested information to the VOX Server, which in turn passes the value to the VRU. This function also updates the name/value pair stored in the VOX Server's value cache. For information about getvdu and the value cache, see the VOX Server Programmer's Guide.



Note:

Always issue the getvdu function instead of getvox if there is any risk of other software running in parallel with your CONVERSANT or Avaya IR application; otherwise, pertinent EDU values may differ.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description

Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	getvdu	Gets a value from the EDU.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the name of the database field to query. The variable must have Length set to 256 and Type set to character.
Arg 3		Select the variable in which to store the returned value of the database field; empty if the field does not exist. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.

getvox

The getvox function retrieves a single value from the EDU's name/value pair stored in the VOX Server's value cache.

Because getvox retrieves the information from the VOX Server's value cache rather than from the EDU Server, it results in faster response time and less network traffic than using getvdu.

Use getvox only when you are certain that the cache holds the field's most recent value. This will always be the case if only the CONVERSANT or Avaya IR application can cause the EDU's value to change. For example, if a field can be set only by the tr2setacctnt function and not by

any other Avaya Computer Telephony software, both the VOX Server cache and the EDU will contain the same field values.

getvox is best suited for accessing stable information such as account numbers and ANI (Automatic Number Identification). For information about getvox and the value cache, see the VOX Server Programmer's Guide.



Note:

Always issue the getvdu function instead of getvox if there is any risk of other software running in parallel with your CONVERSANT or Avaya IR application; otherwise, pertinent EDU values may differ.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	getvox	Gets a value from the VOX Server value cache.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the name of the database field to query. The variable must have Length set to 256 and Type set to character.
Arg 3		Select the variable in which to store the returned value of the database field; empty if the field does not exist. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Glossary on page 737 variable. The

		variable must have Length set to 256 and Type set to character.

setvdu

The setvdu function sets or overwrites a single value in the customer's account information. The VRU passes the name/value pair information to the VOX Server, which in turn passes it to the EDU Server.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	setvdu	Sets a value in the EDU.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the name of the database field to overwrite. The variable must have Length set to 256 and Type set to character.
Arg 3		Select the variable that contains the new value of the field. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.

tr<n><op>

Transaction operation commands communicate with the Workflow Designer. Workflow Designer features are customized for each customer, either by Avaya or by the customer.

**Note:**

Transaction operation commands are custom designed for your site. If desired, you could execute tr<n><op> requests to change information in Avaya Interaction Center databases, fax information, perform complex calculations, pre-fetch images, or whatever else is needed.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	tr<n><op>	Performs a custom transaction.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the value of the database field. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

transfer

The transfer function transfers a call from the VRU to an agent at the specified extension. Transferring the call does not terminate the EDU. The CONVERSANT or Avaya IR application continues to access the EDU and cache until the gone function is called.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	transfer	Transfers the call to an agent.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the PBX extension to which the call will be transferred. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

gone

The gone function notifies the Avaya Computer Telephony software that the call has ended, and releases the EDUID. The gone function must always be issued when a call ends.

**Note:**

Be sure to issue the gone function before the application ends; if you do not, subsequent calls on the same channel could be lost or otherwise improperly handled.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	gone	Notifies Avaya Computer Telephony software that the call has ended.
Arg 1		Select the variable in which to store the EDUID, which is passed to Avaya Computer Telephony to identify the call. The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the reason for ending the call, or leave this argument blank. The variable must have Length set to 256 and Type set to character.
Arg 3		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Dummy variable. The variable must have Length set to 256 and Type set to character.

alarm

The alarm function raises an Avaya Computer Telephony alarm through the VOX Server to indicate some exceptional condition or error. This function is useful for debugging.

Use the Node Inspector to assign and edit the following attributes.

Attribute	Value	Description
Put Return Code In		Select the variable in which to store the return code.
VESP Function Name	alarm	Raises an Avaya Computer Telephony alarm.
Arg 1		Select the variable that contains the alarm's name (one word). The variable must have Length set to 34 and Type set to character.
Arg 2		Select the variable that contains the alarm's priority code: Low, High, Info, or EMERGENCY. The variable must have Length set to 256 and Type set to character.
Arg 3		Select the variable that contains the text that describes the problem. The variable must have Length set to 256 and Type set to character.
Arg 4		Select a Glossary on page 737 variable. The variable must have Length set to 256 and Type set to character.

NICE external functions

The following table describes the NICE external functions that can be used in IVR Designer Applications.

Name	Description

StartRecording	Sends messages to request that a recording be started right now.
StopRecording	Sends a request that a particular recording be stopped right now. If no StopRecording is received, the recording will continue for a number of minutes specified as a parameter in the StartRecording message.
EMStart	Indicates that this is the start of a session where information will be sent to EM. This message contains an application VDN and a port so that EM can begin monitoring the port and collecting the UCID to associate to incoming messages.
EMStop	Indicates that the end of a session has been reached.
EMPAC	Sends a prompt name and collected value for each Prompt and Collect step within the IVR Designer application.

Related topics:

[TAS to CPS \(tas2cps\)](#) on page 715

[TAS to Event Manager \(tas2em\)](#) on page 720

TAS to CPS (tas2cps)

The purpose of the tas2cps package is to provide a link between IVR Designer and AgentAssist. tas2cps enables you to start and stop recordings within IVR Designer applications and to manage recordings with an AgentAssist client such as Administrator.

The tas2cps package works together with two IVR Designer external functions: StartRecording and StopRecording. Both functions send messages to a Data Interface Process (DIP), tas2cpsdip, which part of tas2cps and runs on the CONVERSANT or Avaya IR.

Upon receiving a message to start or stop a recording, the tas2cpsdip process updates the Oracle database and sends a request to the existing AgentAssist plrecord process. The AgentAssist plrecord process then starts or stops a recording in much the same way that it would start or stop an On-Demand recording initiated through an AgentAssist client such as Agent Now.

After a recording has been made, it can be accessed and reviewed using an AgentAssist client or a VUI Playback port. This means that a recording made from within a IVR Designer application is now available in the AgentAssist Administrator and can be played, deleted, or saved on the client PC. These recordings will have a different type, indicating that they were created by IVR Designer. In addition, the recordings have a lower priority than any of the

existing AgentAssist clients (for example, AgentNow! or C.E.O.--Customer Experience Observing).

Because the recordings are made using AgentAssist, please note the following:

- Recordings can be made only for agents entered into Administrator. When the tas2cpsdip process inserts a new recording into the database, the agent or VDN to be recorded must be in the pl_agentlist. That is, the agent or VDN must have been entered as an agent using Agent Administrator.
- An additional port is used while the recording is being made. The recording process does not take place on the port running the IVR Designer application.
- The recordings made this way have the lowest priority. Consequently, if all ports designated as "Usable for recording" are in use by other AgentAssist applications, a recording will not be made and an error will be returned. Ports are set to "Usable for recording" status through the Port Manager application.
- The tas2cpsdip is installed as part of tas2cps which is a separate package that is installed on the CONVERSANT V8, CONVERSANT R9, or Avaya IR.

There is no licensing specifically for this product. Licensed users of AgentAssist and IVR Designer will automatically have access to tas2cps functionality as long as tas2cps is installed.

Related topics:

[tas2cps components](#) on page 716

[Sending a tas2cps request](#) on page 717

[Error handling in tas2cpsdip](#) on page 720

tas2cps components

The following components have been added to enable tas2cps capabilities.

tas2cpsdip

tas2cpsdip is the new DIP installed with tas2pcs which establishes two queues:

- Listen for messages from IVR Designer applications
- To communicate with plrecord.

The tas2cpsdip process is available on CONVERSANT V8, CONVERSANT R9, and Avaya IR platforms.

When a message is received from the IVR Designer application, tas2cps validates the received values, inserts or updates a record in the database, and sends a request to plrecord to start or stop recording. IVR Designer messages are sent with a type of 5 making them the lowest priority recordings being made. As a result, if a request comes in from any AgentAssist module and every port with Usable for Recording status is in use, any type 5 recordings in progress

will be stopped and assigned an error status. The port will then be assumed by the higher priority AgentAssist application.

plrecord

This process has been updated to receive messages from a second queue (the IVR Designer queue) and to handle a new priority level. This is installed as part of package agtasstc (AgentAssist) V2.9.3.

Agent Assist clients

The Agent Assist Administrator, Agent Observing, Customer Experience Observing, and Retriever were updated to show "IVR Designer" under the type column on the display for any recordings initiated by IVR Designer applications via the StartRecording functionality.

Sending a tas2cps request

The StartRecording and StopRecording external functions can be used to send a message to the tas2cpsdip running on the CONVERSANT or Avaya IR . The parameters used are described in StartRecording and StopRecording.

Related topics:

- [StartRecording](#) on page 717
- [StartRecording data validation](#) on page 718
- [StartRecording error codes](#) on page 719
- [StopRecording](#) on page 719
- [StopRecording error codes](#) on page 720

StartRecording

- Length – This is the maximum number of minutes for a recording unless it is stopped sooner by a stop message. For example, if a value of 5 specifies that a recording will be made for up to 5 minutes unless it is stopped by a Stop Recording message.
 - Required.
 - Type: character string.
 - Length: up to 3 characters long.
 - Valid range: 1-120.
- Login – Not currently in use. The user should send a value of VAW for now.
 - Optional.
 - Type: character string.
 - Length: up to 5 characters long.
 - Valid range: should be the string VAW
- Extension – This is the extension to be recorded. The extension must be entered because the StartRecording external function cannot determine this information on its own. Also, the extension to be recorded must be entered into AgentAssist because recordings cannot be made for extensions that are not in the pl_agentlist table.

- Required.
- Type: character string.
- Length: up to 16 characters long.
- Valid range: must be entered as an agent in AgentAssist (that is, must be in the pl_agentlist table)
- Type – This should be set to 5 to reflect a type of IVR Designer On Demand recording. Because this is treated as a priority value within the plrecord process, with a value of 5, these are the lowest priority recordings
 - Required.
 - Type: character string.
 - Length: up to 1 character long.
 - Valid value: 5
- Notes – This is a text field – the DIP will store the characters sent in the database and the notes entered here will be viewed the same way notes entered for S.T.A.R. recordings are viewed.
 - Optional.
 - Type: character string.
 - Length: up to 100 characters long.
 - Valid range: none
- Return code – This will be a negative one (-1) if an error occurred (i.e., no ports in recording status, DIP is not running, etc...) or it will be the Recording ID if the recording was started successfully. The recording ID can then be passed to the StopRecording external function or, if no StopRecording message is sent, the recording will stop after Length number of minutes.
 - Required.
 - Type: number
 - Length: up to 10 digits.
 - Valid range: +, -

StartRecording data validation

The data validation for the StartRecording message occurs in two places--the first is in the StartRecording.t file before the call to tas2cpsdip is made and the second is within the DIP itself. The following table summarizes which checks occur where.

Check	Made In	
	StartRecording.t	tas2cpsdip
Length of recording length parameter is <=3	3	

Value of recording length parameter is between 1 and 120	3
Length of login parameter is <=5	3
Length of extension parameter is <= 16	3
Value of extension parameter is not NULL	3
Length of type parameter is <= 1	3
Value of type parameter is = 5	3
Length of notes parameters is <=100	3

StartRecording error codes

An error value of -1 is returned from the DIP if any of the following problems occurs:

- Validation of input parameters fails
- The Service Observing Feature Access Code cannot be read
- The next recording ID cannot be generated from the Oracle sequence
- The recording request cannot be inserted into the pl_recordings table
- The request to the plrecord process cannot be successfully completed

StopRecording

- RecID – This is the recording ID passed back as the return code of StartRecording when a recording has been successfully started.
 - Required.
 - Type: number
 - Length: up to 10 digits long.
 - Valid range: +



Note:

: There is no specific validation done on the length or value range of the recording id passed in.

- Return code – This will be negative if an error occurred or it will be 0 if everything went without error.

- Required.
- Type: number
- Length: up to 10 digits long.
- Valid range: +, -

StopRecording error codes

An error value of -1 will be returned from the DIP if any of the following problems occurs:

- The pl_recordings table does not contain a record with the recording id specified in a P or A status.
- The pl_recordings table cannot be updated - to put a record with this recording id into a status of S while a call to plrecord takes place.
- The request to the plrecord process cannot be successful completed.

Error handling in tas2cpsdip

The tas2cpsdip process will exit if any of the following conditions occur:

- The process cannot register with IRAPI to establish a communication queue to plrecord.
- The process cannot establish a queue from which to read TAS messages.
- The process cannot create a message object for AgentAssist-related messages.
- The process has an error reading from the TAS queue.
- The process has an error sending a response back to the TAS script.

Once the CONVERSANT or Avaya IR is re-booted after installation of the tas2cps package, the call to the tas2cpsdip process should be included in the inittab file and if the application does not respond for any reason, it should be restarted automatically.

If the CONVERSANT or Avaya IR has not been rebooted after tas2cps has been installed and the application stops responding, tas2cps will need to be restarted manually by typing:

```
/usr/add-on/centerpoint/dip/tas2cpstart
```

To stop the tas2cpsdip process type:

```
/usr/add-on/centerpoint/dip/tas2cpsstop
```

If the CONVERSANT or Avaya IR has been rebooted, tas2cps will restart automatically. It is not necessary to type:

```
/usr/add-on/centerpoint/dip/tas2cpstart
```

TAS to Event Manager (tas2em)

The purpose of the tas2em package is to provide a link between applications built by IVR Designer and Event Manager (EM). Using this functionality, information can be passed from

IVR Designer scripts to EM and from there to applications like CCA. In the current implementation, the information that can be collected and passed on to EM is limited to prompt and collects where the collected value is keypad input (not voice).

When the tas2emdip starts up, it sends a message to the EM, containing the IP address and host name of the machine on which it is running and a list of the ports and their extensions. EM can then use this information to set up observers on these extensions. The tas2emdip then begins listening for messages from IVR Designer scripts. The following external functions can be used in a TAS application in IVR Designer.

- EMStart – indicates that this is the start of a session where information will be sent to EM. This message contains an application VDN and a port so that EM can begin monitoring the port and collecting the UCID to associate to incoming messages.
- EMPAC – which sends a prompt name and collected value for each Prompt and Collect step within the IVR Designer application.
- EMStop – which indicates that the end of a session has been reached.

These external functions must be used properly in a IVR Designer application to ensure that information flows successfully to the Event Manager. For example, if a EMStart is not included before an EMPAC, the EMPAC sent to tas2emdip will not be forwarded to the EM. Likewise, a EMPAC must be included in the IVR Designer application after each Prompt and Collect node or the information will not be forwarded to tas2emdip and from there to EM.

Upon receipt of a EMStart, the tas2emdip will create a new socket connection for the port on which the IVR Designer application is running (this is a field in the EMStart message). This socket connection will remain open until a EMStop or another EMStart is received. All subsequent EMPAC messages received will be forwarded to EM. When a EMStop message is received, it will be forwarded to EM and the socket connection will be closed. If another EMStart is received before an EMStop, the tas2emdip will generate it's own EMStop with a disposition indicating that this EMStop was generated by the dip and not received from the IVR Designer application. The socket connection will then be closed, a new socket connection will be established, and the EMStart will be sent.

While the tas2emdip process is reading the IVR Designer application messages from a queue, it will also be sending a Heartbeat message to the Event Manager each minute. If the Heartbeat message cannot be sent successfully, the tas2emdip will note that the EM process must be down and upon the next successful sending of a Heartbeat, it will resend the initial message containing the list of ports and extensions. The tas2emdip will still try to send all messages it receives during this time.

Some things to note about the current implementation are:

- Currently, we can only collect touch tone input, not voice input.
- All messages sent from tas2emdip to EM are sent as XML over a socket.
- Because timing is critical, the messages must be forwarded from IVR Designer to EM as soon as possible since EM must associate a UCID to the message. There is minimal error handling. A success status is returned to the IVR Designer message upon receipt of the message within the tas2emdip, not upon the successful forwarding to EM.

Related topics:

[tas2em components](#) on page 722

[Sending a tas2em request](#) on page 722

tas2em components

The following components have been added to enable tas2em capabilities:

tas2emdip

This is the new DIP which establishes a queue on which to capture messages from IVR Designer scripts and a socket connection to the Event Manager. It receives messages from IVR Designer applications and reformats them as XML and sends them to EM via a socket connection.

Event Manager

Event Manager has been updated to receive messages from the tas2emdip and forward them to CCA.

Sending a tas2em request

The EMStart, EMPAC, and EMStop external functions can be used to send a message to the tas2emdip, running on the CONVERSANT or Avaya IR. The parameters used are described in EMStart, EMPAC, and EMStop.

Related topics:

[EMStart](#) on page 722

[EMPAC](#) on page 723

[EMStop](#) on page 723

EMStart

- Application – This is the name of the application VDN.
 - Required.
 - Type: character string.
 - Length: up to 19 characters long.
- Port – This the port on which the application is running.
 - Required.
 - Type: integer.
 - Valid range: should be a valid port on the CONVERSANT or Avaya IR
- Return code – This will be negative if an error occurred or positive if the message was read by tas2emdip. A positive value is returned immediately upon the reading of the message so this status only reflects that tas2emdip read the message, not that it has been successfully forwarded to EM.
 - Required.
 - Type: number

- Length: up to 10 digits.
- Valid range: +, -

EMPAC

- Application – This is the name of the application VDN.
 - Required.
 - Type: character string.
 - Length: up to 19 characters long.
- Port – This the port on which the application is running.
 - Required.
 - Type: integer.
 - Valid range: should be a valid port on the CONVERSANT or Avaya IR
- Prompt
 - Required.
 - Type: character string.
 - Length: 63.
- Collect
 - Required.
 - Type: character staring.
 - Length: 127.
- Return code – This will be negative if an error occurred or positive if the message was read by tas2emdip. A positive value is returned immediately upon the reading of the message so this status only reflects that tas2emdip read the message, not that it has been successfully forwarded to EM.
 - Required.
 - Type: number
 - Length: up to 10 digits.
 - Valid range: +, -

EMStop

- Application – This is the name of the application VDN.
 - Required.
 - Type: character string.
 - Length: up to 19 characters long.
- Port – This the port on which the application is running.

- Required.
- Type: integer.
- Valid range: should be a valid port on the CONVERSANT or Avaya IR
- Disposition – This is a status which provides additional information as to why the application is stopping.
 - Required.
 - Type: integer.
 - Valid range: +,-
- Return code – This will be negative if an error occurred or positive if the message was read by tas2emdip. A positive value is returned immediately upon the reading of the message so this status only reflects that tas2emdip read the message, not that it has been successfully forwarded to EM.
 - Required.
 - Type: number
 - Length: up to 10 digits.
 - Valid range: +, -

Phrase external functions

The following table describes the external functions that can be used in conjunction with Phrase files on the target system.

Name	Description
Background	Turns off or on a pre-recorded phrase that is identified by the phrase number or NX number. The prerecorded phrase plays in the background of the call.  Note: This external function is currently not supported on Avaya IR.
Background_Tag	Turns off or on a pre-recorded phrase that is identified by the phrase tag. The prerecorded phrase plays in the background of the call.  Note: This external function is currently not supported on Avaya IR.

Msg_Code	Records a phrase, using by default the phrase number or NX number to identify the phrase.
Msg_Code_Tag	Records a phrase, using by default the phrase tag to identify the phrase.
Msg_Delete	Deletes a phrase, using by default the phrase number or NX number to identify the phrase.
Msg_Delete_Tag	Deletes a phrase, using by default the phrase tag to identify the phrase.
Pack_phrNX	Converts a talkfile number and a phrase number into a combined number to be spoken in NX format.
Unpack_phrNX	Reverses the Pack_phrNX function, separating the combined number into a talkfile number and a phrase number.

PRI external functions

The following table describes the external functions that can be used in conjunction with Primary Rate Interface (PRI) features on the target system.

Name	Description
Attr_ANI	Requests a billing number for incoming calls on a call-by-call basis.  Note: This external function is currently not supported on Avaya IR.
Attr_busy	Sets the call progress tone to "busy." Callers hear a busy tone.
Attr_ring	Sets the call progress tone to "ringing." Callers hear a ringing tone.
Attr_silence	Sets the call progress tone to "silence." Callers hear silence.

ISDN_billing	Retrieves the billing number (also known as ANI or CPN) for the incoming PRI call.  Note: This external function is currently not supported on Avaya IR.
ISDN_CINFO	Retrieves Caller Information Forwarding (CINFO) data. Select the variable in which to store the CINFO data. The variable type must be character. The return codes are: 0 = success greater than 0 = failure
ISDN_service	Sets the service type for outgoing PRI calls.
Tbct_getinfo	Gets up to 30 characters of call information (ANI, DNIS, UUI) from an incoming call that may be used as input to the Tbct_Xfer external function.
Tbct_Xfer	Transfers a Two B-Channel call (incoming and outgoing calls) to the network switch. If you enter text for this attribute, do not use double quotes (").

NLSR external functions

The following table describes the Natural Language Speech Recognition external functions that can be used in IVR Designer applications.

Note: For more information about the NLSR external functions and how to use them, see the Avaya publication Natural Language Speech Recognition with the CONVERSANT® System, 585-310-774, that comes with the Avaya Natural Language Speech Recognition offer.

Name	Description
AllocPROXY	Reserves an NLSR resource of a specified type. In other words, AllocPROXY sets up a connection to an NLSR server. For information about using AllocPROXY, see A caution about

	the AllocPROXY external function on page 729.
DeallocPROXY	Releases an NLSR resource of a requested type.
GetdcPROXY	Requests the speech data collection file name for the most recent recognition attempt. Notes: This external function is currently not supported on Avaya IR Release 1.0 through Release 1.2. It is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases. This external function should be used only to debug an application because it can adversely affect performance. To enable the speech proxy to support this function, you must: edit the /vs/proxy/OPSR#.cfg file to include the line DataCollection=yes.
LreqPROXY	The LreqPROXY external function is similar to the RequestPROXY external function, except that LreqPROXY allows for a larger response string from the speech recognizer. The LreqPROXY external function should be used in IVR Designer applications only. Do not use this external function in Script Builder applications.  Note: Do not use the LreqPROXY external function and the RequestPROXY external function in the same application. The application does not compile correctly and generates the "multiple define" error.
MrcpSetParam	Provides a comprehensive interface to set the following MRCP parameters on the speech server at the start of each recognition request: • confidencelevel • sensitivity • speedvsaccuracy

	• completetimeout • incompletetimeout • maxspeechtimeout • maxnbest • savewaveform • recognitionmode The timeout values for completetimeout, incompletetimeout, and maxspeechtimeout are measured in milliseconds. The valid range for confidencelevel, sensitivity, and speedvsaccuracy is from 0 to 1.0. If you set a parameter value to -1, then the value is not set on the speech server. In such a case, IVR Designer takes the default value available on the speech server.  Note: Do not use the MrcpSetParam external function and the RequestPROXY external function in the same application. The application does not compile correctly and generates the multiple define error message.
RequestPROXY	Sends a speech recognizer-specific command string to the speech recognizer, and receives a response string from the speech recognizer. The RequestPROXY external function must be called after an AllocPROXY external function. The RequestPROXY external function may be called multiple times.  Note: Do not use the LreqPROXY external function and the RequestPROXY external function in the same application. The application does not compile correctly and generates the error "multiple define" error.
RmdcPROXY	Requests that a specified speech data collection file be removed.

	Notes: This external function is currently not supported on Avaya IR Release 1.0 through Release 1.2. It is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases. This external function should be used only to debug an application because it can adversely affect performance. To enable the speech proxy to support this function, you must: edit the /vs/proxy/OPSR#.cfg file to include the line DataCollection=yes

Related topics:

[A caution about the AllocPROXY external function](#) on page 729

A caution about the AllocPROXY external function

Exercise caution when using the AllocPROXY external function in conjunction with an Announce node. If the AllocPROXY external function is placed immediately after the Announce node in the call flow, and the Allow Interrupt option is set to allow the caller to interrupt the Announce prompt, AllocPROXY will fail and you will get a return value of -2.

To avoid this problem, do one of the following:

- If possible, set the Allow Interrupt option to not allow the caller to interrupt the Announce prompt.
- Add a second, "null", Announce node immediately after the first (before the AllocPROXY external function). Leave the prompt blank and set the Allow Interrupt option to not allow the caller to interrupt.
- Place the AllocPROXY external function so that it comes before the Announce node in the call flow.

Speech Features external functions

The following table describes the external functions that can be used in conjunction with Speech Recognition features on the target system.

Name	Description
DPR_Disable	Enables or disables the Dial Pulse Recognizer.

	 Note: This external function is currently not supported on Avaya IR Release 1.0 through Release 1.2. DPR_Disable is supported on Avaya IR Release 1.2.1 and later, and on CONVERSANT R9 and earlier releases.
DPR_Reset	Resets (initializes) Dial Pulse Recognizer training parameters. Training is the ability of the system to learn the characteristics of each caller's telephone.  Note: This external function is supported on Avaya IR Release 1.2.1 and later releases.
Proxylang	Used to select a particular language within an application. This external function should only be used for applications that use multiple languages and the language needs to change within the application.  Note: The Charset argument for this external function is supported only on Avaya IR, and should be used only for the Japanese language.
Proxyparams	Defines the parameters for the Proxy Text-to-Speech (PTTS) speaking voice, including gender, rate of speech, volume, pitch, and intonation. The available settings are: • gender (0=female, 1=male) default=1 • speed (1=slowest, 32000=fastest) default=5 • volume (1=softest, 32000=loudest) default=1

	• pitch (1=lowest, 32000=highest) default=5 • intonation (1=lowest, 32000=highest) default=4  Note: Different TTS engines support different ranges, and some TTS engines might not support all of the parameters. Refer to the documentation for your specific TTS engine to determine the acceptable values.
Proxytext	Converts either a file or a buffer of text to speech.
Proxytts_on	Used to toggle between PTTS and Text-to-Speech (TTS).  Note: This external function is currently not supported on Avaya IR.
Proxyvoice	Used to select a particular voice tag within PTTS. This external function should only be used when an application requires two different voice tags that have the same language and gender.  Note: The Charset argument for this external function is supported only on Avaya IR, and should be used only for the Japanese language.
SP_Allocate	Allocates speech recognition resources.  Note: This external function is currently not supported on Avaya IR.
SR_Prompt	Turn on or off "barge-in" or "speech recognition during prompt."
SWIttsSetup	A SpeechWorks PTTS setup function that is required at the beginning of any

	TAS script that uses SpeechWorks Speechify 2.0 TTS.  Note: This external function is currently not supported on Avaya IR.
Tts_file	Speaks the text from a specified ASCII text file using Text-to-Speech (TTS).  Note: This external function is currently not supported on Avaya IR.

Telephony external functions

The following table describes the external functions that can be used in conjunction with Telephony features on the target system.

Name	Description
Call_Bridge	Places an outbound call; if answered, bridges new and current channels.
Ixfer	Allows a application to place an outbound call to a user-defined telephone number; maintain the connection while the caller interacts with the person on the other channel; and when the called person hangs up, continue with the next action step.
Transfera	Flashes (waits for call progress tones), dials the number, and flashes again.  Note: This external function is currently not supported on Avaya IR.
Transferb	Flashes (waits for call progress tones), dials the number, waits for call progress tones, and flashes again.
Complete	Performs a non-blind transfer call complete routine; connects the caller to a third party; should only be used to

	connect the caller to the attendant (third party) after an answer is detected or during ringing in a non-blind transfer action.
Reconnect	Reconnects the caller to the script; should only be used to reconnect to the caller after a No Answer, Busy, Reorder (Fast Busy), or Error condition was encountered in a non-blind transfer action.  Note: This external function is currently not supported on Avaya IR.
Conv_data	Used with the Converse Vector feature on the DEFINITY to pass data. When a call is sent to the CONVERSANT or Avaya IR, the switch can send data across; when the call ends the CONVERSANT or Avaya IR can send data back to the switch.
Make_Call	Places or dials a call. For more information about the Make_Call external function, see Using the Make_Call external function on page 457.

Vonetix external functions

The following table describes the Vonetix external functions that can be used in IVR Designer Applications.

Name	Description
VntxDBQuery	The VntxDBQuery function runs the given SQL command/query (SQLCMD) and returns the results in the other arguments. Results are treated as strings, so conversion is left to the application (including integers, dates or other types).

VntxDBXCols	This function lets you retrieve more than four column results from Script Builder and more than 14 column results from a IVR Designer application. It is designed to be called after the VntxDBQuery() or VntxDBNxtRow() functions when there are more columns of data than can be handled by an External Function call.
VntxDBNxtRow	This function returns contents from a specific row of the most recent VntxDBQuery() result. Parameter usage and syntax are the same as for VntxDBQuery(), with the exception of the first argument.
VntxDBExec	This function will run the given SQL command (an insert, update or delete) and return the number of rows affected. It may also be used to execute stored procedures that return an integer value.
VntxScanDoc	This function retrieves the document from the given URL (if URL is not blank). It then scans the document for the string patterns START and END. If found, it returns the text between the first occurrence of these tags in the string variable BUFFER (up to LEN bytes).
VntxsetFD	Sets the value of a form data variable.
VntxLoadDoc	Loads the given file as an document.
VntxNewDoc	Creates a new document with the document variable.
VntxClearDoc	Clears the form data associated with the given session name.
VntxClearFD	Clears the document and form data associated with the given session.
VntxClearSes	Clears the document associated with the given session name.
VntxDelTag	Deletes the value of a tag in the XML document with the given session name.

VntxSetTag	Sets the value for a tag in the XML document associated with the given session name.
VntxSetAttr	Sets the attribute value for an instance of a tag in the XML document associate with a given session name.
VntxAddSibl	Creates a new sibling tag in the XML document associated with the given session name.
VntxAddChild	Creates a new child tag in the XML document associated with the given session name.
VntxAddAttr	Adds an attribute to an existing XML tag in the XML document associated with the given session name.
VntxSetDocFD	Sets the document associated with the given session name into an encoded form-data value associated with 'name'.
VntxGetDoc	Will get a document by URL and associate it with the given session name. Any form data associated with the given session name will be sent with the GET request.
VntxPostDoc	Will get a document by URL and associate it with the given session name. Any form data associated with the given session name will be sent with the POST request.
VntxStoreDoc	Will store the document associated with the given session name to a file on the local file system. Useful for troubleshooting purposes.
VntxGetNumAl	Returns the number of all tags in the XML document associated with the given session name.
VntxGetNumTg	Returns the number of a specific tag in the XML document associated with the given session name.

VntxGetTag	Retrieves the value of a tag in the XML document associated with the given session name.
VntxGetAttr	Retrieves the value of a tag attribute in the XML document associated with the given session name.
VntxGetNumTk	Searches for the number of occurrences of a particular token in the document associated with the given session name.
VntxGetToken	Retrieves a particular token from the document associated with the given session name.
EncodeURL	This TAS function takes a SOURCE string and encodes it as a URL.
DecodeURL	This TAS function takes an encoded URL SOURCE and decodes to TARGET it.
VntxParse1	This TAS function will parse the source field for the subfield that is between the separator1 character and the separator2 character.
VntxInit	This function clears all session data, forms and other resources associated with this channel.
VntxFeature	This function executes a plug-in feature.
VntxSpQuery	Run the given SQLCMD (query) or stored procedure and return the results in the other arguments.

Chapter 13: Glossary

3270 interface

The 3270 interface is the link between one or more CONVERSANT® or Avaya IR machines and an IBM host mainframe.

active call flow

The call flow currently showing in the main window (design pad) and available for editing. See also design pad.

Adaptive Differential Pulse Code Modulation

Adaptive Differential Pulse Code Modulation (ADPCM) is a means of encoding analog voice signals into digital signals by adaptively predicting future encoded voice signals. This adaptive modulation method reduces the number of bits required to encode voice recordings.

Adjunct/Switch Application Interface

Adjunct/Switch Application Interface (ASAI) is an optional feature package that provides an Integrated Services Digital Network-based interface between Avaya PBXs and adjunct processors.

ADPCM

See adaptive differential pulse code modulation.

American Standard Code for Information Interchange

American Standard Code for Information Interchange (ASCII) is a standard for assigning numerical values to the set of letters in the Roman alphabet and typographic characters.

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 source files

The files that are used by IVR Designer to generate TAS code during the code generation process. While they are not used by the target system, these files can be transferred to the

target voice response system to assist technical support personnel in troubleshooting applications.

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.

ASAI

See Adjunct/Switch Application Interface.

ASCII

See American Standard Code for Information Interchange.

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 data events

Variables in the application that you ask your system to store in a database for later retrieval. Any data can be designated as call data events, though typically they are counters of some kind. When a call is made to the CONVERSANT or Avaya IR system, the application stores the values of call data events at the end of each call for use in generating Call Data Reports and Call Data Summary Reports.

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.

channel

1. A path of communication between two points. 2. An abbreviation for communication channel. See also communication channel.

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 discontinuous. 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 grammar

See custom vocabulary.

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.

custom vocabulary

A specialized package of unique words or phrases created on a per-customer basis and used by WholeWord speech recognition.

data interface process (DIP)

A software process that communicates with interactive voice response (IVR) applications.

database

A database is a collection of data arranged for ease and speed of search and retrieval. A structured set of files, records, or tables.

database field

A field used to extract values from a local database and form the structure upon which a database is built.

database record

The information in a database for a person, product, event, etc. The database record is made up of individual fields for each information item.

database table

A structure, made up of columns and rows, that holds information in a database. Database tables provide a means of storing information that changes too often to "hard-code," or store permanently, in the transaction outline.

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 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.

dummy variable

A Dummy variable is a custom variable with Length set to 256 and Type set to character. Use this variable only for external function attributes that require a Dummy variable. Do not use the Dummy variable in any other part of the application. You can name the Dummy variable whatever you want. The name Dummy is used here as an example only.

EBS

Enhanced basic speech.

EDU

See Electronic Data Unit.

Electronic Data Unit

An in-memory container for storing a set of data about a contact that enables IC to identify, handle, and report on that contact.

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.

external functions

Specific predefined (or customer-created) system tasks that IVR Designer can call or invoke to interact with other products or services. The function lets the application developer enter arguments for the function to act on. Examples include concat, getarg, length, substring, etc.

field

See database field.

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.

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.

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.

host

See host computer.

host computer

A computer containing data or programs that another computer can access by means of a network or modem.

host flow

A specialized call flow, the primary purpose of which is to set up the interactions between the target voice response system and the host. There is one host flow for each host definition in the application.

icons

A picture on a screen that represents a specific file, directory, window, option, or program.

import

To incorporate a global resource or external file and make it part of or available to the current application.

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.

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.

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.

local database

A database residing on the target voice response system.

logging on/off

Entering or exiting the target voice response system software.

logical unit

Logical unit (LU) is a type of Systems Network Architecture (SNA) Network Addressable Unit.

login

The process of gaining access, or signing in, to a computer system or application. The login process typically involves the authentication of the user's login ID and password.

login ID

A name or identifier that, when combined with a password, uniquely identifies a user to a system or application.

LU

See logical unit.

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.

NLSR

See Natural Language Speech Recognition.

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.

ORACLE

A company that produces relational database management software. It is also used as a generic term that identifies a database residing on a local or remote system which is created and maintained using an Oracle product.

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.

phoneme

A single basic sound of a particular spoken language. For example, the English language contains 40 phonemes that represent all basic sounds used with the language. The English word "one" can be represented with three phonemes, "w" - "uh" - "n." Phonemes vary between languages because of guttural and nasal inflections and syllable constructs.

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.

remote database

Information stored on a system other than the target voice response system that can be accessed by the target system.

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.

SCI tools

See Service Creation Installation tools.

script

The set of instructions for the target voice response system to follow during a transaction.

Service Creation Installation tools

Service Creation Installation (SCI) tools are software tools designed to help you install your IVR Designer applications and related files on the target voice response system.

shared resource

A resource, such as a database table or speech pool, that is used by more than one application or developer. See also global resource.

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.

SQL

See structured query language.

standard external functions

A set of external functions provided by Avaya as part of the IVR Designer package. Sometimes called "system functions." See also external functions.

standard speech

The speech package available in several languages containing simple words and phrases, produced by Avaya for use with CONVERSANT or Avaya IR systems. This package includes digits, numbers, days of the week, and months, each spoken with initial, medial, and falling inflection. The speech is in digitized files stored on the hard disk, to be used in voice prompts and messages to the caller. This feature is also called Enhanced Basic Speech (EBS).

standard vocabulary

A standard package of simple word speech models provided by Avaya and used for WholeWord speech recognition. These phrases include the digits "zero" through "nine," "yes," "no," and "oh," in US English, or the equivalent words in another language.

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.

structured query language

Structured query language (SQL) is a standard data programming language used with data storage and data query applications.

subordinate nodes

Nodes that are connected to one or more "parent" nodes, which appear above them in the call flow. See also parent nodes.

subword technology

A method of speech recognition, used in FlexWord speech recognition, that recognizes phonemes or parts of words. Compare to phoneme.

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 functions

See standard external functions.

system variables

Variables that are predefined and included as part of the IVR Designer package.

table

See database table.

table schema

The structure or design of a database table, established by defining data record fields for the table.

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.

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.

UNIX Operating System

A multiuser, multitasking computer operating system.

US

United States of America

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.

Voice over IP

A type of communication channel where real time voice communications are transmitted over an internet protocol network.

VoIP

See Voice over IP.

whole-word technology

The ability to recognize an entire word, rather than just a phoneme or a part of a word. Compare to subword technology.

WholeWord speech recognition

An optional feature, available in several languages, based on whole-word technology that can recognize the numbers one through zero, "yes", and "no" (the key words). This feature is reliable, regardless of the individual speaker. This feature can identify the key words when spoken in phrases with other words. A string of key words, called connected digits, can be recognized. During the prompt announcement, the caller can speak or use touch tones.

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

Special Characters

+create	701
+getarg	702
+send	703
+setarg	704

A

A caution about the AllocPROXY external function 729	
A single touchtone and speech recognition application	89
A_Callinfo	659
A_Event	661
A_RouteSel	662
A_Tran	663
About Avaya IVR Designer	13
About Help	35
About node branches	72
About sharing application resources	227
about the Help window	36
About the Help window	36
Add Record node	559
Adding a Disconnect Call node	322
Adding a Return node	323
Adding a Transfer Call Node	321
Adding a Transfer Fail Announcement node	323
Adding an Announcement node	320
Adding an Answer Call node	320
Additional right-click menu options	200
Adjusting the volume of generated audio files	258
adjusting volume	258
Advanced TAS application development tours	436
alarm	713
ANI (Automatic Number Identification)	629
Announcement node	561
Announcements	53
Answer Call node	562
Application errors	60
Application errors, checking for	241
Application examples	62
Application resources	226
Application Transfer window menus	268
Application usability	49
ASAI external functions	658
Assigning the installed application using the Application Install tab	293
Asynchronous Event node	562

Attempts	630
Audio recording tips	178
Automenu node	563
Avaya Interaction Center integration	508
Avaya IVR Designer icons	538

B

Basic TAS application development tours	297
bilingual applications	66
Bilingual applications	66
Branching based on table data	422
Branching based on time, day, or date	348
Breaking a long application into sections	288

C

Call Flow node	570
Call flows	90
Callprg	667
Capturing host screens	203
Capturing voice from callers in TAS applications	487
Channel	630
Checking an application for errors	241
Choosing an alternate view for the Prompt Text field 156	
CI_MODE	631
CI_NO_DIGS_GOT	632
CI_RECOG	632
CLEO external functions	658
Code generation file types	253
Code Generation/Application Transfer tool icons	551
CollectedDigits	633
Collecting information in TAS applications	368
configuring SSH settings	264
Configuring TAS interface elements	307
Configuring the TAS IVR Designer desktop	307
Confirming digit entries with callers	58
Control external functions	665
Control node icons	545
Convert an application	278
Converting character variable values	122
Converting currency variable values	123
Converting date variable values	124
Converting number variable values	125
Converting real variable values	126
Converting Script Builder applications	275
Converting TAS variable values	388
Converting time variable values	127

Converting variable values	121
Converting Voice@Work and IVR Designer applications	25
Copying nodes between TAS applications	357
Copying nodes within TAS applications	355
Copying TAS applications	358
Copying TAS nodes	355
Copying TAS nodes and applications	355
Create the call flows and other application components	26
creating	131–133 , 151 , 172
Creating a basic TAS application	312
Creating a database	409
Creating a FlexWord list	131
Creating a flowchart for an application	317
Creating a host application	498
Creating a Natural Language Speech Recognition (NLSR) word list	132
Creating a new phrase within a prompt	157
Creating a new TAS application	319
Creating a TAS application	318
Creating a TAS phrase table	349
Creating a TAS Time Branch application	350
Creating a voice response application	234
Creating an application using Avaya IVR Designer	26
Creating an embedded call flow in TAS	401
Creating an NLSR word list for a TDD application	133
Creating and speaking TAS variables	391
Creating custom external functions	476
Creating database applications	408
Creating database tables	211
Creating host applications	496
Creating new (custom) variables	116
Creating new external functions	138
Creating new Host Definitions	187
Creating new phrases	172
Creating new prompts	151
Creating phrases in TAS applications	395
Creating stub grammars	133
Creating the host definition	499
CRM Central external functions	671
CTI DIP external functions	671
CtiCallInfo	674
CtiCallState	675
CtiConfer	676
CtiDial	677
CtiDisconn	678
CtiHold	678
CtiPrivData	679
CtiRetrieve	680
CtiTransfer	681
Custom phrase table names	286

D

Data Collection options	237
Data events	234
Database Node icons	549
Database Table Provisioning window	259
Database tables	209 , 210
Date	634
Date/Time external functions	681
Default	634
Defining a Transfer Call node failure announcement	329
Defining an Announcement node prompt	326
Defining fields using the Table Schema Editor	221
Defining successful voice response applications	41
Defining TAS asynchronous events	452
Defining the external function's appearance	146
Delete Record node	573
Deleting a prompt element	164
Developing a step analysis	315
Diagram your application design	45
Dial through and barge-in	65
Dialer external functions	682
Digit input	56
Discard Previous Touchtones always set to True	284
Disconnect Call node	574
Displaying the Used By view	92
Displaying the Uses view	91
DNIS (Dialed Number Identification Service)	635
Docking and undocking resource managers	97
Docking/undocking palettes	557
DoNotCall external function	685
DPR barge-in	99
Dragging a call flow to create a copy	93

E

Edit Menu icons	539
Edit menu options	531
Editing a Host Definition	189
Editing a phrase in a prompt	160
Editing a tableschema	213
Editing a word list	134
Editing an external function	140
Editing existing phrases	172
Editing node attributes	77
Editing prompts in TAS applications	397
Editing tabledata	213
EMPAC	723
EMStart	722
EMStop	723
Enabling DPR barge-in	99
Encapsulating a TAS call flow	403
End_softdisc external function	686

Entering digit sequences	56
Errno	635
Error handling in tas2cpsdip	720
Events String Editor icons	552
Events String Editor menus	236
Examining the simulation script	246
Example of how to use the Screen Mapping tool with the Host Definition Editor	208
External Function palette icons	550
External functions	136

F

F_HOST_ERROR	636
F_HOST_SCREEN	637
FAX external functions	696
Feature Compatibility Warnings	19
File Menu icons	539
File menu options	529
FINDBEST	637
FinishedItem external function	686
FlexWord list	131
FlexWord list, importing	131
FlexWord speech recognition	129
Form Filler external functions	699

G

Generate the source code	29
Generating a menu from database records	411
Generating and transferring an application	249
Generating code and transferring and installing a TAS application	333
Generating TAS code	334
Generating TAS instruction code	249
Generating the source code files	252
GeoTel external functions	700
Getting started with Avaya IVR Designer	25
Getting the most out of Text-to-Speech	68
getvdu	707
getvox	708
Globals	94
gone	712
graphical user interface	23
Guidelines for sharing resources	228

H

Handle_disc external function	687
Handling strings in TAS applications	436
Help	35 , 36
Help Menu icon	545
Help menu options	537

Help topics	36
HookflashLine external function	687
Host application development guidelines	496
Host Connection node	574
Host definition issues	286
Host Flows	201
Host Node icons	549
Host Send node	576
Host Send/Receive node	577
Hosts	185
HOURS_CLOSED	638

I

Importing additional resources	229
Importing external functions	137
Importing FlexWord grammars (word lists)	131
Importing Host Definitions	186
Importing or removing a professionally recorded speech file	295
Importing phrase tables	171
Importing text	181
Inline Code node	579
Inserting a control into a prompt	162
Inserting a new line (alternate view)	164
Inserting a phrase break	161
Inserting a variable into a prompt	160
Inserting an existing phrase into a prompt	159
Install the application on the target system	30
Installing an application on the target system	289
Installing Avaya IVR Designer on your PC	16
Installing TAS applications	338
Installing the application	290
Installing the SCI tools on the target system	18
Interaction Center external functions	700
Internationalization and globalization	22
Introduction to Avaya IVR Designer	13
IVR (Interactive Voice-Response) Node icons	548

L

LANGUAGE	638
Large application, breaking into sections	288
legal notices	2
ListCbackFmt external function	688
Load Variables node	580
Locating information in tables	375
LogloStart external function	689
LogloStop external function	689

M

Creating the Host and	505

Making backup copies of database tables	212	Phrase Editor window elements	175
Maneuvering in the Globals Manager	96	Phrase external functions	724
Mapping tab of the Screen Mapping tool	206	Phrases	172
Menu node	581	PivotYear	640
Menu node attributes	584	Placing a node in a call flow	71
Menus	54	Plan for disabled callers	45
Menus in the Code Generation window	256	Plan the application design	26
Miscellaneous icons	550	Predictive Dialing System (PDS) integration	514
Miscellaneous Node icons	547	Prepare application for conversion	275
Modem	133	Preparing to install an application	290
Modify a touch-tone-only application to include spoken input	64	Presenting menus to callers in TAS applications	359
MoFlashBlind external function	690	PRI external functions	725
More about tabstops	96	printing	35 , 36
Moving variables on the Selected Variables list	236	Printing Help topics	36
MrcpSetParam external function	11	Printing reports	35
multilingual applications	66	Printing the Help file	35
Multilingual applications	67	Privacy property for TAS applications	12
		Prompt and Collect node	588
N		Prompt and Collect node attributes	591
Name conflicts during paste actions	75	Prompts	52 , 150 , 151
New database tables are not the most current	285	Prompts, Response, Confirmation, and Validation 370	
new in release	11	Provide an attendant	45
newcall	705	Providing remote access to database records	426
NICE external functions	714	pseudo_ani	706
NLSR external functions	726		
NLSR speech recognition	129	Q	
NLSR word list (grammar)	132	Query Table Editor	600
NMSPlayGain	668	Query Table node	598
Node descriptions	559		
Node editing guidelines	73	R	
Node Inspector	82	ReadField external function	691
Node Palette icons	545	Rearranging icons on a palette	555
Nodes	71	RecordCounter	641
NoDigit	639	Recorded speech	258
non-US English applications	66	RecordsFound	641
notices, legal	2	Reference	529
		Refreshing shared-in resources	238
O		ReleaseLine external function	691
Offer a user guide	44	Remote database tables do not appear	282
Offer caller training	44	Removing an application	294
Opening a Palette Configuration window	554	Removing variables	235
OPSR	132	Renaming a Host Definition	188
		Renaming a Word List	134
P		Renaming an external function	139
Pace the application	59	Renaming custom variables	117
Palettes	554	Renaming database tables	213
Passing data between TAS applications	480	Renaming prompts	153
Pattern matching	568 , 586 , 596	Reports	35
Performing calculations in TAS	379	Resetting the system	455
		Resolving Script Builder conversion problems	279
		Return node	601

Reusing and Copying TAS Call Flows	405	Set and Test node	603
Right-click menu options for nodes and empty node branches	77	Set Test Branch Condition Editor	611
Right-click menu options for palettes	558	Set Test Expression Editor	605
Right-click menu options for the Application Resources tool	238	Set_early_hu external function	694
Right-click menu options for the Call Flow Inventory tool . 93		Set_Softdisc external function	695
Right-click menu options for the External Function Editor 147		SetCallback external function	693
Right-click menu options for the External Function Source Editor	149	SetNotifyFld external function	694
Right-click menu options for the External Functions Manager	140	Setting application transfer options	261
Right-click menu options for the FlexWords Manager ... 135		Setting Code Generation options	250
Right-click menu options for the Globals Manager	98	Setting or editing variableattributes	118
Right-click menu options for the Host Definition Editor ... 198		Setting TAS preferences	310
Right-click menu options for the Hosts Manager	189	Setting the external function parameters	143
Right-click menu options for the Node Inspector	84	Setting the function arguments attributes	144
Right-click menu Options for the Phrases Manager 173		Setting the Function Return Values attributes (optional)	145
Right-click menu options for the Prompt Editor	164	Setting Transfer Call node options	328
Right-click menu options for the Prompts Manager 154		Setting up privacy property in TAS applications	526
Right-click menu options for the Simulation tool	247	Setting user preferences	30
Right-click menu options for the Table Editor	225	setvdu	710
Right-click menu options for the Tables Manager	214	SFTP service	264
Right-click menu options for the Variables Manager 118		Sharing in TAS resources	346
Right-click menu options for the Verification Results window	242	Sharing Menu icons	544
Running an application simulation	245	Sharing menu options	537
		Sharing out TAS resources	342
S		Sharing TAS resources	342
Save and Exec node	601	Sharing-out an application resource	233
SCI tools, installing on target system	18	Simulating an application	243
SCI variables not found	283	Simulating TAS application performance	332
Screen Capture tab	204	Simulation halted	280
Script Builder conversion problems, resolving	279	Simulation tool icons	552
Scripting an application	313	Speaking caller input in TAS	390
Searching for information in applications	85	Specifying parameters for TAS applications	326
Seasonal greetings do not work	285	Specifying TAS menu error prompts	365
Securing user data	526	Specifying TAS menu response handling	367
Selecting a node to view or edit attributes	83	Specifying TAS Prompt and Collect parameters	372
Selecting multiple nodes for cut, copy, and paste	75	Speech conversion	258
Selecting variables	234	Speech Features external functions	729
Sending a tas2cps request	717	Speech recognition	129
Sending a tas2em request	722	Spoken as options for character variables	108
SendMessage external function	692	Spoken as options for currency variables	109
Separate touchtone and speech recognition applications 88		Spoken as options for date variables	111
		Spoken as options for number variables	112
		Spoken as options for real variables	114
		Spoken as options for time variables	115
		Standard external functions	645
		StartRecording	717
		StartRecording data validation	718
		StartRecording error codes	719
		StopRecording	719
		StopRecording error codes	720
		StoredProc	668
		StoredProcNext	670
		Stub grammar	133
		System requirements	15

System variables	626	The Table Schema Editor	215
T		The Unused tab	87
TAS Application design overview	313	The Use All Selected Records in Prompt option	166
TAS application development tours overview	297	The Uses Of tab	86
TAS Globals	303	The Variables Manager window elements	101
TAS IVR Designer interface	298	Time	642
TAS Menu error prompts and response handling	364	Time Branch node	614
TAS Menus	299	Time Branch nodes show redundant date branches	285
TAS Node Palette	302	Tone masking	598
TAS Spoken As formats	388	Toolbar icons	538
TAS Substring and Parse comparison	446	Tools Menu icons	541
TAS to CPS (tas2cps)	715	Tools menu options	535
TAS to Event Manager (tas2em)	720	Touch-tone and speech recognition	62
TAS to VXI node	612	Touchtone input used with spoken input	88
TAS to VXI node attributes	612	tr	711
TAS Toolbar	302	Trace Variables node	615
TAS Variable formats	386	Tracing a TAS application	521
TAS Variable types and formats	386	Tracking system events	417
TAS Variables and arithmetic operations	387	transfer	712
tas2cps components	716	Transfer Call node	617
tas2em components	722	Transfer the application to the target system	29
TAS2VXI node	612	TRANSFER_RESULT	642
TAS2VXI node attributes	612	Transferring an application from floppy disks	296
TAS2VXI node enhancements	12	Transferring an application to the target system	265
TDD application	133	Transferring application source files	267
Telephony external functions	732	Transferring files from floppy disks to the target system	267
Telephony Node icons	546		267
Testing a TAS application	330	Transferring files using a modem	269
Testing and troubleshooting TAS applications	521	Transferring screen-capture files from the target system	206
Testing Text-to-Speech applications	69		206
Text-to-Speech applications	67	Transferring TAS applications	335
The Application Parameters tab	31	Transferring the application	261
The code generation process	255	Tries	643
The Environment tab	33	Two B-Channel Transfer	725
The External Function Editor	141		
The External Function Source Editor	148	U	
The Field Definition tab (optional)	194	UCID external functions	699
The Find Text tab	87	UNIX_TIME	644
The Globals Manager	94	Update Table Editor	622
The graphical user interface	23	Update Table node	621
The Helper DIP	196	UpdateField external function	695
The Host Definition Editor	190	Updating shared-out resources	233
The Host Definition tab	191	Use flowcharts	46
The Logins tab	193	Use human factors and usability engineering resources	43
The Named tab	85		43
The New Phrase Assignments window	157	Use outlines	46
The Node Inspector	82	Use prerecorded speech	43
The Phrase Editor	175	Use touch-tone input when speech recognition fails	63
The Phrase Table Editor	179		63
The Printing tab	35	Using asynchronous events in TAS applications	451
The Prompt Editor	155	Using both Text-to-Speech and prerecorded speech	
The Table Editor	223	prompts and announcements	68

Using calculations in TAS applications	379	Using the Voice capture node	487
Using Dialer external functions	514	Using variables and speech in TAS applications	385
Using drag-and-drop to cut, copy, and paste	76		
Using embedded call flows in TAS applications	400	V	
Using external functions in TAS applications	456	Validate a digit sequence entry	57
Using Inline Code	475	Variable comparisons	128
Using intelligent transfer	383	Variables	99, 626
Using Prompt and Collect in TAS applications	369	Variables and arithmetic operations	119
Using Set and Test to change TAS variable types	390	Variables are not spoken correctly	284
Using stored procedures in TAS applications	483	Verifying a TAS application	331
Using TAS menus	359	Verifying and testing an application	241
Using TAS variable attributes	387	Vesp_dip	700
Using Text-to-Speech for prompts and announcements	67	View Menu icons	540
Using the Allow Scrolling option	556	View menu options	533
Using the Application Install tab	290	Viewing external functions	474
Using the CLEO Screen Capture Utility	207	Viewing results with SQL*Plus	524
Using the Create Default Phrases dialog box	184	Viewing shared resources	232
Using the Datetime_u and Getday external functions	461	Voice application basics	39
Using the Execute and Getarg external functions	468	Voice Capture node	623
Using the Execute external function	465	Voice Capture node attributes	624
Using the FAX external functions	471	Voice response application uses	39
Using the Jump to Node function	324	Voice response system versus an attendant	41
Using the Make_Call and Sleep external functions	460	VoIP station transfer	663
Using the Make_Call external function	457	Vonetix external functions	733
Using the Mon_year external function	464		
Using the Msg_Delete external function	491	W	
Using the Phrase Editor	177	Window menu options	534
Using the Phrase Text Import dialog box	182	Windows 2000 PPP configuration	271
Using the Save and Exec node and Load Variables node	480	working with ...71, 90, 94, 99, 136, 150, 185, 209, 210, 226, 234	
Using the sci command	292	Working with application resources	226
Using the StoredProc and StoredProcNext external functions	484	Working with call flows	90
Using the Table Editor	224	Working with data events (Events String Editor)	234
Using the Tabs option	556	Working with database tables	209, 210
Using the TAS Concat expression	442	Working with external functions	136
Using the TAS ItemCount and ItemOf expressions	440	Working with hosts	185
Using the TAS Length expression	437	Working with nodes	71
Using the TAS Parse expression	450	Working with phrases	169
Using the TAS Substring expression	447	Working with prompts	150
Using the Vesp_dip external function	509	Working with speech recognition	129
		Working with variables	99
		Writing node notes	83
		Y	
		Yes/no questions	59
		YesDigit	644

