IVR Interface Option 8.1

IVR Driver for WVR for AIX

System Administrator's Guide

About Genesys

Genesys is the world's leading provider of customer service and contact center software - with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service - and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to www.genesyslab.com for more information.

Each product has its own documentation for online viewing at the Genesys Technical Support website or on the Documentation Library DVD, which is available from Genesys upon request. For more information, contact your sales representative.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Genesys Telecommunications Laboratories, Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Genesys products.

Trademarks

Genesys, the Genesys logo, and T-Server are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders. © 2012 Genesys Telecommunications Laboratories, Inc. All rights reserved.

The Crystal monospace font is used by permission of Software Renovation Corporation, www.SoftwareRenovation.com.

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the regional numbers provided on page 9. For complete contact information and procedures, refer to the ***Genesys Technical Support Guide***.

Ordering and Licensing Information

Complete information on ordering and licensing Genesys products can be found in the ***Genesys Licensing Guide.***

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 81iv_ad_wvr-aix_08-2012_v8.1.001.00

Table of Contents

List of Procedures 5

Preface 7

About IVR Interface Option 8 7
Intended Audience 8
Making Comments on This Document 8
Contacting Genesys Technical Support 9
Document Change History 9

Chapter 1 **IVR Driver Overview** 11

New in Release 8.1 11
Deployment 11
Architecture 12
- IVR Server 12
- IVR Driver 15
- IVR Library 15
- IVR Server Redundancy Methods 15

Chapter 2 **Pre-Installation Setup for the IVR Driver** 19

Before Configuring the IVR Driver 19
- Component Compatibility 19
- Installing the IVR 19
- Installing the IVR Server 20
- Installing LCA and Other Configuration 20

Configuring the IVR Driver 21
- Defining IVRs and IVR Ports 21
- Configuring IVR Driver Options 21
- Managed Service Availability Parameters 22

Chapter 3 **Installing the IVR Driver** 25

Identifying the Driver Version 25
- Example for WVR for AIX 25

Supported IVR Versions 26
Installing the IVR Driver 26
Importing d2is*.tar into WVR for AIX 27
Custom Server Installation 28
Configuring d2is Custom Server Properties 30
Configuring Multiple IVR Drivers for WVR for AIX 32
Upgrading the I-Library 33
Testing the Installation and Configuration 34

Chapter 4 **Starting and Stopping the IVR Driver 35**

Prestart Information 35
Starting the IVR Driver 35
Stopping the IVR Driver 36
Automatically Stopping the IVR Driver 37

Chapter 5 **Functions 39**

Input Constraints 39
Genesys-Provided Functions 40
Notify Functions 42
Telephone Functions 44
User Data Manipulation Functions 50
Call Information Functions 53
Call Data Transfer Functions 54
Route Functions 56
General-Purpose Functions 62
Statistical Functions 64

Appendix **Customer Test Package 67**

Introduction 67
Configuring and Using the Customer Test Package 67
CTP Voice Menu 69

Supplements **Related Documentation Resources 73**

Document Conventions 75

Index **..... 79**

List of Procedures

Installing LCA and Other Configuration 20
Un-tarring the Genesys software package (WVR) 26
Importing d2is*.tar into WVR for AIX 27
Installing the custom server 29
Defining main() arguments 30
Creating multiple IVR Drivers 32
Implementing the I-Library file 33
Compiling multiple IVR Drivers 33
Starting the IVR Driver manually 35
Automatically starting the IVR Driver 36
Manually stopping the IVR Driver 36
Configuring and running the CTP 67

Preface

Welcome to the *IVR Interface Option 8.1 IVR Driver for WVR for AIX System Administrator's Guide.* This document describes the IVR Driver for WVR for AIX, which is the driver component of the Genesys IVR Interface Option 8.1. This document also describes the Genesys-provided functions supported in IVR Interface Option 8.1.

The full product name for the vendor-provided IVR that this Genesys driver supports is IBM WebSphere Voice Response for AIX. However, in this document, the IVR product is generally referred to as *WVR for AIX*.

Note: For versions of this document created for other releases of this product, visit the Genesys Technical Support website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

This document is valid only for the 8.1 release of this product.

This preface contains the following sections:

- About IVR Interface Option 8, page 7
- Intended Audience, page 8
- Making Comments on This Document, page 8
- Contacting Genesys Technical Support, page 9
- Document Change History, page 9

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 73.

About IVR Interface Option 8

Interactive Voice Response (IVR) technology has emerged as an integral part of contact centers, financial institutions, and the travel industry. IVR components provide the initial interface when a client calls a business. Using IVRs, businesses can realize significant savings, and customers can conduct their business more efficiently.

The IVR Interface Option 8.1 architecture simplifies the integration of vendor-provided IVRs with the Genesys environment. Genesys IVR Interface Option 8.1 has two major components, the IVR Server and the IVR Driver. For more information about these and other IVR Interface Option 8.1 components, see Architecture, page 12.

Intended Audience

This document is primarily intended for contact center administrators, contact center managers, operations personnel, and IVR developers, and assumes that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications
- Network design and operation
- Your own network configurations

You should also be familiar with the Genesys Framework architecture and functions.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to Techpubs.webadmin@genesyslab.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Technical Support if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Genesys Technical Support

If you have purchased support directly from Genesys, contact Genesys Technical Support at the regional numbers below.

Note: The following contact information was correct at time of publication. For the most up-to-date contact information, see the Contact Information on the Tech Support website. Before contacting technical support, refer to the *Genesys Technical Support Guide* for complete contact information and procedures.

Genesys Technical Support Contact Information

Region	Telephone	E-Mail
North America and Latin America	+888-369-5555 (toll-free) +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-1276-45-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp
India	000-800-100-7136 (toll-free) +61-7-3368-6868	support@genesyslab.com.au

Document Change History

This is the first release of the *IVR Interface Option 8.1 IVR Driver for WVR for AIX System Administrator's Guide.* In the future, this section will list topics that are new or that have changed significantly since the first release of this document.

Genesys

Chapter

1 IVR Driver Overview

This chapter describes the architecture of Genesys IVR Interface Option 8.1, and how the IVR Driver is used in this solution. It also discusses deployment tasks and tips. This chapter contains the following sections:

- New in Release 8.1, page 11
- Deployment, page 11
- Architecture, page 12

New in Release 8.1

The following changes have been implemented in release 8.1 of the IVR Driver for WVR for AIX:

- A new `Typeinfo` argument of `UUID` has been added to the `GetCallInfo` function. See "GetCallInfo" on page 53 for details.
- The `Typeinfo` argument of `All` has been extended to optionally include the `UUID` parameter.
- IVR Driver now supports IPv6. For more information, refer to the *Framework 8.1 Deployment Guide.*

Deployment

This *System Administrator's Guide* describes how to install the IVR Driver for WVR for AIX, and how to configure it in Configuration Manager.

The IVR Interface Option 8.1 deployment process includes the installation and configuration of the following components:

- IVR Driver
- IVR Server

Before deploying IVR Interface Option 8.1, see the deployment planning chapters in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

In order for the IVR Driver to operate successfully, the IVR Server must be installed and running. For compatible IVR Driver and IVR Server releases, see the *Genesys Migration Guide.*

For information about installing and configuring the IVR Server, see the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

For information about the Genesys functions provided for this IVR Driver, see Chapter 5 on page 39. For information about the Customer Test Package, see the Appendix on page 67.

Architecture

This section describes the architecture of IVR Interface Option 8.1. This software application integrates vendor-provided Interactive Voice Response (IVR) software and hardware with Genesys Framework.

IVR Server

IVR Server, the key component of Genesys IVR Interface Option 8.1, provides the following functionality:

- Tracks call flow
- Interfaces multiple drivers with multiple T-Servers
- Works with other Genesys services (such as T-Server, Stat Server, and Universal Routing Server)
- Can be used in Load Sharing or Warm Standby mode

Genesys provides the following configuration modes for the IVR Server:

- IVR-Behind-Switch, a basic configuration in which a T-Server that is connected to the premise switch (using computer-telephony integration [CTI] links) can monitor the call activity on IVR channels. For more information, see "IVR-Behind-Switch Configuration."
- IVR-In-Front, in which a CTI link is not involved in the call processing. For more information, see "IVR-In-Front Configuration."
- IVR Network T-Server, in which the IVR Server is a link to a user-provided Network IVR application. The Service Control Point (SCP) and a Genesys Network T-Server are used to redirect calls to the Network IVR for processing. In this mode, IVR Server functions as a Network T-Server. Although Genesys IVR Drivers do not support the IVR Network T-Server configuration mode, you can use it with a driver that you build

using the IVR XML SDK. For more information about the IVR Network T-Server configuration mode, see the *IVR Interface Option 8.1 IVR Server System Administrator's Guide.*

Library Usage

IVR Server is built with 8.x T-Library, and therefore it can connect to T-Servers. IVR Server supports connection to a regular T-Server, the `TServer_IVR` function of an IVR Server operating in IVR-In-Front mode, and a Network T-Server.

IVR-Behind-Switch Configuration

In the IVR-Behind-Switch configuration mode, an incoming call arrives at the premise switch before going to the vendor-provided IVR (see Figure 1). The premise switch and T-Server are at the same site.

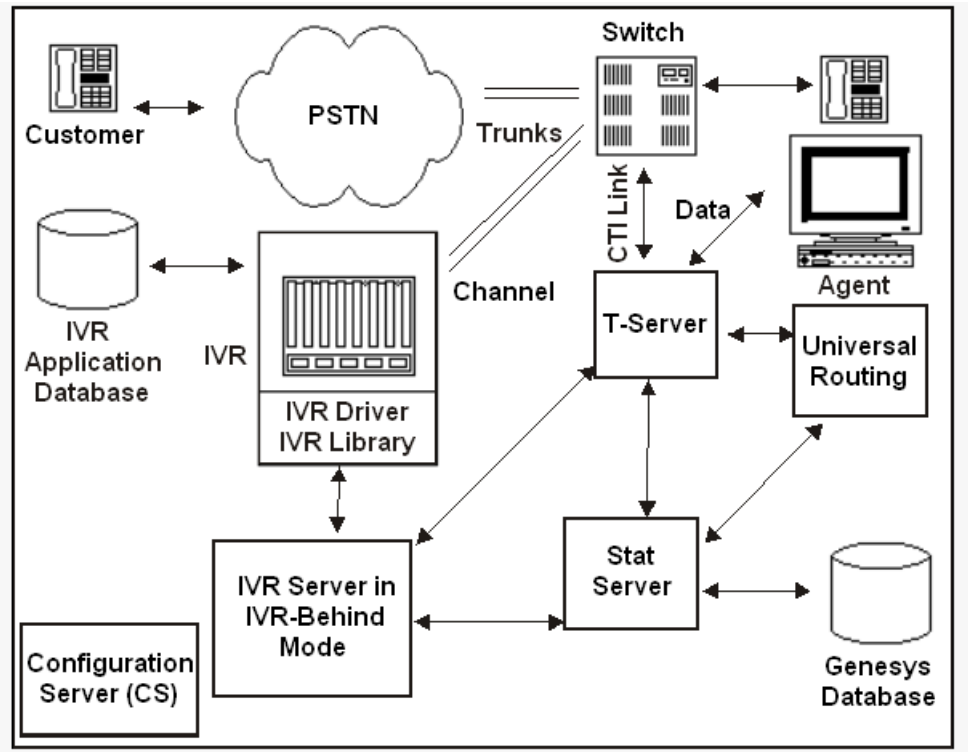


Figure 1: IVR-Behind-Switch Configuration

In this configuration, a T-Server is connected to a premise switch, and the IVR is connected directly to both the switch (through phone lines) and the IVR Server (through data lines). The IVR Server communicates with T-Server and Stat Server.

IVR-In-Front Configuration

If a vendor-provided IVR is connected directly to the PSTN (Public Switched Telephone Network), without a premise switch, the configuration is called IVR-In-Front. In the Site A configuration shown in Figure 2, there is no T-Server to connect to, because there is no premise switch. The `TServer_IVR` function resides within the IVR Server.

An IVR Server operating in IVR-In-Front mode supports IVRs that are connected directly to a PSTN, by performing functions similar to those of a regular T-Server. If an IVR is considered a termination point for incoming calls, no premise switch is involved, and no local T-Server receives notification of the incoming call. Instead, the IVR Server provides this functionality.

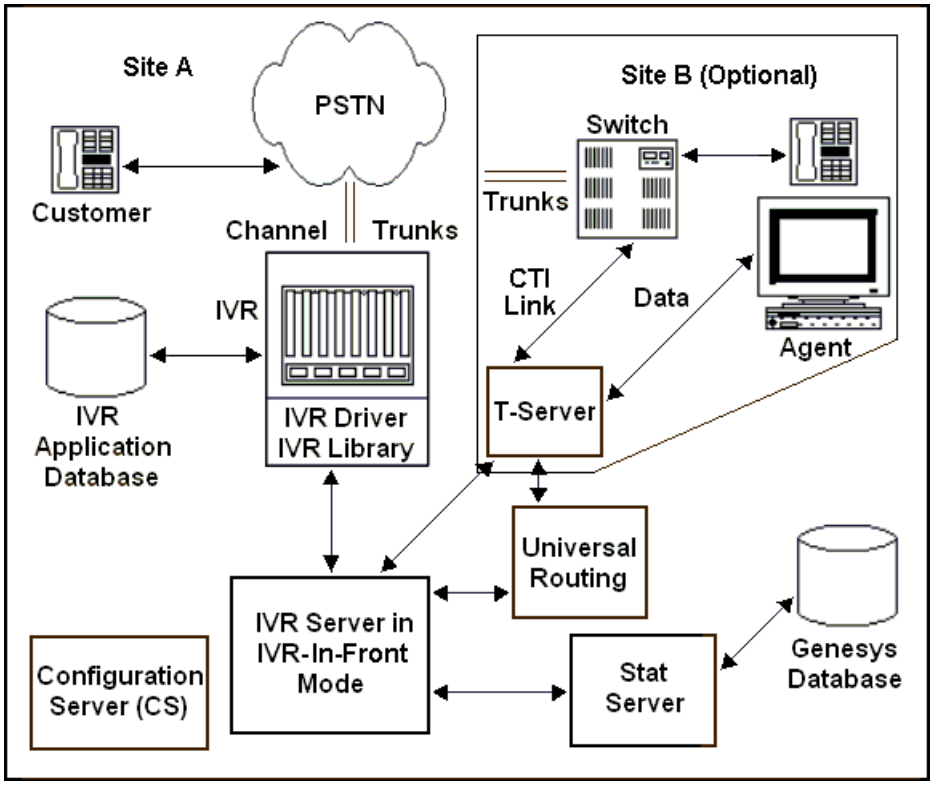


Figure 2: IVR-In-Front Configuration

In the IVR-In-Front configuration shown in Figure 2, Site A is configured for IVR-In-Front mode. The IVR Server communicates with the IVR, the Universal Routing Server, and Stat Server. The IVR Server also simulates a T-Server, which enables it to communicate with other T-Servers, such as the T-Server at Site B. The IVR at Site A is physically connected to the public telephony network for phone lines, and to the IVR Server for data lines.

Site B includes a physical switch connected to a T-Server, which, in turn, provides data to agents in an agent pool.

This distributed configuration across Sites A and B enables coordinated Call Data Transfers.

IVR Driver

The IVR Driver component integrates vendor-specific IVR hardware and software with the Genesys environment. This adds to the IVR's user interface a set of functions or calls that can be used to generate scripts and to integrate the vendor-provided IVR with the Genesys environment. All interactions between the IVR Driver and other IVR Interface Option components are based on the request-response architecture of the IVR Library and use a TCP/IP connection.

The major functions provided by the IVR Driver include:

- Telephony function support (such as transfer, conference, answer, and release).
- Call data manipulation (such as attach, update, and delete).

Each vendor-provided IVR needs one Genesys IVR Driver in order to operate in the Genesys environment. If you want to run vendor-provided IVRs from various manufacturers, each must have a corresponding IVR Driver that is designed for it.

IVR Library

The IVR Library component is typically used to return any IVR Driver error messages. It is embedded in Genesys IVR Drivers in releases up to and including release 8.1. From release 8.0, IVR Library is also available as a stand-alone SDK product.

With IVR Interface Option 8.1, the IVR Library's communication interface uses the industry-standard XML (eXtensible Markup Language) protocol for the transport layer. For more information about the XML interface, see the *IVR SDK 8.1 XML Developer's Guide*, which is available only with purchase of the Genesys IVR SDK.

For more information about the IVR Library interface, see the *Genesys Developer Program IVR SDK 8.1 C Developer's Guide*.

IVR Server Redundancy Methods

You can achieve redundancy for IVR Servers by using either Warm Standby, Hot Standby or Load Balancing. The following sections briefly describe each of these configuration modes in turn. For more information about how to

configure the IVR Server applications, see the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

Note: You configure Load Balancing, Warm Standby and Hot Standby in Configuration Manager, on the `Options` tab of the `Properties` dialog box for the `I-Server` and `IVR_Driver` applications. For more information, see the concepts and the configuration option sections in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide.*

Warm Standby

If two IVR Servers are in a Warm Standby configuration, the primary IVR Server handles all calls on all IVR ports until it fails. At that point, the IVR Server that is configured as the Warm Standby backup server takes over the load.

Benefit You can perform Inter Server Call Control (ISCC) routing with the IVR Servers as the target of an ISCC transfer.

Drawback If the primary server fails, you lose all calls that were in progress at the time of failure.

Hot Standby

If two IVR Servers are in a Hot Standby configuration, the backup server application remains initialized, clients connect to both the primary and backup servers at startup, and the backup server data is synchronized from the primary server. Data synchronization and existing client connections to the backup guarantee higher availability of a component.

Benefits You can perform Inter Server Call Control (ISCC) routing with the IVR Servers as the target of an ISCC transfer.

If the primary server fails, all call interactions in progress are maintained and call loss is minimized or eliminated.

Drawback Load balancing is not supported.

Load Balancing

If two or more IVR Servers are in a Load Balancing configuration, calls on the IVR ports are distributed according to the following formula:
`<number of ports> modulo <number of active IVR Servers>`

If one IVR Server fails, this configuration enables the surviving IVR Server(s) to continue handling their current calls, and new incoming calls are distributed according to the preceding formula, with the number of active IVR Servers

now decreased by one. When the failed IVR Server is restored, it is automatically added back into the distribution.

Benefit If an IVR Server fails, you lose only the active calls that were in progress on that server.

Drawback You cannot perform ISCC routing with the Load Balancing IVR Servers as the target of an ISCC transfer (with the exception of DNIS pool type ISCC when all TSCP components including IVR Server must be at version 7.5.009.01 as a mnimum, and the URS must be at version 7.5.002.02 as a mnimum).

Note: If you want redundancy with scalability and you do not need to use ISCC, load balancing provides better reliability. This is because as the number of IServer instances increases, the proportion of calls lost decreases, as follows:

```
1 / <number of Load Balancing IVR Servers>
```

For example, if you configure three Load Balancing IVR Servers, and one fails, you will lose one-third of the calls that are in progress—namely, the calls on the IVR Server that failed.

If the primary concern is reliability, hot standby is the better choice.

Limitation

Hot Standby mode and Load Balancing cannot operate together.

Chapter

2 Pre-Installation Setup for the IVR Driver

This chapter describes the steps that you must perform before you can successfully install the IVR Driver. It contains the following sections:

- Before Configuring the IVR Driver, page 19
- Configuring the IVR Driver, page 21

Before Configuring the IVR Driver

Before you configure the IVR Driver, you must complete the tasks described in this section.

Component Compatibility

Important: Before you can configure Configuration Manager objects and install IVR Interface Option 8.1, you must install a supported release of Genesys Framework. For IVR-Behind-Switch configurations, you must also install a compatible release of Genesys T-Server. For more information, see the *Framework 8.1 Deployment Guide*, and the IVR Interface Option 8.1 chapters in the *Genesys Migration Guide*.

Installing the IVR

Before you manually install the IVR Driver for WVR for AIX on the AIX operating system, you must install both the hardware and software for the WVR for AIX IVR application. For more information, see the vendor-provided WVR for AIX IVR documentation.

Install the IVR Driver only on the computer on which the vendor-provided application is installed. Also, keep in mind that the IVR Driver is intended to be used with the IVR Server.

Installing the IVR Server

You can install the IVR Server on any computer that belongs to the site where the IVR Interface Option 8.1 product is used (including the computer on which the IVR Driver is installed). However, make sure that the operating system of the host on which you install the IVR Server matches the operating system on which the IVR Server was built.

Note: Genesys strongly recommends that, before installing the IVR Driver, you install the IVR Server, and that you configure it, the `IVR` object, and the `IVR_Driver` application in Configuration Manager.

For information about setting up and configuring the Genesys IVR Server, see the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

Installing LCA and Other Configuration

Procedure: Installing LCA and Other Configuration

Purpose: To enable correct startup and function of IVR Driver.

Start of procedure

1. Install the Genesys Local Control Agent (LCA) on the same computer as the IVR Driver, and then restart the computer. LCA is enabled and starts automatically during system startup.
2. In Configuration Manager, create a `Host` object for your IVR.
3. Open the `Properties` dialog box for the `IVR_Driver` application, and click the `Server Info` tab.
4. In the `Name` box, enter the IVR host name that you specified for the IVR's `Host` object.
5. Configure a log file name that is valid for the operating system on which the IVR Driver is running.

Note: If you omit any of these steps, the IVR Driver will not operate or log properly.

End of procedure

Configuring the IVR Driver

This section describes how to configure the IVR Driver.

Note: For information about how to migrate from IVR Driver 6.x releases, see the *Genesys Migration Guide.*

Defining IVRs and IVR Ports

You can use the IVR Interface Option Wizard to define an `IVR` object, and to define an entire range of IVR ports at once. For information about how to use the wizard, see the wizard configuration chapter in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

You can also define an `IVR` object or a single IVR port manually in Configuration Manager. For more information, see the manual configuration chapter in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

After you define an `IVR` object in Configuration Manager (either by using the wizard or manually), you must configure the IVR for use with an IVR Driver, using the procedure in "Configuring IVR Driver Options."

Configuring IVR Driver Options

You must configure all configuration options for the IVR Driver in Configuration Manager, on the `Options` tab of the `IVR_Driver` application's `Properties` dialog box. For information about how to import and configure the `IVR_Driver` application, see the pre-installation setup chapter and the IVR configuration options chapter in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

You can use the IVR Interface Option Wizard to configure the `IVR_Driver` application and its options. For more information, see the wizard configuration

chapter in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

Note: IVR Driver configuration options that in 6.x releases were configured on the `Annex` tab of the `IVR` object's `Properties` dialog box are configured in the `IVR_Driver` application starting with release 7.2. For more information, see the *Genesys Migration Guide*.

For each IVR Driver running on the same IVR, you must define a separate `IVR_Driver` Application.

Managed Service Availability Parameters

The parameters to enable and configure the managed service availability features are located on the `Annex` tab of the `IVR` object's `Properties` dialog box in the `AgentControl` section (see the IVR configuration options chapter in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*).

AgentControl

There are four parameters in the `AgentControl` section: `LegacyMode`, `DriverIgnoreReady`, `DriverReadyWorkMode`, and `DriverRetryTimeout`. These parameters are used to specify which `AgentControl` values IVR Library expects, and what effect they have. Any values other than those described in this section are ignored.

Warning! It is important to set the `Shutdown Timeout` option (on the `Server Info` tab of the `IVR_Driver` application's `Properties` dialog box) to a time interval that will allow all calls to end normally on the IVR. Any calls that are still in progress when the timer expires will be terminated immediately.

Note: All parameter names are case-sensitive.

LegacyMode

Default Value: `true`

Valid Values: `true` The IVR Server controls the agent activity.

`false` The IVR Driver controls the agent activity.

Changes Take Effect: Immediately

Specifies whether the IVR Server or the IVR Driver controls agent state, to provide consistency with the values configured for the IVR ports in the `IVR` object.

Notes: If `LegacyMode` is set to `false`:

- Only one IVR Driver at a time may connect to this `IVR` object.
- Dynamic disabling of IVR ports while the IVR Driver is running is not supported. If you attempt to dynamically disable IVR ports while the IVR Driver is running, the IVR port's agents might not being logged off.

DriverIgnoreReady

Default Value: `false`

Valid Values:

`true`	The IVR Driver ignores the `SetReady` parameter.
`false`	The IVR Driver attempts to set agents to the configured `SetReady` state.

Changes Take Effect: Immediately

Specifies whether the IVR Driver attempts to use the `SetReady` parameter. This option applies only when `LegacyMode` is set to `false`.

DriverReadyWorkMode

Default Value: `ManualIn`

Valid Values:

`ManualIn`	The IVR Driver sends `AgentReady` and `AgentNotReady` messages with `workmode` = `ManualIn`. An `AgentReady` message is sent whenever an `AgentNotReady` status is received from IVR Server.
`AutoIn`	The IVR Driver sends `AgentReady` and `AgentNotReady` messages with `workmode` = `AutoIn`.
`Unknown`	The IVR Driver sends `AgentReady` and `AgentNotReady` messages with `workmode` = `Unknown`.

Changes Take Effect: Immediately

Specifies the `workmode` for `AgentReady` and `AgentNotReady` messages. This option applies only when `LegacyMode` is set to `false`.

DriverRetryTimeout

Default Value: `60`

Valid Values: Any integer > `0`

Changes Take Effect: Immediately

Specifies the amount of time (in seconds) that the IVR Driver waits to make another attempt, after receiving an error message from IVR Server for a previous `AgentControl` message on that port. This option applies only when `LegacyMode` is set to `false`.

Genesys

Chapter

3 Installing the IVR Driver

This chapter describes how to install the IVR Driver. It contains the following sections:

- Identifying the Driver Version, page 25
- Installing the IVR Driver, page 26
- Importing d2is*.tar into WVR for AIX, page 27
- Custom Server Installation, page 28
- Configuring d2is Custom Server Properties, page 30
- Testing the Installation and Configuration, page 34

Before you install the IVR Driver, complete the tasks described in Chapter 2 on page 19.

Identifying the Driver Version

To help users identify the version number of the IVR Driver, the file name for the IVR Driver for WVR for AIX package consists of the following subfields, separated by underscores:

- The name and version of the vendor-provided IVR
- The name and version of the operating system on which the product was designed to run
- The Genesys IVR Interface Option type
- The Genesys IVR Driver version

Example for WVR for AIX

```
dtalk4-3_AIX5-3_IS_8-1-0-000-00.tar
```

The file name in the example indicates the following:

1. `dtalk4-3`—This Genesys IVR Driver package was created for the WVR for AIX IVR, version 4.3.
2. `AIX5-3`—It runs on the AIX version 5.1 or later operating system.
3. `IS`—The IVR Interface Option type is `IS` (IVR Server).
4. `8-1-000-00`—The Genesys IVR Driver version is 8.1.000.00.
5. `.tar`—The package is self-installing.

Supported IVR Versions

For supported WVR versions and operating systems, refer to the *Genesys Supported Media Interfaces Reference Manual*.

Installing the IVR Driver

Procedure:
Un-tarring the Genesys software package (WVR)

Purpose: To untar the Genesys software package so that you can install the IVR Driver for WVR for AIX (`d2is`).

Start of procedure

1. Become `superuser`.
2. Insert the IVR Driver 8.1 installation DVD.
3. On the local WVR for AIX host, create a directory called `/home/dtuser/gcti` and ensure that it has 20 MB of free space.
4. Locate the `directtalk` folder on the installation DVD.
5. Open the subfolder for the operating system that you use for your IVR, and locate the `.tar` file (for example, `dtalk4-3_AIX5-3_IS_8-1-0-000-00.tar`).
6. Copy the `.tar` file into the `/home/dtuser/gcti` directory that you created on the local IVR host.
7. At the command line, enter the command to untar the package file—for example:

   ```
   tar -xvf dtalk4-3_AIX5-3_IS_8-1-0-000-00.tar
   ```

   The preceding command extracts the following files from the package file:

   - `d2is_aix53.tar`—This file contains the Genesys IVR Driver for WVR for AIX and the Customer Test Package (CTP) for use on AIX 5.1 or later.

- `install.man`—This text file contains brief instructions for installing, configuring, and starting the interface.
- `Genesys_libs.tar`—This file contains the libraries used by the driver.
- `install_libs`—This shell script is used to install the libraries.

8. Ensure that the `d2is` driver (the custom server) is not already running. Then, still as `superuser`, execute the command `install_libs`.

End of procedure

Importing d2is*.tar into WVR for AIX

To import the `d2is*.tar` file (which contains the `d2is` driver file and the CTP) into the WVR for AIX IVR application.

Note: You must import the `d2is_aix53.tar` file, regardless of whether you are installing the IVR Driver on any of the AIX 5.x operating systems.

Procedure: Importing d2is*.tar into WVR for AIX

Start of procedure

1. Open the WVR for AIX application.
2. In the `WVR for AIX Welcome` window, click `Applications`.

   The `Applications` dialog box appears (see Figure 3).

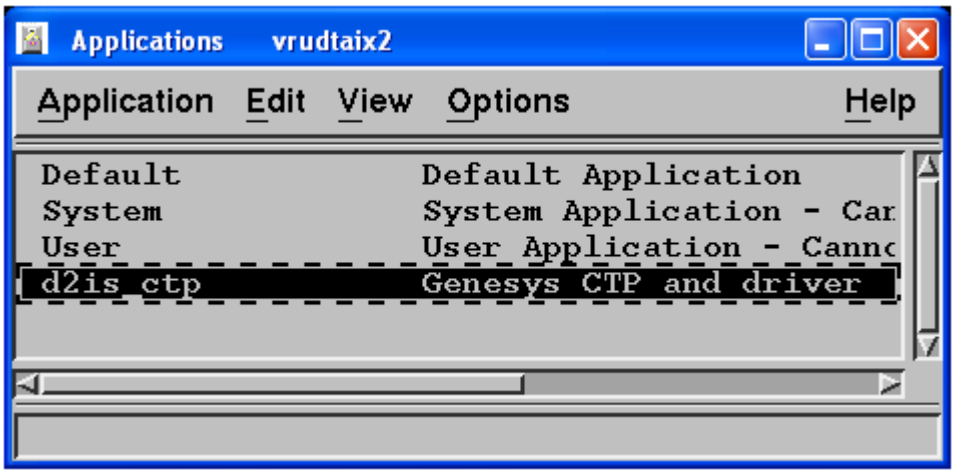


Figure 3: Applications Dialog Box

3. Select `Application > Import > Replace > File`. The `Import File` dialog box appears (see Figure 4).

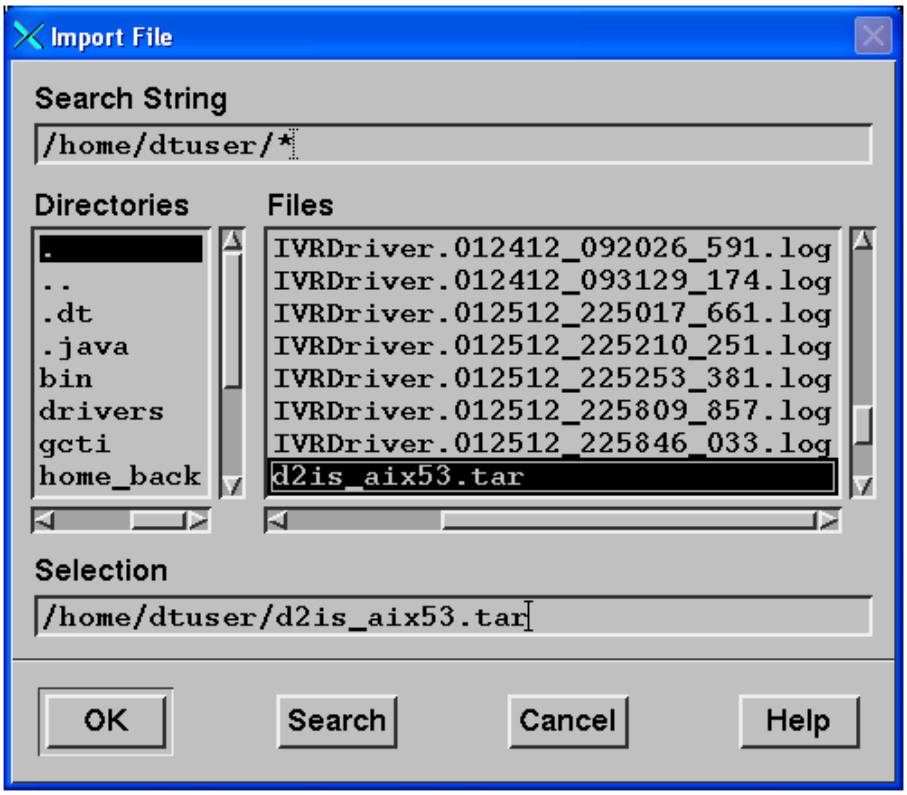


Figure 4: Import File Dialog Box

The `d2is_aix53.tar` file should be listed in the `Files` box. The path shown in the `Search String` box is the directory where each package is located.

4. Select `d2is_aix53.tar`, and click `OK` to import the file.

   For more information about importing objects into the WVR for AIX application, see the WVR for AIX IVR documentation.

End of procedure

Custom Server Installation

After importing the `d2is` custom server and the Customer Test Package, install the `d2is` custom server using the menu on the WVR `Welcome` window.

Procedure:
Installing the custom server

Start of procedure

1. In the WVR for AIX `Welcome` window, select `Applications > Custom Server`. The `Custom Server...` dialog box appears.
2. Locate and double-click the `d2is` line in the list. The `Custom Server (d2is)` dialog box appears (see Figure 5).

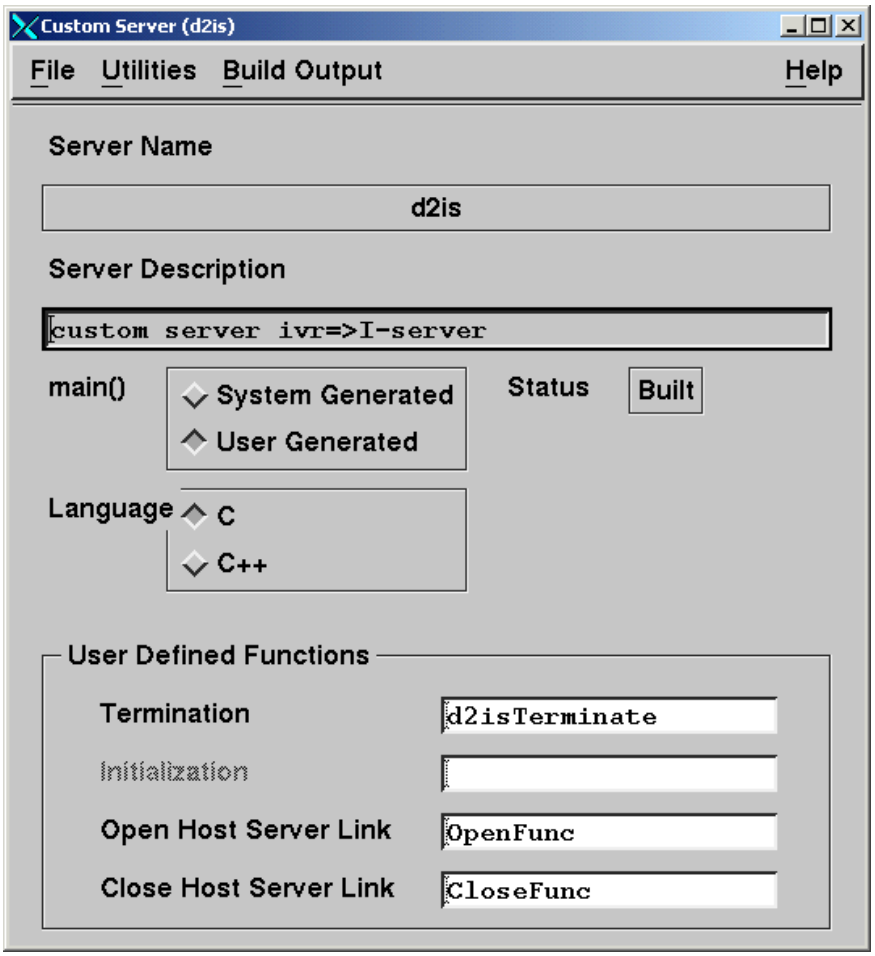


Figure 5: Custom Server (d2is) Dialog Box

3. Select `Utilities > Install`.

End of procedure

Next Steps

After installation, the `Custom Server (d2is)` dialog box remains open, so that you can configure the custom server properties as described in "Configuring d2is Custom Server Properties."

Configuring d2is Custom Server Properties

The `d2is` custom server requires a connection to IVR Server. To establish this connection, you must configure parameters for `d2is` in the WVR for AIX IVR application. You must define these parameters as `main()` arguments, which can be updated and saved without rebuilding or re-installing the custom server. Modified `main()` arguments are used the next time the custom server is started.

Procedure: Defining main() arguments

Start of procedure

1. In the `Custom Server (d2is)` dialog box, select `File > Properties`. The `Properties (d2is)` dialog box appears (see Figure 6).

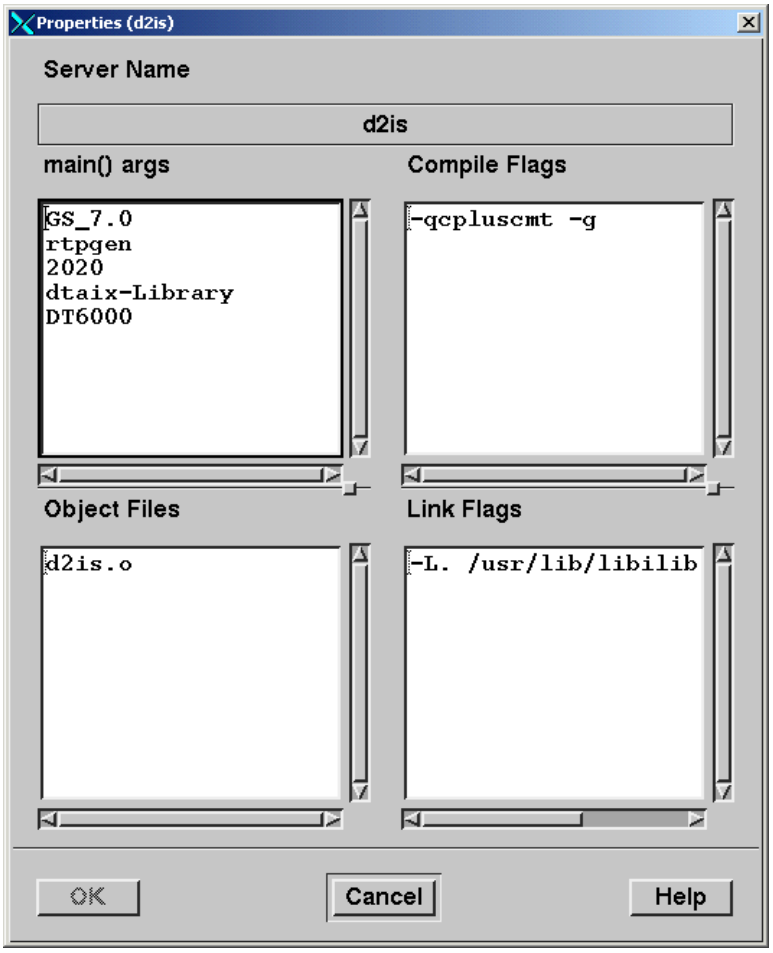


Figure 6: Properties (d2is) Dialog Box

2. Edit the contents of the `main() args` box:
   a. On the first line, enter `GS_7.0`.
   b. On the second line, enter the name of the host where the Configuration Server will be running.
   c. On the third line, enter the communications port number of the Configuration Server, as specified in the Configuration Server application's `Properties` dialog box in Configuration Manager. The default value is `2020`.
   d. On the fourth line, enter the name of the IVR Driver, as specified in the `IVR_Driver` application's `Properties` dialog box in Configuration Manager.

   Note: For each IVR Driver that is running on the same IVR, you must define a separate IVR Driver application.

   e. On the fifth line, enter the name assigned to the IVR object in Configuration Manager (for example, `DT6000`).

   The following parameters are optional:

   f. On the sixth line, enter the number of milliseconds between queue reads. The default value is `10`.
   g. On the seventh line, enter the number of `Receive` messages processed. The default value is `15`.
   h. On the eighth line, enter the number of `GetReply` functions processed. The default value is `7`.
   i. For the `backup_cfg_server_host` parameter, enter the name of the host where the backup (warm standby) Configuration Server will be running or a comment `#` if not.
   j. For the `backup_cfg_server_port` parameter, enter either:
      — The port number specified for the backup Configuration Server `Application` object in Configuration Manager
      — `0` (zero) if not to be used
   k. For the `clientside_port_for_cfg_server` parameter, enter either:
      — The desired client-side port number of the connection to the Configuration Server application
      — `0` (zero) if not to be used.

   Note: These arguments are case-sensitive.

3. Click `OK`, which returns you to the `Custom Server (d2is)` dialog box.
4. Select `File > Save`.

End of procedure

Configuring Multiple IVR Drivers for WVR for AIX

If you want to run multiple applications on a single WVR for AIX IVR and you want these applications to communicate with two or more IVR Servers, you can configure multiple IVR Drivers. The following sections describe how to create, compile, configure, and use multiple IVR Drivers.

Note: This is not the recommended approach for Load Sharing. To achieve redundancy or Load Sharing, configure a single IVR Driver for use with multiple IVR Servers that have been configured for Load Sharing. For more information about configuring IVR Servers for Load Sharing, see "IVR Server Redundancy Methods" on page 15.

Procedure: Creating multiple IVR Drivers

Start of procedure

1. Install and configure the IVR Driver, using the instructions in "Installing the IVR Driver" on page 26, and in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*. Verify that the IVR Driver and IVR Server are both functioning correctly (you can use the CTP—see the Appendix on page 67).
2. In the WVR for AIX `Welcome` window, select `Applications > Custom Servers`. The `Custom Servers` dialog box appears.
3. Select the `d2is` custom server, and then select `Copy` from the `Server` drop-down list.
4. In the `Copy Target` box, enter a name for the copy of your IVR Driver (`d2is2` in this example), and then click `OK` to create a copy of the custom server (the original IVR Driver) within the WVR for AIX environment.
5. Manually copy the custom server files from the existing IVR Driver to the folder for the new custom server (the copy of the IVR Driver):
   a. Locate the directory for the new custom server.
   b. In the `Custom Servers` dialog box, select the name of your new IVR Driver.
   c. Select `Server > Open`.
   d. Select `Utilities > Import`. The `Import` dialog box appears.
   e. Enter the following command to copy the files from the original IVR Driver:

      ```
      cp ../d2is_dir/* .
      ```

End of procedure

Upgrading the I-Library

Starting with release 8.1, it is now possible to replace just the I-Library portion of your driver with a new I-Library file. If you are so instructed by your support center, they will help you obtain a newer I-Library file (`libilib.so`).

Procedure: Implementing the I-Library file

Purpose: To put the new I-Library file into use.

Start of procedure

1. Stop the driver process (the default name is `d2is`).
2. Copy the `libilib.so` file into the directory where you previously placed the driver's I-Library called `libilib.so`.

   Note: You may wish to save the original one with a different name first

3. Restart the driver.
4. Note the newer I-Library version number provided in the log.

End of procedure

Procedure: Compiling multiple IVR Drivers

Purpose: To configure each new IVR Driver (WVR for AIX custom server) that you create.

Start of procedure

1. In the `Custom Servers` dialog box, select `Utilities > Build` to change the new custom server to the `built` state.
2. Select `Utilities > Import`. The `Import` dialog box appears.
3. In the `Import` dialog box, enter the command to copy the original IVR Driver executable file into the directory that contains your new copy of the IVR Driver—for example:

   ```
   cp ../d2is_dir/d2is d2is2
   ```

4. Select `Utilities > Install` to install the new IVR Driver as a WVR for AIX custom server.

5. At the command line, enter the following command to verify that the new IVR Driver was successfully installed:

```
ls -l $CUR_DIR/ca/bin
```

End of procedure

Configuring Multiple IVR Drivers

Configure the new IVR Driver, using the instructions in "Configuring the IVR Driver" on page 21. Keep in mind the following configuration considerations:

- Configure the new driver to communicate with a different IVR Server than the original driver.
- If both IVR Servers are defined and configured in the same Configuration Server database, you must use different names for your original IVR Driver and the copied IVR Driver.
- If you want to use this new IVR Driver on multiple IVRs, see the IBM WVR for AIX manuals for information about how to export and import a custom server.

Using Multiple IVR Drivers

Start and stop the new IVR Driver by using the instructions in "Starting and Stopping the IVR Driver" on page 35. Keep in mind the following usage notes for multiple IVR Drivers:

- Channels (IVR ports) are not assigned to a particular driver. Any channel can be used with any driver.
- Ensure that your IVR applications use the correct driver and that their corresponding IVR Servers and Configuration Servers are appropriately configured.
- The Genesys CTP will test the driver only if it is called `d2is` (the default driver name).

Testing the Installation and Configuration

As part of the standard installation, Genesys has provided a Customer Test Package (CTP) to help test the installation and configuration of the IVR Driver. You should configure the IVR Driver before using the CTP. For information about how to use the CTP, see the Appendix on page 67.

Genesys

Chapter

4 Starting and Stopping the IVR Driver

This chapter describes how to start and stop the IVR Driver, which you can do only after you have properly installed and configured both the IVR Driver and the IVR Server. For more information about installing and configuring the IVR Server, see the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

This chapter contains the following sections:

- Prestart Information, page 35
- Starting the IVR Driver, page 35
- Stopping the IVR Driver, page 36

Prestart Information

After installing and configuring the IVR Server and the IVR Driver, you can start the vendor-provided IVR application. Genesys recommends that you start the IVR Server before the IVR Driver.

Starting the IVR Driver

The IVR Driver for WVR for AIX can be started manually, or it can be started automatically by using the automatic start/stop option.

Procedure: Starting the IVR Driver manually

Purpose: To manually start the IVR Driver for WVR for AIX (`d2is`).

Start of procedure

1. In the WVR for AIX `Welcome` window, select `Operations > Custom Server Manager`. The `Custom Server Manager` dialog box appears. The `Custom Server Name` should be `d2is`.
2. Select `Run Status > Start`. The `Run Status` value changes to `Active`.

End of procedure

Procedure:
Automatically starting the IVR Driver

Purpose: To have the IVR Driver automatically start each time WVR for AIX is stopped and restarted.

Start of procedure

1. In the WVR for AIX `Welcome` window, select `Operations > Custom Server Manager`. The `Custom Server Manager` dialog box appears. The `Custom Server Name` should be `d2is`.
2. In the `IPL Status` column, click `Installed` and select `Auto-Start On`. The `Run Status` value changes to `Autoexec`.

End of procedure

Stopping the IVR Driver

The IVR Driver for WVR for AIX can be stopped manually, or it can be stopped automatically by using the automatic start/stop option.

Procedure:
Manually stopping the IVR Driver

Purpose: To manually stop the IVR Driver for WVR for AIX (`d2is`).

Start of procedure

1. In the WVR for AIX `Welcome` window, select `Operations > Custom Server Manager`. The `Custom Server Manager` dialog box appears. The `Custom Server Name` should be `d2is`.
2. In the `Run Status` column, click `Active` and select `Stop`. The `Run Status` value changes to `None`.

3. Repeat steps 1 and 2 for each instance of the IVR Driver that you want to stop.

End of procedure

Automatically Stopping the IVR Driver

When the `Run Status` value is `Autoexec` (as described in "Automatically starting the IVR Driver" on page 36), the IVR Driver is automatically stopped when WVR for AIX is stopped, and it is restarted when WVR for AIX is restarted.

Chapter

5 Functions

After you install, configure, and start the IVR Driver, IVR Interface Option 8.1 is ready to be used and tested as a user function within the vendor-provided Interactive Voice Response (IVR) application. This chapter describes the functions supported in IVR Interface Option 8.1. It contains the following sections:

- Input Constraints, page 39
- Genesys-Provided Functions, page 40

Input Constraints

Some of the functions described in this chapter use key-value pairs. The following constraints apply:

- Characters with a value less than `0x20` are not valid in key names or data values. The only exceptions are the characters `0x09`, `0x0A`, and `0x0D`, which correspond to the ASCII control characters `TAB`, `LINE FEED`, and `CARRIAGE RETURN`. No other ASCII control characters are allowed.
- Although you can use a colon (:) when defining the values for a key, a key that includes a colon can be used to perform only the following operations:
  - `UDataAddKVP`
  - `UDataAddList`
  - `UDataDelKVP`
- You cannot issue a `UDataGetKVP` function call from the IVR Driver by using a key that contains a colon (although other Genesys software might be able to access this type of key-value pair).
- The length of the key can be no more than 1023 characters. The length of any individual data string is limited to 2047 characters. The combined key-value pairs (including delimiters) on a given call instance can total no more than 3399 bytes.

Genesys-Provided Functions

The Genesys IVR Driver for WVR for AIX (`d2is`) provides functions that can be called within the State Table of DirectTalk. For specific instructions on how creating voice applications using Genesys-provided functions, refer to the following documents:

- *IBM WebSphere Voice Response for AIX with DirectTalk Technology: Designing and Managing State Table Applications*
- *IBM WebSphere Voice Response for AIX with DirectTalk Technology: Application Development using State Tables*

All input parameters specify parameters in the `SendData` State Table action, and all output parameters specify parameters in the `ReceiveData` State Table action. The types of parameters (`String` or `Number`) are given using DirectTalk terminology.

If a function returns more than one variable, you must check the `Result` variable before using any other variable. You can use other variables only if the `Result` is successful.

The Genesys IVR Driver is part of the custom server category, as described in the *DirectTalk for AIX Custom Servers* document. IVR administrators should refer to this document for information about the interrelationship between the state tables and custom server.

Table 1 lists the Genesys-provided functions.

Table 1: Functions

Function	Summary	Page
Notify Functions		
NotifyCallStart	Notifies the IVR Server that the call has started.	Page 42
NotifyCallEnd	Notifies the IVR Server that the call has ended.	Page 43
Telephone Functions		
CallInit	Initiates a new call to the destination DN.	Page 44
CallTransfer	Makes a transfer to the DN (mute transfer).	Page 44
CallTransferKVList	Attaches (updates) a list of key-value pairs to a call before making a transfer.	Page 45
CallComplete	Completes the current call.	Page 46
CallConsultInit	Initiates a consulting call with a DN.	Page 46
CallConsultRetrieve	Retrieves the original consulting call.	Page 47

Table 1: Functions (Continued)

Function	Summary	Page
CallConsultTransfer	Completes the consulting call by making a transfer.	Page 48
CallConference	Connects the original two parties of a conversation with a third party.	Page 49
CallConsultConference	Completes the consulting call by creating a conference.	Page 49
User Data Manipulation Functions		
UDataAddKVP	Attaches (updates) a key-value pair to a call.	Page 50
UDataGetKVP	Requests an attached key value, which is associated with a key-value pair.	Page 51
UDataDelKVP	Deletes a key-value pair from the call database.	Page 51
UDataAddList	Attaches (updates) a list of key-value pairs to a call.	Page 52
UDataDelAll	Deletes all user data from the call database.	Page 52
Call Information Functions		
GetCallInfo	Returns the requested information from the call database.	Page 53
Call Data Transfer Functions		
CDT_Init	Requests an access number in order to make a Call Data Transfer to a remote destination.	Page 54
CDT_Cancel	Cancels the `CDT_Init` request.	Page 56
Route Functions		
RouteRequest	Requests a service point from the router.	Page 57
RouteStart	Requests the start of a new session from the router (assigned to the service point).	Page 58
RouteAbort	Aborts the previously requested service session from the router.	Page 59
GetRequest	Requests the next action from the router.	Page 59
SendReply	Sends a reply from the IVR script to the router.	Page 61
General-Purpose Functions		
GetReply	Retrieves the reply for a request that was sent earlier.	Page 62

Table 1: Functions (Continued)

Function	Summary	Page
GetErrorCode	Returns additional information about a previous function call that returned an error.	Page 62
GetVersion	Returns the version of the IVR Driver, IVR Server, or IVR Library.	Page 63
ToLog	Places a specified text string into a log file.	Page 64
Statistical Functions		
StatPeek	Sends a request to Stat Server to provide information about a predefined statistic.	Page 64

The remainder of this chapter describes the function calls that the IVR Driver for WVR for AIX supports.

Notify Functions

NotifyCallStart

This function notifies the IVR Server that the IVR channel has answered the call.

Your script must call the `NotifyCallStart` function as the first Genesys function at the start of each call instance—typically after the script's `Answer` function. After a successful `NotifyCallStart`, the next call must be `GetCallinfo(EventName)`. This call must return a valid Genesys event (`EventEstablished`, `EventAttachedDataChanged`, or `EventPartyChanged`) before the script can use *any* other function calls (except `NotifyCallEnd`). The script must call `NotifyCallEnd` as its last function. The script must call only one `NotifyCallStart` and one `NotifyCallEnd` function for each call.

To ensure that your application has received an `EventEstablished`, `EventAttachedDataChanged`, or `EventPartyChanged` event:

1. Issue the `NotifyCallStart` call.
2. Issue a `GetCallInfo` call on the line, asking for the `EventName`.
3. Verify that your application receives an `EventEstablished`, `EventAttachedDataChanged`, or `EventPartyChanged` event.

The IVR Driver will be ready to perform additional API requests only after the `EventEstablished`, `EventAttachedDataChanged`, or `EventPartyChanged` event has been received.

Input

`String CallID`

The Call ID (a unique identifier assigned to each call by the switch). The `CallID` should be empty when the call is issued.

`String ANI`

The ANI.

`String DNIS`

The DNIS.

`String CDT_Tag`

A tag that can be used to provide the Call Data Transfer function. The `CDT_Tag` must be empty when the call is issued.

The `String ANI` and `String DNIS` parameters can be included (if available) when the PBX is not present at the site, and IVR Server is operating in IVR-In-Front mode. When the PBX is present at the site, these strings must be empty.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

NotifyCallEnd

This function notifies the IVR Server that the IVR channel has disconnected the call.

This function must be called as the last Genesys-provided function in the script. Therefore, the `NotifyCallEnd` function must be invoked before the `TerminateCall` action of the State Table. No Genesys function call other than a `GetCallInfo(LastEvent)` is expected for the call session after this function.

Input

None

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

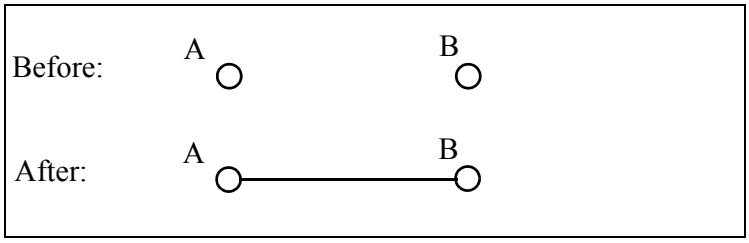
Note: The `NotifyCallStart` function must be used at the start of each script, typically after the `Answer` function. The `NotifyCallEnd` function must be called as the last function in the script.

Telephone Functions

CallInit

This function initiates a new call to the destination DN (B) (see the scenario). The `CallInit` function works with the switch, which is monitored by T-Server. This function is used only for the IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

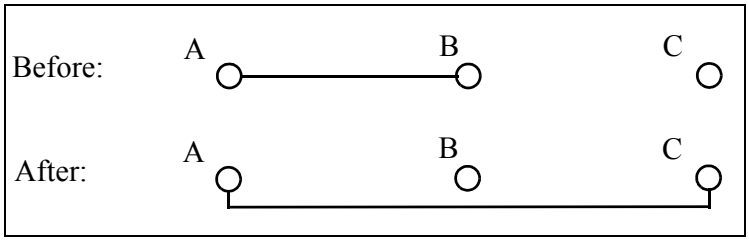
If `1`, the request succeeded.

If `0`, the request failed.

CallTransfer

This function produces a direct transfer without an intervening conference call. The call moves from an extension (B), where a transfer is initiated, to a new extension (C) specified in the destination DN (see the scenario). In other words, the party (A) who initiates the call is disconnected from the original DN (B) and reconnected to the destination DN (C). This function is only used for the IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

CallTransferKVList

This function adds the list of key-value pairs to the current call before making the transfer.

Input

`String DstDN`

The phone number of the new destination party.

`String KVList`

A list of key-value pairs, in the format:

`<delimiter>Key<delimiter>Value`

Note: The length of the `Key` string can be no more than 1023 characters. The combined key-value pairs (including delimiters) in the `KVList` string can total no more than 3399 bytes.

If the `UU_DATA` key is used, T-Server will handle the data part of the key-value pair as `UUIData`.

Example 1 `*Customer ID*id01*UU_DATA*uui info*CustomerName*Jim Doe`

In this example, `*` is the delimiter.

Example 2 `#Customer ID#id02#UU_DATA#uui info#Customer2 ID#1234`

In this example, `#` is the delimiter.

Output

`Number Result`

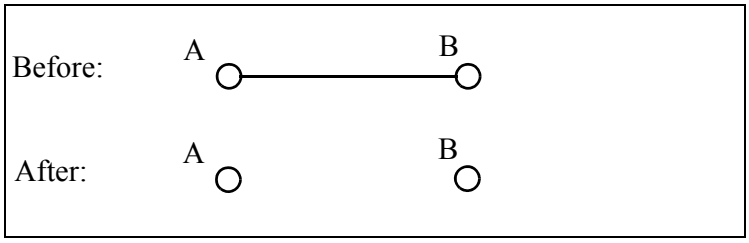
If `1`, the request succeeded.

If `0`, the request failed.

CallComplete

This function ends the current call (see the scenario). This function is used only for the IVR-Behind-Switch configuration.

Scenario



Input

None

Output

`Number Result`

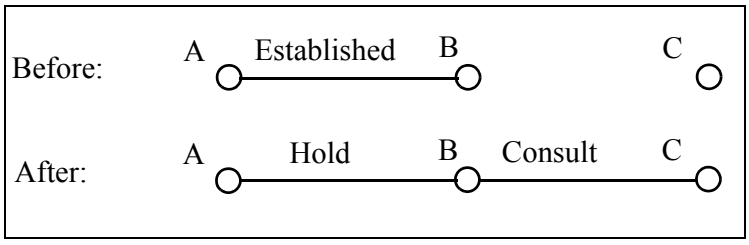
If `1`, the request succeeded.

If `0`, the request failed.

CallConsultInit

This function initiates a consulting call. The original party (A) is placed on hold for the duration of the consulting call. The party (B) who requests the `CallConsultInit` function is involved in a new consulting call with a third party (C) (see the scenario). This function is used only for the IVR-Behind-Switch configuration.

Scenario



Note: After this function is called, use one of the following functions to complete the operation:

- "CallConsultRetrieve" on page 47
- "CallConsultTransfer" on page 48
- "CallConsultConference" on page 49

Input

`String DstDN`

This phone number describes the new destination party.

Output

`Number Result`

If `1`, the request succeeded.

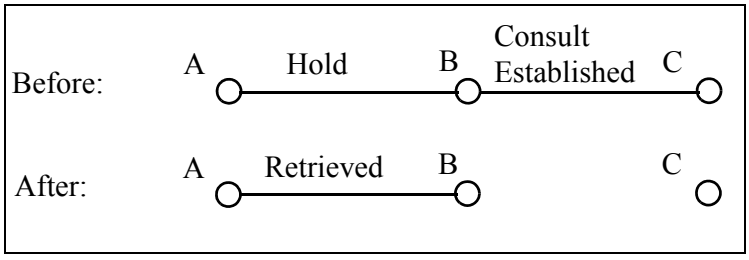
If `0`, the request failed.

CallConsultRetrieve

This function completes a consulting call. For the duration of the consulting call, the original party (A) is place on hold. The second party (B) then creates a consulting call to the third party (C) (see the scenario). After the `CallConsultRetrieve` function, the third party (C) is released, and the call between the first party (A) and second party (B) is restored. This function is used only for the IVR-Behind-Switch configuration.

Note: Use this function only after `CallConsultInit` has been successfully completed. For more information, see "CallConsultInit" on page 46.

Scenario



Input

None

Output

`Number Result`

If `1`, the request succeeded.

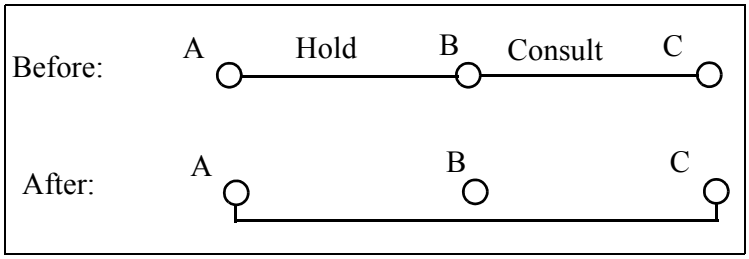
If `0`, the request failed.

CallConsultTransfer

This function completes a transfer. During the transfer, the caller (A) is placed on hold. The original recipient (B) makes a call to a new party (C) (see the scenario). A consulting call is established between the original recipient (B) and the new party (C). The `CallConsultTransfer` function initiates the completion of the transfer. When the transfer is complete, the original recipient (B), who initiated the transfer, is dropped. The ongoing call involves only two participants: the original caller (A) and the new party (C). This function is used only for the IVR-Behind-Switch configuration.

Note: Use this function only after `CallConsultInit` has been successfully completed. For more information, see "CallConsultInit" on page 46.

Scenario



Input

None

Output

`Number Result`

If `1`, the request succeeded.

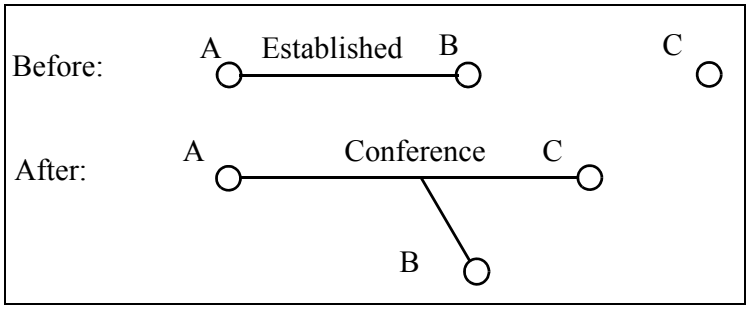
If `0`, the request failed.

CallConference

This function makes a conference call to a new DN, by connecting the two parties (A and B) of the original conversation with an additional party (C) (see the scenario). As a result, three or more people can participate in the same

phone conversation, talking from three or more extensions. This function is used only for the IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

If `1`, the request succeeded.

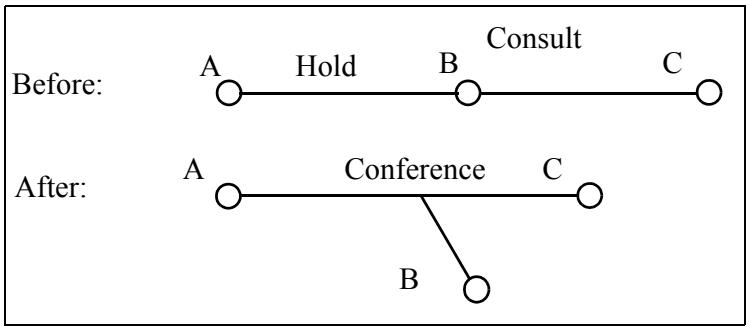
If `0`, the request failed.

CallConsultConference

This function completes a consulting call, by connecting the two parties (A and B) of the original conversation with an additional party (C) from the consulting call. See the scenario below. As a result, several people can participate in one phone conversation at the same time, talking from several extensions. This function is used only for the IVR-Behind-Switch configuration.

Note: Use this function only after `CallConsultInit` has been successfully completed. For more information, see “CallConsultInit” on page 46.

Scenario



Input

None

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

User Data Manipulation Functions

UDataAddKVP

This function attaches User Data (a value) and a corresponding name (key) to the current call. T-Server (or the IVR Server in IVR-In-Front mode) uses this key-value pair to track data for call that is handled at the contact center. If the key of the key-value pair (which is created when new user data is attached) duplicates a key that already exists, the new pair replaces the existing pair.

Note: Both the user key and user value must be present—for example:

- `Key = Customer ID`
- `Value = 1672`

Input

`String Key`

The key for the user data.

`String UserValue`

Any customer data or other relevant data.

Note: The length of the `Key` string can be no more than 1023 characters. The length of the `UserValue` data string is limited to 2047 characters. The combined key-value pairs (including delimiters) on a given call instance can total no more than 3399 bytes.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

UDataGetKVP

This function returns the user data associated with a key-value pair that was previously attached to the call. If a key is given that is not in the current call information, the string `NoMatch` is returned by default.

Input

`String Key`

The key for the requested user data.

Note: The length of the `Key` string can be no more than 1023 characters.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String UserValue`

A value for the specified key.

Note: The length of the `UserValue` data string can be no more than 2047 characters.

UDataDelKVP

This function deletes one key-value pair specified by the key from the current call.

Input

`String Key`

The key string for deleting the pair.

Note: The length of the string `Key` can be no more than 1023 characters.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

UDataAddList

This function adds a list of key-value pairs to the current call.

Input

`String KVList`

A list of key-value pairs in the format:

`<delimiter>Key<delimiter>Value`

Note: The length of the `Key` string can be no more than 1023 characters. The combined key-value pairs (including delimiters) in the `KVList` string can total no more than 3399 bytes.

Example 1 `*Customer ID*4942*CustomerName*Jim Doe`

In this example, `*` is the delimiter.

Example 2 `#Customer ID#4943#Customer2 ID#1234`

In this example, `#` is the delimiter.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

UDataDelAll

This function deletes all user data from the current call.

Input

None

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

Call Information Functions

GetCallInfo

This function requests information related to an active call. If a particular value is not available (for example, `ANI`), the string `NULL` is returned by default.

Input

`String TypeInfo`

The type of requested information. Table 2 lists the possible `TypeInfo` values.

Table 2: TypeInfo Arguments

Value	Requested Information
All[a]	All call information available from the IVR Server, as described in the rest of this table
ANI	ANI information
CallID	PBX call ID
ConnID	T-Server connection ID
DN	Current port assigned to the IVR channel
DNIS	DNIS information
EventName	Name of the last event received on the current IVR channel
FirstHomeLocation	First Home Location of the call
OtherDN	Calling DN (if any)
OtherTrunk	Calling Trunk (if any)
OtherQueue	Calling Queue (if any)
ThisDN	Current (called) DN assigned to the IVR channel
ThisTrunk	Current (called) Trunk assigned to the IVR channel

Table 2: TypeInfo Arguments (Continued)

Value	Requested Information
ThisQueue	Called Queue (if any)
UUID	Call UUID (if any)

a. Starting with release 8.1: If `callinfo_all_returns_uuid = true` is present in the `ivr_server_interface` section of the IVR Driver configuration, then the current value of `AttributeCallUUID` is also returned for a request for `All`. If it is not present, or set to `false`, then the content returned for `All` is the same as in the prior release.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String Info`

Contains the requested information.

Call Data Transfer Functions

CDT_Init

This function requests an access number to make a Call Data Transfer to a remote destination.

This access number can be used as a DN to make a transfer either through the IVR itself or by using the `CallTransfer` function of `d2is`.

Input

`String DN`

The destination DN for the requested Call Data Transfer.

`String Location`

The name of the destination Call Data Transfer server.

`String CDT_Type`

The type of Call Data Transfer protocol. Table 3 on page 55 lists the possible Call Data Transfer types.

Table 3: Call Data Transfer Types

Value	Description
Default	Uses the type already configured in your multi-site routing environment.
Indirect	`CDT_Type` is changed to `Route`.
DirectNT	`CDT_Type` is changed to `DirectNotoken`.
DirectTO	`CDT_Type` is changed to `Direct`.
DirectTI	`CDT_Type` is changed to `Direct`.
ReRoute	Passes unchanged to the IVR Server.

Note: If your IVR application passes in a string that is not equal to one of these Call Data Transfer types, the string is passed on to the IVR Server unchanged.

The IVR Server supports the following types:

```
Default|Route|Reroute|Direct|DirectAni|DirectNotoken|DirectAniDnis|
DirectUUI|DirectDigits|DnisPool
```

`String CDT_Tag`

The Call Data Transfer tag.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String AccessDN`

The access DN that should be used for the requested external transfer.

`String CDT_Tag`

The Call Data Transfer tag (as the input argument, or as generated by the Call Data Transfer Layer).

`String RefID`

The Reference ID that can be used to cancel a transfer using the `CDT_Cancel` function.

CDT_Cancel

This function can be used only after the `CDT_Init` function has been called. This function cancels the preceding request by the `CDT_Init` function for an access number.

Input

`String RefID`

The Reference ID taken from the previous `CDT_Init` request.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

Route Functions

When you use the IVR Driver routing functions in your applications, they must be coded in a certain order. The pseudocode example in Figures 7 and 8 shows the correct order in which the routing APIs must be used. You will need to convert this example into valid code for your applications.

```
/*
 Port             = type ilPORT with the port of IVR channel
                         (see interface.h)
 ilRQRouteStart   = type ilRQ - returned Request ID for Route Start
                         (see interface.h)
 psRouteStartRep  = type PSTR - pointer to buffer - preallocated
                         for Get Reply to return result in
 iRepLen          = type int - length of above buffer
 ilGetReplyRet    = type ilRet - returned from Get Reply indicating
                          result of request (see interface.h)
 ilRQGetRequest   = type ilRQ - returned Request ID for Get Request -
                         used in Send Reply (see interface.h)
 ilSendReplyRet   = type ilRet - returned from Send Reply
                           indicating result of request (see interface.h)
 bResult          = type BOOL - set true if treatment was success -
                         else false (see interface.h)
 psReply          = type CPSTR - pointer to buffer with result of
                      treatment - if return is required
      (see interface.h)

  ilRQ_ANY - generate the request id (see interface.h)

 nlrepeat = a number if type int to indicate strategy still active
*/
```

Figure 7: Comments About the Pseudocode

```
ilRQRouteStart = ilSRqRouteStart(ilRQ_ANY,Port,"7000");
                 /* route sequence start on route dn */
if (ilRQRouteStart > 0)
                /* make sure Route Start worked */
{
    nlrepeat = 1;
    while(nlrepeat == 1)
    {
        ilGetReplyRet = ilGetReply(ilRQRouteStart,psRouteStartRep,iRepLen);
                /* check reply for Route Start */
                /* timeout means that the routing strategy */
        if(ilGetReplyRet == ilRET_TIMEOUT)
                /*  is still active */
        {
            ilRQGetRequest = ilGetRequest(Port,psRep,iRepLen);
                /* retrieve the next treatment - if one exists */
                /* treatment details in the return buffer      */

            if(ilRQGetRequest>0)
                /* make sure Get Request worked before        */
            {
             /* processing the returned treatment          */
                /* processing by application to apply treatment */
               ilSendReplyRet = ilSendReply(ilRQGetRequest,bResult,psReply);
                     /* send treatment result to URS */
            }
        }
        else
           nlrepeat = 0;
                /* the get reply returned other than timeout */
    }
    /* this implies the strategy has ended - and */
    /* a RouteResponse has been processed by the library */
    /* the route destination - if it exists in the */
    /* RouteResponse will be returned in psRouteStartRep */
}
else /* RouteStart failed */
```

Figure 8: Pseudocode for Routing Functions

RouteRequest

This function sends a request to the router, and the IVR waits for routing instructions. The router returns the destination DN of the next service point. This function is used for both IVR-In-Front and IVR-Behind-Switch configurations.

Input

`String Service`

The service name requested from Interaction Router.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String NextService`

The name of the next service point/destination DN provided by Universal Routing Server (URS).

Notes:
- URS returns the destination of the next service point, and the IVR Server immediately responds to URS with a `RouteDone` event. If the IVR is unable to route the call to the destination (or service point) that URS has requested, there is no way to indicate that the new route has failed. If this error condition could occur, you must use the `RouteStart` API instead of `RouteRequest`.
- For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 67.

RouteStart

This function requests the start of a new session from the router (assigned to the service port).

This function begins the information interchange between the IVR channel and the router. During this session, the router can send IVR commands (*treatments*), which the IVR must execute. The IVR uses the `GetRequest` function to receive treatments, and the `SendReply` function to inform the router that treatments have been executed. When the session ends, the router has the option of sending the name of the next service (usually a phone number for the transfer call) to the IVR. The IVR can also end the session by using the `RouteAbort` function.

This function returns no router response. To receive a response from the router, the IVR script must call either the `GetReply` or `GetRequest` function. The `GetReply` function can be used only when the router strategy sends the next service (DN for transfer only), not when it sends the treatment command. If the router strategy uses treatments, the `GetRequest` function must be used in the IVR script.

Note: For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 67.

Input

`String Route Point`

The name associated with a service that was requested from the router. The router must be registered for monitoring this resource.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`Number Request ID`

A unique ID that can be used later in the `RouteAbort` function.

RouteAbort

This function aborts the previously requested service session from Universal Routing Server. When URS receives a `RouteAbort` request, it starts the `transfer_time` timer. If it receives no other events or routing operations before the timer expires, it exits the current routing strategy.

Note: You must wait until the timer expires before issuing any other routing requests.

Input

`Number Request ID`

A unique ID assigned by the `RouteStart` function.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

GetRequest

This function takes the request for the next action from the IVR Server. If this function is used in the router session, it returns the IVR treatment or the name of the next service (usually a number for transfer).

Note: For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 67.

Commands coming from the Genesys router are represented by strings with the following (ordered list) format:
`":","CommandType",":","Data",...`

This format begins with a delimiter that is also used later in the string to divide fields. The number of actual parameters coming with a command depends on the particular command type. The command string contains at least a delimiter and a `CommandType`. Table 4 lists the set of Genesys routing commands and their equivalent T-Server treatment types.

Table 4: Routing Commands

Routing Command	T-Server Treatment
IVR	TreatmentIVR
Music	TreatmentMusic
RingBack	TreatmentRingBack
Silence	TreatmentSilence
Busy	TreatmentBusy
CollectDigits	TreatmentCollectDigits
PlayAnnounce	TreatmentPlayAnnouncement
PlayAnnouncementAndDigits	TreatmentPlayAnnouncementAndDigits
VerifyDigits	TreatmentVerifyDigits
RecordAnnouncement	TreatmentRecordUserAnnouncement
DeleteAnnounce	TreatmentDeleteUserAnnouncement
CancelCall	TreatmentCancelCall
PlayApplication	TreatmentPlayApplication
SetDefaultRoute	TreatmentSetDefaultRoute
TextToSpeech	TreatmentTextToSpeech
TextToSpeechAndDigits	TreatmentTextToSpeechAndDigits
FastBusy	TreatmentFastBusy
RAN	TreatmentRAN

For a detailed description of treatments and parameters, see the *Universal Routing 8.1 Reference Manual*. See your vendor-provided IVR documentation for a description of whether and how the IVR supports these treatments.

Input

None

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String RouteRequestID`

The request ID number assigned to the current router request. This request ID can be for either a treatment or a route. (A route is sometimes referred to as the next service.)

If `RouteRequestID = 0`, a route has been returned in the data string, and your route sequence has been completed.

If `RouteRequestID > 0`, a treatment has been returned in the data string. Save the `RouteRequestID`, and send it back to the router when you issue the next `SendReply` function. Then, after the `SendReply`, issue another `GetRequest` to see whether another treatment (or a route) follows.

`String Data`

A request (either a treatment or the next service from the router), as determined by the `RouteRequestID` string.

SendReply

This function sends a reply from the IVR script to IVR Server. Within the router session, this function sends the result of the treatment execution.

Note: For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 67.

Input

`String RouteRequestID`

The request ID received from the router by using the `GetRequest` function.

`String sResult`

The execution result of the request (treatment). `OK` indicates successful execution of the request.

`String Data`

The response from the IVR script. Depends on the treatment type.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

General-Purpose Functions

GetReply

This function retrieves the reply for a previously sent request.

Input

`Number Request ID`

The request ID received from a previously submitted request.

Output

`Number Result`

If `1`, the function succeeded.

If `0`, the function failed.

If `-1`, a timeout occurred, and the `GetReply` function must be called later for the required `Request ID`.

`String Reply`

Request-specific information. For the exact values of each `Reply` string, see the "Output" section in the description of each function.

Note: If the value of configuration option `getreply_with_location` is set to `true` in the related driver application, then the `getreply` string returned for a route response will include the name of the premise T-Server.

GetErrorCode

This function requests additional information about a previously requested function call that was not successful and that returned an error. `GetErrorCode` must be submitted immediately after the function call that failed.

Input

None

Output

`Number Result`

Table 5 lists the possible `Result` values.

Table 5: Result Values

Number	Result	Description
0	`NOERROR`	The previously requested function call did not generate an error.
-5	`CONNCLOSED`	The connection to the IVR Server is closed.
-7	`BADARGS`	One or more arguments set for the previously requested function call are not valid.
-9	`UNSUPPORTED`	The previously requested function call is not supported.
-11	`TIMEOUT`	The previously requested function call timed out, but at the time when it was called, it was still in progress at the server.
-12	`REQEXPIRED`	The previously requested function call has expired. Discontinue attempts to complete that function.
-1000	`REQFAILURE`	The previously requested function call has failed at the server.
-2000	`CANTFINDCHANNEL`	There has not been a previous request on this IVR channel. The channel cannot be found.

GetVersion

This function prints the string to either the IVR Driver log or the log of IVR Server.

Input

`String Service`

The name of the service for which the version number is requested.

If this argument is null or a single space, the IVR Library version is returned.

If this argument is the name of the IVR, the IVR Driver version is returned.

If this argument is anything else, the IVR Server version is returned.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String Version`

The version of the Genesys product with the name `Service`.

ToLog

This function prints the string either to the `d2is` log or the log of another Genesys product.

Input

`String PrintString`

The string that is printed to the log.

`String Service`

If this argument is the name of the IVR, the function prints the string to the IVR Server log.

If this argument is any other value, or if it is empty, the function prints the string to the IVR Driver log.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

Statistical Functions

StatPeek

This function requests that Stat Server provide information about a predefined statistic. The parameters of the statistic are defined in the configuration environment.

Note: The supported statistics (which must be configured in the IVR Server) are `CurrNumberWaitingCalls` and `ExpectedWaitTime`. For more information, see the section about IVR Server options in the *IVR Interface Option 8.1 IVR Server System Administrator's Guide*.

Input

`String StatName`

The name of the requested statistic, which must be defined in the configuration environment using Configuration Manager.

Output

`Number Result`

If `1`, the request succeeded.

If `0`, the request failed.

`String StatInfo`

The data value of the requested statistic.

Genesys

Appendix

Customer Test Package

This appendix provides information about the Customer Test Package (CTP). It contains the following sections:

- Introduction, page 67
- Configuring and Using the Customer Test Package, page 67
- CTP Voice Menu, page 69

Introduction

The IVR Driver includes a CTP that contains an Interactive Voice Response (IVR) application and a corresponding voice prompt database. The IVR application is a complete application that you can install on a vendor-provided IVR, and use to test and better understand the integration with Genesys Framework 8.1.

Configuring and Using the Customer Test Package

Procedure:
Configuring and running the CTP

Prerequisites

Installation of the CTP.

Start of procedure

1. Configure the CTP application to run over one or more IVR channels, by associating channels with a CTP application profile:
   a. In the WVR for AIX `Welcome` window, select `Configuration > Pack Configuration > Change > Channel IDs`. The `Channel Identification` dialog box appears.
   b. Enter values in the `Area Code` and `Telephone Number` boxes.
   c. Click `OK`.
   d. Select the `Pack 1` check box.
   e. Select `File > Save`, and then `File > Close`.
2. Create an application profile, which associates the specified area code and phone number with the CTP application:
   a. In the WVR for AIX `Welcome` window, select `Configuration > Application Profiles`. The `Application Profiles` dialog box appears.
   b. Select `File > New`. The `Application Profile (unnamed)` dialog box appears.
   c. In the `Name` box, enter `CTP`.
   d. Click `State Table` and select `d2is_ctp`.
   e. Click `Ok`, and then select `File > Save`. The `Enter Data` dialog box appears.
   f. Enter the same concatenated string of area code and telephone number that you entered in Step 1, and then click `Ok`. You are returned to the `Application Profile (unnamed)` dialog box, which is now renamed `Application Profile (<your concatenated string>)`.
   g. Select `File > Close`. You are returned to the `Application Profiles` dialog box.
   h. Select `File > Close`.
3. Verify that the IVR is active:
   a. In the WVR for AIX `Welcome` window, select `Operations > System Monitor`. The `System Monitor` dialog box appears.
   b. Verify that the `Trunk Status` value is `In Service`. If it is not, change it to `In Service` by selecting `Enable` and then clicking `Ok`.
   c. Click `In Service` and select `Channels In Service`. The IVR is now active and ready to take calls.
4. If you are using the IVR-Behind-Switch configuration, start the T-Server.
5. Start the `TServer_IVR` application.
6. Start the IVR Driver.
7. Using a telephone set, dial the Directory Number (DN) associated with the IVR channel to activate the CTP and test the integration of IVR Interface Option 8.1 with the Genesys Framework software.

8. Follow the voice menu prompts to run the CTP. For more information, see "CTP Voice Menu."
9. To learn how to invoke the Genesys IVR Interface Option 8.1 functions, follow the script example within the application.
10. To learn how to code the IVR Driver function calls, review the CTP State Tables in the WVR for AIX development environment.

End of procedure

CTP Voice Menu

Table 6 lists the voice menu options available in the CTP.

Table 6: CTP Voice Menu Options

Option	Key
Main Menu	
Open the User Data Menu	Press 1
Open the Information Menu	Press 2
Open the Transfer Menu	Press 3
Open the Call Data Transfer Menu	Press 4
Open the Router Instruction Menu	Press 5
Open the Statistics Menu	Press 6
Quit	Press 0
User Data Menu	
Attach data	Press 1
Get attached data	Press 2
Delete a key-value pair	Press 3
Remove all attached data	Press 4
Attach a list of user-data pairs	Press 5
Print a string to a log	Press 6
Return to the Main Menu	Press 0

Table 6: CTP Voice Menu Options (Continued)

Option	Key
Information Menu	
Get the Last Event Name	Press 1
Get the PBX Call ID	Press 2
Get the T-Server Connection ID	Press 3
Get the DNIS	Press 4
Get the ANI	Press 5
Go to the Called call information	Press 6
Go to the Calling call information	Press 7
Get the Version of the IVR Driver	Press 8
Get the UUID **Note:** Not vocalized in example CTP, but functionally present.	Press 9
Return to the Main Menu	Press 0
Called Information Menu	
Get the called DN	Press 1
Get the called ACD queue	Press 2
Get the called Trunk	Press 3
Return to the Information Menu	Press 0
Calling Information Menu	
Get the calling DN	Press 1
Get the calling ACD queue	Press 2
Get the calling Trunk	Press 3
Return to the Information Menu	Press 0
Transfer Menu	
Transfer using the IVR	Press 1
Transfer using T-Server	Press 2
Initiate a transfer	Press 3

Table 6: CTP Voice Menu Options (Continued)

Option	Key
Complete a transfer	Press 4
Retrieve an original call	Press 5
Initiate a conference	Press 6
Complete a conference	Press 7
Perform a single-step conference	Press 8
Return to the Main Menu	Press 0
Call Data Transfer Menu	
Perform an indirect Call Data Transfer	Press 1
Perform a direct Call Data Transfer with no tag	Press 2
Perform a direct Call Data Transfer with a tag generated by the caller	Press 3
Perform a direct Call Data Transfer with a tag generated by the Call Data Transfer Layer	Press 4
Return to the Main Menu	Press 0
Transfer Methods Menu	
Perform a transfer by IVR	Press 1
Perform a transfer by T-Server	Press 2
Cancel the Call Data Transfer request	Press 0
Router Instruction Menu	
Perform a RouteStart, GetRequest, SendReply sequence	Press 1
Perform a RouteRequest	Press 2
Return to the Main Menu	Press 0
Statistics Menu	
Perform a StatPeek	Press 1
Return to the Main Menu	Press 0

Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

IVR Server

- *IVR Interface Option 8.1 IVR Server System Administrator's Guide*, which describes/provides information about how to install, configure and run the IVR Server component.
- Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at http://genesyslab.com/support.

Genesys

- *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact Genesys Technical Support for more information.

Information about supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys Supported Operating Environment Reference Manual*
- *Genesys Supported Media Interfaces Reference Manual*
- *Genesys Hardware Sizing Guide,* which provides information about Genesys hardware sizing guidelines for the Genesys 8.x releases.

- *Genesys Interoperability Guide,* which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide,* which introduces you to the concepts, terminology, and procedures relevant to the Genesys licensing system.
- *Genesys Database Sizing Estimator 8.x Worksheets,* which provides a range of expected database sizes for various Genesys products.
- *IVR SDK 8.1 C Developer's Guide*
- *IVR SDK 8.1 XML Developer's Guide*
- *Voice Platform SDK 8.x .NET (or Java) API Reference*
- *Framework 8.1 Deployment Guide*
- *Framework 8.1 T-Server Deployment Guide*
- *Framework 8.1 Configuration Options Reference Manual*

For additional system-wide planning tools and information, see the release-specific listings of System Level Documents on the Genesys Technical Support website, accessible from the `system level documents by release` tab in the Knowledge Base `Browse Documents` Section.

Genesys product documentation is available on the:

- Genesys Technical Support website at `http://genesyslab.com/support`.
- Genesys Documentation wiki at `http://docs.genesyslab.com/`.
- Genesys Documentation Library DVD and/or the Developer Documentation CD, which you can order by e-mail from Genesys Order Management at `orderman@genesyslab.com`

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

81fr_ref_06-2012_v8.1.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

Table 7 describes and illustrates the type conventions that are used in this document.

Table 7: Type Styles

Type Style	Used For	Examples
Italic	• Document titles • Emphasis • Definitions of (or first references to) unfamiliar terms • Mathematical variables Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 76).	Please consult the *Genesys Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula, $x + 1 = 7$ where x stands for . . .
Monospace font (Looks like `teletype` or `typewriter text`)	All programming identifiers and GUI elements. This convention includes: • The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. • The values of options. • Logical arguments and command syntax. • Code samples. Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.	Select the `Show variables on screen` check box. In the `Operand` text box, enter your formula. Click `OK` to exit the `Properties` dialog box. T-Server distributes the error messages in `EventError` events. If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls. Enter `exit` on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	`smcp_server -host [/flags]`
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	`smcp_server -host <confighost>`

Index

Symbols

[] (square brackets) . . . 76
< > (angle brackets) . . . 76

A

ACD queue . . . 70
AgentControl section
  DriverIgnoreReady parameter . . . 23
  DriverReadyWorkMode parameter . . . 23
  DriverRetryTimeout parameter . . . 23
  LegacyMode parameter . . . 22
AIX version . . . 26
angle brackets . . . 76
ANI . . . 70
Annex tab . . . 22
applications
  IVR . . . 67
  voice . . . 40
architecture . . . 12
attached data . . . 69
audience, for document . . . 8
automatic start/stop . . . 36, 37

B

brackets
  angle . . . 76
  square . . . 76

C

call data transfer functions
  CDT_Cancel . . . 56
  CDT_Init . . . 54
  overview . . . 41
call data transfers
  cancelling . . . 71
  direct . . . 71
  indirect . . . 71
  layer . . . 71
  menu options . . . 71
call flow . . . 12
call information functions
  GetCallInfo . . . 53
  overview . . . 41
called information menu options . . . 70
calling information menu options . . . 70
calls
  information . . . 70
  original . . . 71
  PBX ID . . . 70
cancelling call data transfers . . . 71
commenting on this document . . . 8
compatibility . . . 19
compiling multiple IVR Drivers . . . 33
completing
  conferences . . . 71
  transfers . . . 71
conferences
  completing . . . 71
  initiating . . . 71
  single-step . . . 71
Configuration Server
  host name . . . 31
  port number . . . 31
configuring
  IVR Driver . . . 11
  IVR Server . . . 35
  multiple IVR Drivers . . . 32
  options . . . 21
conventions
  in document . . . 75
  type styles . . . 76
creating multiple IVR Drivers . . . 32
CTP . . . 26, 28, 34, 67
custom server . . . 27, 28, 30, 40

D

d2is file, importing . . . 27
d2is_aix43.tar file . . . 26
database, voice prompt . . . 67
defining
  IVRs . . . 21
  ports . . . 21
deleting key-value pairs . . . 69
direct call data transfers . . . 71
DNIS . . . 70
DNs . . . 68, 70
document
  audience . . . 8
  change history . . . 9
  conventions . . . 75
  errors, commenting on . . . 8
  version number . . . 75

F

files
  d2is . . . 27
  d2is_aix43.tar . . . 26
  install.man . . . 27
font styles
  italic . . . 76
  monospace . . . 76
functions
  call data transfer . . . 41
  call information . . . 41
  general-purpose . . . 41
  Genesys-provided . . . 12
  notify . . . 40
  route . . . 41
  statistical . . . 42
  telephone . . . 40
  user data manipulation . . . 41

G

general-purpose functions
  GetErrorCode . . . 62
  GetReply . . . 62
  GetVersion . . . 63
  overview . . . 41
  ToLog . . . 64
getting attached data . . . 69

H

hosts
  Configuration Server . . . 31
Hot Standby . . . 16

I

identifying driver version . . . 25
importing d2is file . . . 27
indirect call data transfers . . . 71
information menu options . . . 70
initiating
  conferences . . . 71
  transfers . . . 70
input parameters . . . 40
install.man file . . . 27
installing
  IVR . . . 19
  IVR Driver . . . 11, 26
  IVR Server . . . 20, 35
  LCA . . . 20
  setup tasks . . . 19
intended audience . . . 8
Inter Server Call Control . . . 16
italics . . . 76
IVR
  application . . . 67
  defining . . . 21
  installing . . . 19
  name . . . 31
  transfers . . . 70, 71
  vendor-provided . . . 25, 67
IVR Driver
  configuring . . . 11
  configuring multiple . . . 32
  defined . . . 15
  installing . . . 11, 26
  Server Info tab . . . 22
  starting automatically . . . 36
  starting manually . . . 35
  stopping automatically . . . 37
  stopping manually . . . 36
  version . . . 70
  WVR for AIX . . . 7
IVR Interface Option . . . 7, 12
IVR Library . . . 15
IVR Server
  configuration modes . . . 12
  defined . . . 12
  installing . . . 20
  IVR Network T-Server . . . 12
  IVR-Behind-Switch . . . 12, 13
  IVR-In-Front . . . 12, 14

K

key-value pairs . . . 39, 69

L

last event name . . . 70
LCA, installing . . . 20
library usage . . . 13
Load Balancing . . . 16
Load Sharing . . . 16

M

main menu options . . . 69
managed service availability . . . 22
manual start/stop . . . 35, 36
migration . . . 21
monospace font . . . 76
multiple IVR Drivers . . . 32

N

name
  Configuration Server host . . . 31
  IVR . . . 31
  operating system . . . 25
  vendor-provided IVR . . . 25
new in this release . . . 11
notify functions
  NotifyCallEnd . . . 43
  NotifyCallStart . . . 42
  overview . . . 40
number parameters . . . 40

O

objects, importing . . . 28
options
  call data transfer menu . . . 71
  called information menu . . . 70
  calling information menu . . . 70
  configuring . . . 21
  information menu . . . 70
  main menu . . . 69
  router instruction menu . . . 71
  statistics menu . . . 71
  transfer menu . . . 70
  transfer methods menu . . . 71
  user data menu . . . 69
  voice menu . . . 69
original calls . . . 71
output parameters . . . 40

P

parameters . . . 30
  DriverIgnoreReady . . . 23
  DriverReadyWorkMode . . . 23
  DriverRetryTimeout . . . 23
  input . . . 40
  LegacyMode . . . 22
  number . . . 40
  output . . . 40
  string . . . 40
PBX call ID . . . 70
ports
  Configuration Server number . . . 31
  defining . . . 21
pre-installation tasks . . . 19
prestart information . . . 35
printing strings to a log . . . 69
prompts voice menu . . . 69
pseudocode examples . . . 56

R

redundancy methods . . . 15
removing attached data . . . 69
retrieving original call . . . 71
route functions
  GetRequest . . . 59, 71
  overview . . . 41
  pseudocode . . . 56
  RouteAbort . . . 59
  RouteRequest . . . 57, 71
  RouteStart . . . 58, 71
  SendReply . . . 61, 71
router instruction menu options . . . 71
routing, ISCC . . . 16

S

script . . . 69
Server Info tab . . . 22
server redundancy . . . 15
service availability, managed . . . 22
Shutdown Timeout . . . 22
single-step conferences . . . 71
square brackets . . . 76
starting IVR Driver
  automatically . . . 36
  manually . . . 35
state tables . . . 40
statistical functions
  overview . . . 42
  StatPeek . . . 64, 71
statistics menu options . . . 71
stopping IVR Driver
  automatically . . . 37
  manually . . . 36
string parameters . . . 40
superuser . . . 26

T

tags . . . 71
tar files . . . 25
tasks, pre-installation . . . 19
telephone functions
  CallComplete . . . 46
  CallConference . . . 48
  CallConsultConference . . . 49
  CallConsultInit . . . 46
  CallConsultRetrieve . . . 47
  CallConsultTransfer . . . 48
  CallInit . . . 44
  CallTransfer . . . 44
  CallTransferKVList . . . 45
  overview . . . 40
test package. *See* CTP
transfer menu options . . . 70
transfer methods menu options . . . 71
transfers
  call data layer . . . 71
  cancelling call data . . . 71
  completing . . . 71
  direct call data . . . 71
  indirect call data . . . 71
  initiating . . . 70
  using IVR . . . 70, 71
  using T-Server . . . 70, 71
trunk . . . 70
T-Server . . . 68
  connection ID . . . 70
  using for transfers . . . 70, 71
TServer_IVR . . . 13
type styles
  conventions . . . 76
  italic . . . 76
  monospace . . . 76
typographical styles . . . 75, 76

U

user data manipulation functions
  overview . . . 41
  UDataAddKVP . . . 50
  UDataAddList . . . 52
  UDataDelAll . . . 52
  UDataDelKVP . . . 51
  UDataGetKVP . . . 51
user data menu options . . . 69
user-data pairs list . . . 69
using
  library . . . 13
  multiple IVR Drivers . . . 34
utilities . . . 29
UUID . . . 70

V

variable . . . 40
version
  IVR Driver . . . 25, 70
  operating system . . . 25
  vendor-provided IVR . . . 25
version numbering, document . . . 75
voice applications . . . 40
voice menu
  options . . . 69
  prompts . . . 69

W

Warm Standby . . . 16
WVR for AIX
  example . . . 25
  installing IVR . . . 19
  tar file . . . 25

X

XML protocol . . . 15