IVR Interface Option 8.5

IVR Driver for MPS

System Administrator's Guide

The information contained herein is proprietary and confidential and cannot be disclosed or duplicated without the prior written consent of Genesys Telecommunications Laboratories, Inc.

Copyright © 2001–2014 Genesys Telecommunications Laboratories, Inc. All rights reserved.

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.genesys.com for more information.

Each product has its own documentation for online viewing at the Genesys Documentation 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 and the Genesys logo are registered trademarks of Genesys Telecommunications Laboratories, Inc. All other company names and logos may be trademarks or registered trademarks of their respective holders.

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

Technical Support from VARs

If you have purchased support from a value-added reseller (VAR), please contact the VAR for technical support.

Customer Care from Genesys

If you have purchased support directly from Genesys, please contact Genesys Customer Care. Before contacting Customer Care, please refer to the ***Genesys Care Program Guide*** for complete contact information and procedures.

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.genesys.com

Document Version: 85iv_ad_mps_06-2014_v8.5.001.00

Table of Contents

List of Procedures

Preface

Welcome to the *IVR Interface Option 8.5 IVR Driver for MPS System Administrator's Guide* This document describes the IVR Driver for MPS, which is the driver component of Genesys IVR Interface Option 8.5. This document also describes the Genesys-provided functions supported in IVR Interface Option 8.5.

This document is valid only for the 8.5 release of this product.

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

This preface contains the following sections:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 71.

About IVR Interface Option

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 8.5 architecture simplifies the integration of vendor-provided IVRs with the Genesys environment. Genesys IVR Interface Option 8.5 has two major components, the IVR Server and the IVR Driver. For more information about these and other IVR Interface Option 8.5 components, see "Architecture" on page 12.

Intended Audience

This document is primarily intended for contact center administrators, contact center managers, operations personnel, and IVR developers. It has been written with the assumption 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 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@genesys.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 Customer Care 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 Customer Care

If you have purchased support directly from Genesys, please contact Genesys Customer Care.

Before contacting Customer Care, please refer to the *Genesys Care Program Guide* for complete contact information and procedures.

Document Change History

This is the first release of the *IVR Interface Option 8.5 IVR Driver for MPS 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.

Chapter

1 IVR Driver Overview

This chapter describes the architecture of Genesys IVR Interface Option 8.5, 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 8.5

The following changes have been implemented in release 8.5 of the IVR Driver for MPS:

- IVR Driver is built with IVR Library 8.5.

Note: To take advantage of the full range of IVR Driver 8.5 functions, you must use IVR Server 8.5, Genesys Framework 8.x, and Genesys 8.x licensing.

Deployment

This *System Administrator's Guide* describes how to install the IVR Driver for MPS500, and MPS1000, and how to configure it in Configuration Manager.

The IVR Interface Option 8.5 deployment process includes the installation and configuration of the following components:

- IVR Driver
- IVR Server

Before deploying IVR Interface Option 8.5, see the deployment planning chapters in the *IVR Interface Option 8.5 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 8.5 IVR Server System Administrator's Guide*.

For information about the Genesys functions provided for this IVR Driver, see Chapter 5 on page 37. For information about the Customer Test Package, see the Appendix on page 65.

Note: You must make sure that the `LD_LIBRARY_PATH` environmental variable is properly set and in effect when the driver process (`mps2is`) is in use. Alternatively, ensure that the I-Library file is located in the same directory as the driver.

Architecture

This section describes the architecture of IVR Interface Option 8.5. 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 8.5, 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 Balancing, Warm Standby or Hot 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 8 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 8.5 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, 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.

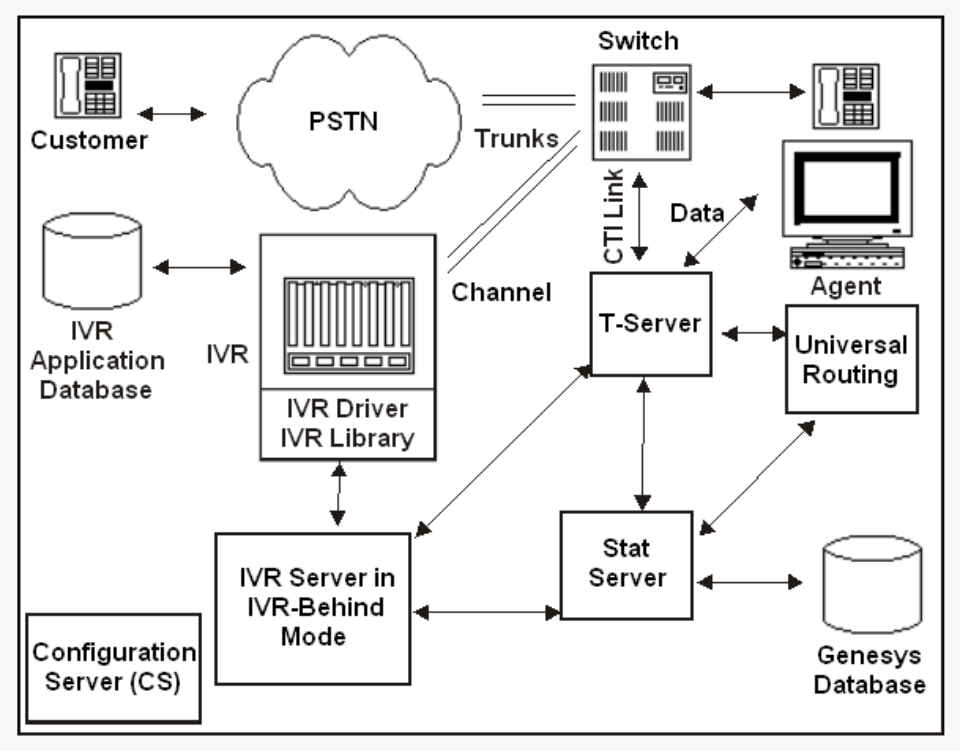


Figure 1: IVR-Behind-Switch Configuration

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.

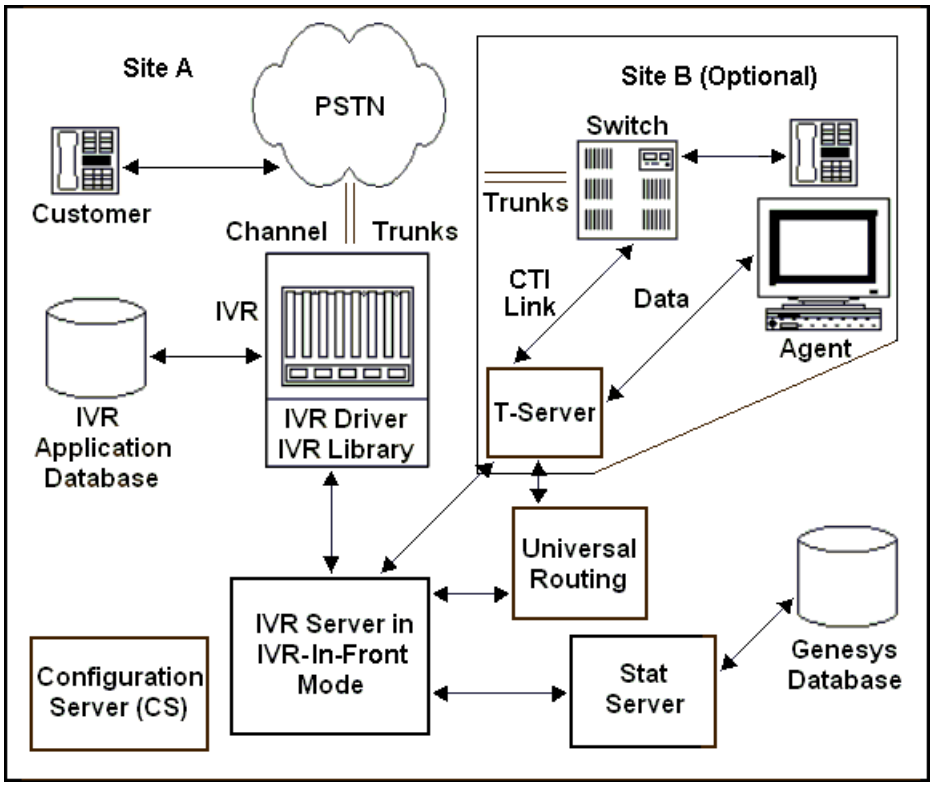


Figure 2: IVR-In-Front Configuration

In the IVR-In-Front configuration shown here, 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 8.5. From release 8, IVR Library is also available as a stand-alone SDK product.

With IVR Interface Option 8.5, 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 8.5 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 8.5 IVR SDK 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 8.5 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 8.5 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 are at version 7.5.009.01as a minimum, and URS is at version 7.5.002.02 as a minimum).

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.

N+1 Redundancy Warm Standby Mode (MPS 3.0 Only)

N+1 redundancy is a system backup configuration in which one node serves as a backup (secondary) node for multiple operational (primary) nodes. The group composed of primary components and their secondary component(s) is referred to as a *cluster*. For more information, see the *Media Processing Server Series System Reference Manual*.

Managed Service Availability

In the management of a complex contact center environment, it is occasionally necessary to remove individual IVR ports, entire IVRs, or entire IVR Server applications from service, due either to planned maintenance, or to unplanned hardware or software failures. Starting with the 7.2 release of IVR Interface Option, a feature called managed service availability enables you do to this.

To use this feature for the IVR-Behind-Switch configuration, you must implement a switch-specific ACD queue that is serviced by one or more IVR Drivers and/or IVR Servers. For the IVR-In-Front configuration, you must use a facility (for example, a router) that prevents additional calls from arriving at

a specific IVR Driver and/or IVR Server, based on Agent state (see “AgentControl Section” on page 22).

In order to use managed service availability to shut down and start drivers, you must enable the agent login/logout and agent ready/not ready mechanisms. Agent activity monitoring is available only for those IVR ports that are configured as ACD agents within the switch.

Shutting Down a Driver

To shut down a driver by using managed service availability, open the Solution Control Interface (SCI), select the IVR Driver application that you want to shut down, and stop it. The following sequence of events occurs:

1. For all ports that are not in use, the IVR Driver immediately sets the corresponding agents to `notready`.
2. The preconfigured shutdown timer begins counting down.
3. As the calls that are in progress end, the IVR Driver sets the corresponding agents to `notready`.
4. The IVR Driver shuts itself down when either of the following occurs:
   - All calls on all ports have ended.
   - The shutdown timer expires. Any calls that are still in progress are terminated immediately.

Note: It is recommended that you set the shutdown timer to a value that is longer than your longest normal call duration.

To set the timer, configure the `Shutdown_Timeout` option, on the `Server Info` tab of the `IVR_Driver` application’s `Properties` dialog box in Configuration Manager.

For information about using managed service availability to stop the IVR Server, see the IVR Interface Option overview chapter in the *IVR Interface Option 8.5 IVR Server System Administrator’s Guide*.

Starting a Driver

To start a driver by using managed service availability, follow the startup procedure described in Chapter 4 on page 33. After the driver is started and makes contact with the IVR Server, the agents will be placed into the state that you have configured for them.

Chapter

2 Pre-Installation Setup for the IVR Driver

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

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 8.5, 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.x chapters in the *Genesys Migration Guide*.

Installing the IVR

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

Install the IVR Driver only on the computer where the vendor-provided IVR 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 8.5 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 8.5 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 in Step 2.

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.

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 8.5 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 8.5 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 8.5 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 8.5 IVR Server System Administrator's Guide*.

Note: 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 8.5 IVR Server System Administrator's Guide*).

AgentControl Section

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.

The following three parameters apply only when LegacyMode is set to false.

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.

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.

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.

Chapter

3 Installing the IVR Driver

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

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

Identifying the Driver Version

To help you identify the driver version, the file name of the Genesys IVR Driver package for MPS 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 MPS

```
mps2-1_SOS5-8_IS_8-5-000-01.tar
```

The file name in this example indicates the following:

1. `mps2-1`—This Genesys IVR Driver package was created for the MPS 2.1 IVR.
2. `SOS5-8`—It runs on the Sun operating system, version 5.8.

3. `IS`—The IVR Interface Option type is `IS` (IVR Server).
4. `8-5-000-01`—The Genesys IVR Driver version is 8.5.000.01.
5. `.tar`—The package is self-installing.

Supported IVR Versions

The IVR Driver for MPS supports only the following Nortel IVR versions and operating systems:

- MPS1000 3.5 on Sun Operating System 5.10 (or Solaris 10)
- MPS1000 3.0 on Sun Operating System 5.10 (or Solaris 10)

The Customer Test Package (CTP) and Genesys `pprotoolkit` are supported by:

- MPS Developer 3.5 for the MPS version 3.5
- MPS Developer 3.0.1 for the MPS version 3.0

Starting the Installation for MPS

Note: The same installation package (`.tar` file) is used for both MPS500 and MPS1000.

Procedure:
Preparing to use N+1 warm standby (MPS 3.0 only)

Start of procedure

1. Install the Genesys IVR Driver on each node and configure it in `/opt/vps/common/etc/GenesysIVRDriver.cfg` on each node.
2. On the primary node, set `startup mode = warm` in the `$MPSHOME/common/etc/rcd.cfg` file.
3. On the secondary node set `startup mode = warm` in the `$MPSHOME/common.standby/etc/rcd.cfg` file.
4. In the 8.x configuration mode, configure two differently named IVR applications. The first host will be designated as the primary MPS host, while the second one will be designated as the secondary MPS host. Both hosts must point to the same `IVR` object.

   For the primary host, configure the IVR Driver specified in the `GenesysIVRDriver.cfg` file to run the first IVR application.

   For the secondary host, configure the IVR Driver to run the second IVR application. The redundant driver then operates in the Nortel warm standby mode.

End of procedure

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

Start of procedure

1. Become `superuser`.
2. On the local MPS1000 host, create a suggested directory called `/home/mps/gcti`. Make sure that it has 20 MB of free space.
3. Insert the IVR Interface Option 8.5 installation DVD.
4. Locate the `MPS1000/solaris` 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, `mps2-1_SOS5-8_IS_8-5-000-01.tar` for MPS 2.1 or `mps3-0_SOS5-10_IS_8-5-000-01.tar` for an MPS 3.0).
6. Copy the `.tar` file into the `/home/mps/gcti` directory that you created in Step 2.
7. At the command line, enter the command to untar the `.tar` file—for example:

   ```
   tar -xvf mps2-1_SOS5-8_IS_8-5-000-01.tar
   ```

This procedure extracts the files shown in Table 1 from the `.tar` file.

Table 1: Files Extracted by Un-tarring Genesys MPS Package

Extracted File	Description
`mps2is`	The IVR Driver for MPS1000.
`*.so`	Dynamic libraries used by `mps2is`.
`GenesysIVRDriver.cfg`	The configuration file for `mps2is`.
`CTP.ppr`, `CTP.mmi`, `CTP.mmd`	These files provide the Customer Test Package that shows how user-defined function calls can be used.
`cf_gsys.c`	This file facilitates the communication between the MPS1000 IVR system and the IVR Driver, `mps2is`.
`Genesys.pprotoolkit`	This toolkit file makes the IVR Driver for MPS1000 function call icons available in the PeriPro environment. You add the toolkit (folder) of icons, and then drag and drop the icons as you author a Genesys-enabled application.
`genesys` subdirectory	Contains the function calls (`function_name.ppr`) and the icons for the function calls (`function_name.icon`)

End of procedure

Installing the IVR Driver

This section describes the main tasks that you must complete in order to install the IVR Driver for MPS.

Installing the Genesys Toolkit

Procedure: Configuring the GenesysIVRDriver.cfg file

Prerequisites

Before beginning this procedure, you must untar the relevant Genesys software package.

Start of procedure

1. Edit the `GenesysIVRDriver.cfg` file, adding your configuration values for the command parameters:

   Note: All the configuration values are specified in one command line.

   a. For the `New style indicator` parameter, enter `GS_7.0`.
   b. For the `locating token` parameter, enter the name of the token that identifies the IVR Driver instance.
   c. For the `cfg_server_host` parameter, enter the name of the host on which the Configuration Server will be running.
   d. For the `cfg_server_port` parameter, enter the port number of the Configuration Server, as defined in the `TServer_IVR` application's `Properties` dialog box in Configuration Manager. The default value is `2020`.
   e. For the `driver_app_name` parameter, enter the name of the `IVR_Driver` application, as defined in the `TServer_IVR` 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.

   f. For the `ivr obj` parameter, enter the name assigned to the IVR in Configuration Manager (for example, `mps`).

g. The `cps/tms_number` parameter is the system identifier for the CPS/TMS equipment with which the IVR Driver for MPS interacts. Enter the Nortel CPS or TMS number. The default value is `1`.

h. For the `first_ivr_port` parameter, enter the number to be assigned to the first port in this IVR Driver's port range.

i. For the `last_ivr_port` parameter, enter the number to be assigned to the last port in this IVR Driver's port range.

j. For the `internal_port` parameter, enter the number of the port on which the IVR Driver communicates with the IVR.

k. 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.

l. For the `backup_cfg_server_port` parameter, enter the port number specified for the backup Configuration Server `Application` object in Configuration Manager or `0` (zero) if not to be used.

m. For the `clientside_port_for_cfg_server` parameter, enter the desired client side port number for the connection to the Configuration Server application or `0` (zero) if not to be used.

n. If you choose to start your IVR Driver from the Genesys Solution Control Interface (SCI) (see page 34), you must:

- Install the Genesys Local Control Agent (LCA) that is provided on the Genesys Framework 8.1 product CD.
- Run the `install.sh` script that is located in the directory to which you installed the IVR Driver. This script creates a `gs_start_drv.sh` script that the SCI can invoke.

2. Using root access, move the `Genesys.pprotoolkit` file and the `genesys` subdirectory to the `$VPSHOME/PERIppro` directory. The `genesys` directory contains all the necessary `.icon` and `.ppr` files for creating the Genesys toolkit.

3. Move the `GenesysIVRDriver.cfg` file to the `/opt/vps/common/etc` directory.

Note: After you complete Steps 2 and 3, you no longer need root access.

4. Restart the PeriProducer application to enable the Genesys toolkit.

End of procedure

Generating the Libraries

Procedure: Generating libraries

Prerequisites

Before you can use the IVR Driver for MPS, you must compile and build the libraries that are necessary for communication between the IVR Driver and the MPS IVR.

Note: A version of the C compiler that is compatible with your operating system must be installed on the same computer as the IVR; otherwise, the compile in Step 4 will fail.

Purpose: The `cf_gsys.c` file is used to communicate with the IVR Driver application.

Start of procedure

1. Open a `Command Tool` window, and go to the directory in which the IVR Driver is installed.
2. If debug is not required, proceed to Step 4. If debug is required, do the following:
   a. Open the `cf_gsys.c` file in a text editor.
   b. Edit the `#define DEBUG` line to change the `DEBUG` flag.
3. Save the `cf_gsys.c` file. This will cause the creation of debug log files (`*.dbglog`) in `/home/peri`.
4. Compile the `cf_gsys.c` file by entering the following command at the command line:

   ```
   makecall cf_gsys.c
   ```

   The MPS IVR begins generating the libraries.

End of procedure

For more information about generating libraries, see the vendor-provided MPS IVR documentation.

Note: The `makecall` function is provided by Nortel on the MPS IVR for the purpose of compiling and building libraries that can be used to call external functions—that is, functions that are not provided through the built-in features of PeriProducer.

To use it successfully, a Sun-supported compiler must be installed.

Upgrading the I-Library

Starting with release 8.0, 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_32.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 `mps2is`.
2. Copy the `libilib_32.so` file into the directory where you previously placed the driver's I-Library.

   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

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 65.

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 Server and the IVR Driver. For more information about installing and configuring the IVR Server, see the *IVR Interface Option 8.5 IVR Server System Administrator's Guide*.

This chapter contains the following sections:

Prestart Information

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

Starting the IVR Driver

You can start the IVR Driver for MPS manually or through the Genesys Solution Control Interface (SCI).

Starting Manually

Procedure: Starting the IVR Driver manually

Start of procedure

1. Open a `Command Tool` window, and go to the directory in which the IVR Driver is installed.
2. At the command line, enter the following command:

   ```
   set LD_LIBRARY_PATH = /home/peri/gcti
   ```

   In this command, `/home/peri/gcti` is this library path setting. It must match the directory to which you unpacked the `*.so` files from the `.tar` file at installation. If you used a directory other than `/home/peri/gcti`, enter that directory instead.

   Alternatively, if you have the I-Library file located in the same directory where the driver is, you do not need to set `LD_LIBRARY_PATH`.
3. Enter the following command:

   ```
   mps2is [ivr_DriverName]
   ```

   In this command, `ivr_DriverName` is the name of the IVR Driver (as configured in the `GenesysIVRDriver.cfg` and in Configuration Manager).

End of procedure

Starting Through SCI

Procedure: Starting the IVR Driver using the SCI

Start of procedure

1. Create an `IVR_Driver` application as described in "Configuring the IVR Driver" on page 21, and in the *IVR Interface Option 8.5 IVR Server System Administrator's Guide*.
2. Click the `Start Info` tab of the `IVR_Driver` application's `Properties` dialog box, and configure the following settings (see Figure 3 on page 35):
   - In the `Working Directory` box, enter the directory path to which you installed the IVR Driver.
   - In the `Command Line` box, enter the following line:

     `<Working Directory path>/gs_start_drv.sh`

- In the `Command Line Arguments` box, enter the name of the IVR Driver instance to be started (as configured in the `GenesysIVRDriver.cfg` file).

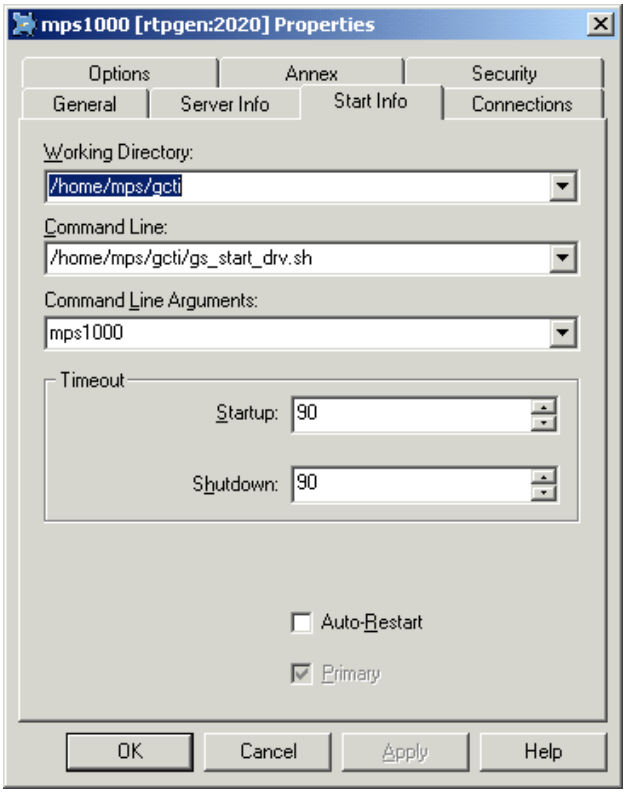


Figure 3: IVR_Driver Application Properties Dialog Box—Start Info Tab

3. Make sure that the `gs_start_drv.sh` script exists in the working directory of your IVR. If it does not, run the `install.sh` script to create it.
4. In the SCI, right-click your `IVR_Driver` application, and then select `Start`.

End of procedure

Stopping the IVR Driver

You can stop the IVR Driver manually or through the SCI.

Stopping Manually

To stop the IVR Driver manually, open a `Command Tool` window, and then do one of the following at the command line:

- Press `CTRL+C`.
- Enter the UNIX `kill` command.

Stopping Through SCI

To stop the IVR Driver by using the SCI, right-click your `IVR_Driver` application, and then select `Stop`.

Running Multiple Instances of the IVR Driver

Note: You must configure a unique `ivr_DriverName` for each instance of an MPS IVR Driver.

Procedure: Manually starting multiple instances of the IVR Driver

Start of procedure

1. Open a `Command Tool` window, and go to the directory in which the IVR Driver is installed.
2. At the command line, enter the following command:

   ```
   set LD_LIBRARY_PATH = /home/peri/gcti
   ```

   In this command, `/home/peri/gcti` is this library path setting. It must match the directory to which you unpacked the `*.so` files from the `.tar` file at installation. If you used a directory other than `/home/peri/gcti`, enter that directory instead.
3. Enter the following command for each IVR Driver (using a unique `ivr_DriverName` for each instance):
   - `mps2is [ivr_DriverName]`

In this command, `ivr_DriverName` must match the name of an IVR Driver (as configured in the `GenesysIVRDriver.cfg` file and in Configuration Manager).

End of procedure

Chapter

5 Functions

After you install, configure, and start the IVR Driver, you can use and test IVR Interface Option 8.5 as a user function within the vendor-provided IVR application. This chapter describes the functions that IVR Interface Option 8.5 supports. It contains the following sections:

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 2048 bytes. The length of any individual data string is limited to 2048 bytes. The combined key-value pairs (including delimiters) on a given call instance can total no more than 4096 bytes.
- Only ISO-8859-1 characters are supported.

Genesys-Provided Functions

The Genesys IVR Driver provides all the functions described in this chapter. You can call these from within the application of the MPS system. When a script is initiated, these external functions make the corresponding IVR Driver interface functionality available to the Interactive Voice Response (IVR) system. For more information about using the external functions during the script building process, see the vendor-provided MPS IVR documentation.

All *input parameters* specify parameters that are sent from the script to the `mps2is` driver, and all function *output parameters* specify parameters that are returned by `mps2is`.

Note: The type of parameter (`String` or `Number`) is given in MPS terminology.

When a function returns more than one variable, the script logic must check the `Result` before any other variable can be used. Other variables can be used only if the `Result` is successful.

To execute a request:

1. Call the required function. After the function is executed, a Request ID (`ReqID`) is returned to confirm this.
2. Call the `GetReply` function to obtain the result of this executed function. `GetReply` has a `ReqID` parameter that verifies the previously called function (see Step 1).

The "Output" section for each function description describes the results that are returned by the `GetReply` function.

Each Genesys-provided function is available through a function button in the PeriPro application (see Figure 4 on page 39).

Note: The function buttons shown in Figure 4 on page 39 are from PeriPro version 3.0.

Figure 4: PeriPro Function Buttons

Table 2 on page 40 lists the Genesys-provided functions, including a graphic of the PeriPro function button for each. The remaining sections of this chapter describe each of these functions in detail.

Table 2: Functions

Function	Button	Summary	Page
Notify Functions			
NotifyCallStart		Notifies the IVR Server that the call has started.	Page 43
NotifyCallEnd		Notifies the IVR Server that the call has ended.	Page 44
Telephone Functions			
CallInit		Initiates a new call to the destination DN.	Page 45
CallTransfer		Makes a transfer to the DN (mute transfer).	Page 45
CallComplete		Completes the current call.	Page 46
CallConference		Completes the call by creating a conference.	Page 47
CallConsultInit		Initiates a consulting call with a DN.	Page 47
CallConsultRetrieve		Retrieves the original consulting call.	Page 48
CallConsultTransfer		Completes the consulting call by making a transfer.	Page 49

Table 2: Functions (Continued)

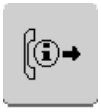
Function	Button	Summary	Page
CallConsultConference		Completes the consulting call by creating a conference.	Page 50
User Data Manipulation Functions			
UDataGetKVP		Returns an attached key value that is associated with a key-value pair.	Page 50
UDataAddKVP		Attaches (updates) a key-value pair to a call.	Page 51
UDataDelKVP		Deletes a key-value pair from the call database.	Page 52
UDataAddList		Attaches (updates) a list of key-value pairs to a call.	Page 52
UDataDelAll		Deletes all user data from the call database.	Page 53
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

Table 2: Functions (Continued)

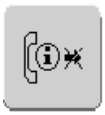
Function	Button	Summary	Page
CDT_Cancel		Cancels the `CDT_Init` request.	Page 55
Route Functions			
RouteRequest		Requests a service point from the router.	Page 57
RouteStart		Requests the start of a new session from the router (assigned to the service point).	Page 58
RouteAbort		Aborts the previously requested service session from the router.	Page 59
GetRequest		Takes the request (or reply) for the next job that the IVR is to perform.	Page 60
SendReply		Sends a reply for the previously called `GetRequest` function.	Page 61
General-Purpose Functions			
GetReply		Takes the reply for a previously issued request.	Page 62
GetVersion		Returns the version number of the IVR Driver or the IVR Library.	Page 63
ToLog		Places a specified text string into the log file.	Page 63

Table 2: Functions (Continued)

Function	Button	Summary	Page
Statistical Functions			
StatPeek		Sends a request to the Stat Server to provide information about a predefined statistic.	Page 64

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 calls must be `GetCallInfo(EventName)` and `GetReply`. These calls 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 `GetReply` call for this request.
3. Issue a `GetCallInfo` call on the line, asking for the `EventName`.
4. Issue a `GetReply` call for the `GetCallInfo` call.
5. Verify that your application receives an `EventEstablished`, `EventAttachedDataChanged`, or `EventPartyChanged` event.

The IVR Driver is ready to perform additional application programming interface (API) requests only after the `EventEstablished`, `EventAttachedDataChanged`, or `EventPartyChanged` event has been received.

Input

`String szCallID`

The Call ID.

`String szDNIS`

DNIS information.

`String szANI`

ANI information.

`String szCDT_Tag`

A tag used by the Call Data Transfer protocol.

Note: The `String szDNIS` and `String szANI` parameters can be included (if available) if the PBX is not present at the site, and the IVR Server operates in IVR-In-Front mode. If the PBX is present at the site, these strings must be empty.

Output

`ReqID`

If `= 0`, the function failed.

If `= Request ID`, a Request ID must be used later with the `GetReply` function.

NotifyCallEnd

This function notifies the IVR Server that the IVR channel has disconnected the call. No Genesys function call other than a `GetCallInfo(LastEvent)` is expected for the call session after this function.

Genesys recommends that this function be called as the last function in the script. This function must be invoked *after* the `Disconnect` action of the State Table.

Input

None.

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

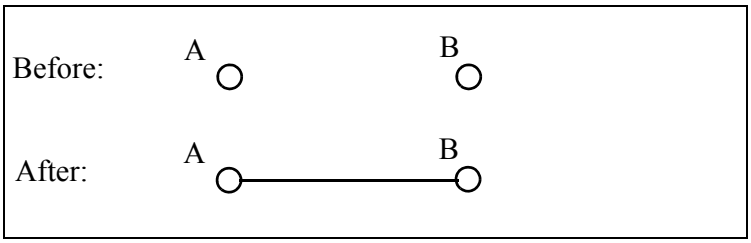
Telephone Functions

CallInit

This function initiates a new call to the destination DN (B) (see the scenario that follows). The `CallInit` function works with the switch, which is monitored by T-Server.

This function is used only for an IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

If `= 0`, the function failed.

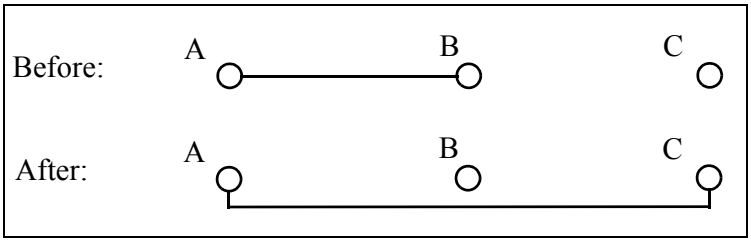
If `= Request ID`, a Request ID can be used later with the `GetReply` function.

CallTransfer

This function produces a *direct transfer* without an intervening conference call (see the description in the *Genesys Events and Models Reference Manual*). 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 that follows). In other words, the party who initiates the call (A) is disconnected from the original DN (B) and reconnected to the destination DN (C).

This function is used only for an IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

If `= 0`, the function failed.

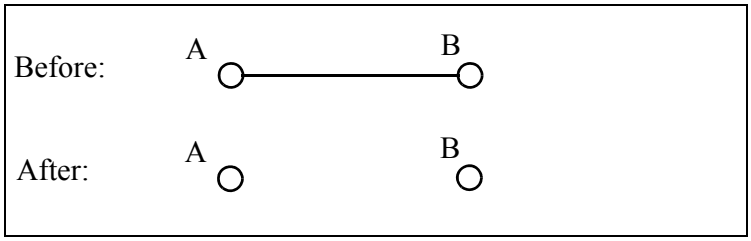
If `= Request ID`, a Request ID can be used later with the `GetReply` function.

CallComplete

This function completes the current call (see the scenario that follows).

This function is used only for an IVR-Behind-Switch configuration.

Scenario



Input

None.

Output

`Number Result`

If `= 0`, the function failed.

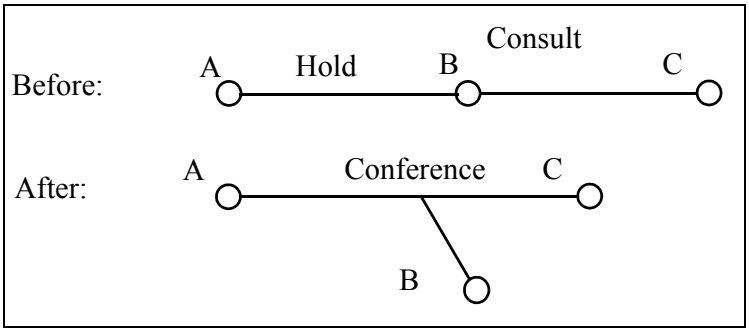
If `= Request ID`, a Request ID can be used later with the `GetReply` function.

CallConference

This function makes a *conference call* to a new DN (see the description in the *Genesys Events and Models Reference Manual*) by connecting the two parties (A and B) of the original conversation with an additional party (C) (see the scenario that follows). 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 an IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

CallConsultInit

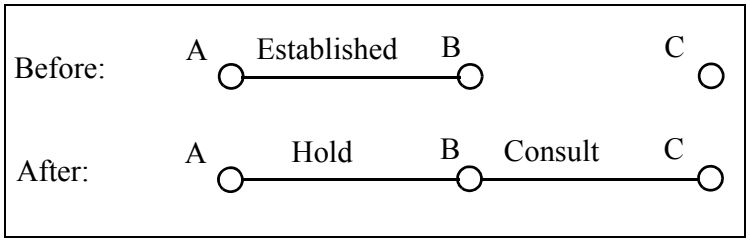
This function initiates a *consulting call* (see the description in the *Genesys Events and Models Reference Manual*). The original party (A) is placed on hold for the duration of the consulting call, and the party (B) who requests the `CallConsultInit` function is involved in a new consulting call with a third party (C) (see the scenario that follows).

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

- `CallConsultRetrieve` (see page 48)
- `CallConsultTransfer` (see page 49)
- `CallConsultConference` (see page 50)

This function is used only for an IVR-Behind-Switch configuration.

Scenario



Input

`String DstDN`

The phone number of the new destination party.

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

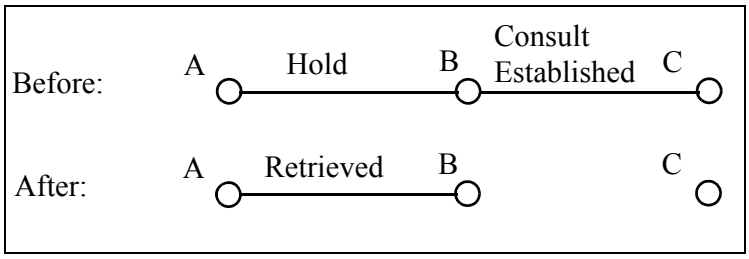
CallConsultRetrieve

This function completes a consulting call. The original party (A) is placed on hold for the duration of the consulting call; the second party (B) then creates a consulting call to the third party (C) (see the scenario that follows). 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 an IVR-Behind-Switch configuration.

Note: Use this function only after `CallConsultInit` (see page 47) has been successfully completed.

Scenario



Input

None.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

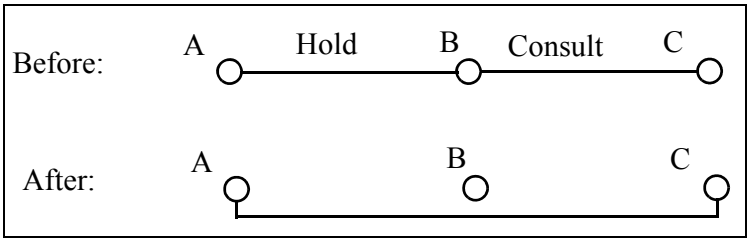
CallConsultTransfer

This function completes the consulting call by making a *transfer* (see the description in the *Genesys Events and Models Reference Manual*). During the transfer, the caller (A) is placed on hold; the original recipient (B) makes a call to a new party (C); and 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 an IVR-Behind-Switch configuration.

Note: Use this function only after `CallConsultInit` (see page 47) has been successfully completed.

Scenario



Input

None

Output

`Number Result`

If = `0`, the function failed.

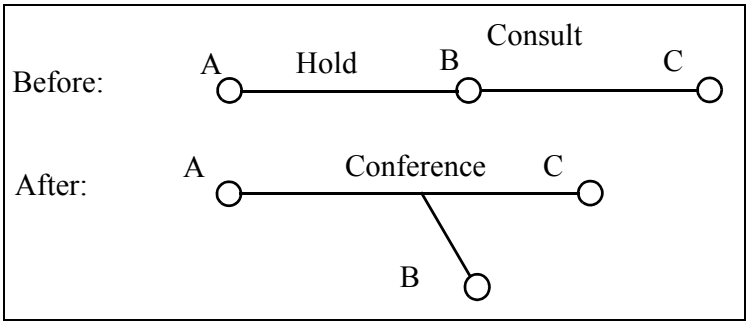
If = `Request ID`, a Request ID can be used later with the `GetReply` function.

CallConsultConference

This function completes a consulting call by creating a connection between the two parties (A and B) of the original conversation with an additional party (C) from the consulting call (see the scenario that follows). As a result, three or more people can participate in the phone conversation, talking from three or more extensions. See the description in the *Genesys Events and Models Reference Manual*. This function is used only for an IVR-Behind-Switch configuration.

Note: Use this function only after `CallConsultInit` (see page 47) has been successfully completed.

Scenario



Input

None.

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

User Data Manipulation Functions

UDataGetKVP

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

Input

`String szKey`

The key value of the searched pair.

Note: The length of the `szKey` string can be no more than 2048 bytes. The length of any individual data string is limited to 2048 bytes. The combined key-value pairs (including delimiters) on a given call instance can total no more than 4096 bytes.

Output

`ReqID`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

UDataAddKVP

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

Note: Both the `User Key` and `User Value` fields must be filled out—for example:

- `Key = Customer ID`
- `Value = 1583`

Input

`String szKey`

The key value of the attached pair.

`String szValue`

The data value of the attached pair.

Output

`ReqID`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

UDataDelKVP

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

Input

`String szKey`

The key string for deleting the pair.

Note: The length of the `szKey` string can be no more than 2048 bytes.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

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 following format:

`<delimiter>Key<delimiter>Value`

Note: The length of any individual `Key` string can be no more than 2048 bytes. The length of any individual `Value` data string is limited to 2048 bytes. The combined key-value pairs (including delimiters) in the `KVList` string can total no more than 4096 bytes.

Example:

```
*Customer ID*4942
```

This example shows a key-value pair that contains the key `Customer ID`, with the value `4942`, separated by the delimiter `*`.

Example:

```
#Customer ID#4943#Customer2 ID#1234
```

In this example, the delimiter is `#`.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

UDataDelAll

This function deletes all user data from the current call.

Input

None.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

Call Information Functions

GetCallInfo

This function requests information related to an active call.

Input

`String TypeInfo`

The type of requested information. Table 3 lists the possible `TypeInfo` values.

Note: If a particular value (for example, `ANI`) is not available, the string `NULL` is returned by default.

Table 3: TypeInfo Values

Value	Requested Information
All	All call information that is 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

Table 3: TypeInfo Values (Continued)

Value	Requested Information
EventName	Name of the last event received on the current IVR channel
FirstHomeLocation	First Home Location of the call
OtherDN	Destination DN (if any)
OtherQueue	Destination queue (if any)
OtherTrunk	Destination trunk (if any)
ThisDN	Current DN assigned to the IVR channel (port)
ThisQueue	Current queue assigned to the IVR channel (port)
ThisTrunk	Current trunk assigned to the IVR channel (port)

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

`String Info`

The requested information from the call database.

Call Data Transfer Functions

CDT_Init

This function requests an access number to make a Call Data Transfer (CDT) between two T-Servers. The request can be canceled later by calling the `CDT_Cancel` function (see page 55).

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 4 lists the possible Call Data Transfer types.

Table 4: 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	Used for direct dialing to the destination DN, with a tag coming *in* to the client (that is, the Call Data Transfer generates the tag).
ReRoute	Passes unchanged to the IVR Server.

Note: If your IVR application passes in a string that is not equal to one of these CDT 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 = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

CDT_Cancel

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

Input

None.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

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 5 and 6 shows the correct order in which the routing APIs should 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 5: 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 6: 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.

Note: For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 65.

Input

`String Service`

The name of the service that is being requested from the router.

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

`String NextService`

The name of the next service point/destination DN provided by the router..

Note: Universal Routing Server (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 that URS has requested, there is no way to indicate that the new route has failed. If this error condition could occur, you need to use the `RouteStart` function instead of `RouteRequest`.

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, with the `RouteAbort` function.

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

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 65.

Input

`szRouterPoint`

The name associated with the service that is being requested from the router. The router must be loaded with a valid routing strategy at this location.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

RouteAbort

This function aborts the previously requested service session from the router.

Note: For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 65.

Input

`ReqID`

The Request ID assigned to the `RouteStart` function.

Output

`Number Result`

If = `0`, the function failed.

If = `Request ID`, a Request ID can be used later with the `GetReply` function.

GetRequest

This function takes the request (or reply) for the next operation that the IVR is to perform.

Note: For more information about how to code and use the `RouteStart`, `GetRequest`, `SendReply`, and `RouteAbort` function calls, see the Appendix on page 65.

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

This format begins with a delimiter that is used later in the string to divide fields. The number of actual parameters that come with a command depends on the particular command type. The command string contains at least a delimiter and a `CommandType` field. Table 5 lists the set of Genesys routing commands and their equivalent T-Server treatment types.

Table 5: Routing Commands

Routing Command	T-Server Treatment
IVR	TreatmentIVR
Music	TreatmentMusic
RingBack	TreatmentRingBack
Silence	TreatmentSilence
Busy	TreatmentBusy
CollectDigits	TreatmentCollectDigits
PlayAnnouncement	TreatmentPlayAnnouncement
PlayAnnouncementAndDigits	TreatmentPlayAnnouncementAndDigits
VerifyDigits	TreatmentVerifyDigits
RecordAnnounce	TreatmentRecordUserAnnouncement
DeleteAnnounce	TreatmentDeleteUserAnnouncement
CancelCall	TreatmentCancelCall
PlayApplication	TreatmentPlayApplication
SetDefaultRoute	TreatmentSetDefaultRoute

Table 5: Routing Commands (Continued)

Routing Command	T-Server Treatment
TextToSpeech	TreatmentTextToSpeech
TextToSpeechAndDigits	TreatmentTextToSpeechAndDigits
FastBusy	TreatmentFastBusy
RAN	TreatmentRAN

For a detailed description of treatments and parameters, see the *Universal Routing 8 Reference Manual*. See the vendor-provided MPS IVR documentation for a description of whether and how the IVR supports these treatments.

Input

`Timeout`

The amount of time (in seconds) that the function waits for a new request/reply. The function is blocked until either a new request/reply arrives or the timeout expires, whichever comes first.

Output

`Number Result`

If `= 0`, if function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

In the case of success, it is possible for the Request ID and request/reply string to be taken in the `szData` string. An error means that I-Library did not receive a request/reply within the specified time frame.

`String szData`

Request-specific information.

SendReply

This function sends a reply from the IVR script to the 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 65.

Input

`ReqID`

The Request ID from the router, which was received by using the `GetRequest` function.

`szResult`

The result of the requested job. The possible values are `OK` and `NOTOK`. These values are case-sensitive.

`szData`

A string that contains reply information. The specific content depends on the type of request.

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

General-Purpose Functions

GetReply

This function gets the reply for a previously sent request.

Input

`ReqID`

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 can be called later for the required Request ID.

In the case of success, it is possible to obtain the expected value from the `Reply` string.

`String Reply`

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

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.

GetVersion

This function requests the version number of the `mps2is` server or another service.

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 `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

ToLog

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

Input

`String PrintString`

The string that is printed to the log.

`String Service`

The name of the service for which the log is written.

- If this argument is the name of the IVR, as defined in Configuration Manager, the function sends the string to the IVR Server.
- 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 `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

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 IVR Server options section in the *IVR Interface Option 8.5 IVR Server System Administrator's Guide.*

Input

`String StatName`

The name of the required statistic. The name must be defined in the configuration environment (by using Configuration Manager).

Output

`Number Result`

If `= 0`, the function failed.

If `= Request ID`, a Request ID can be used later with the `GetReply` function.

Appendix

Customer Test Package

This appendix provides information about the Customer Test Package (CTP) that is included on the IVR Driver product DVD. It contains the following sections:

Introduction

The IVR Driver includes a CTP that contains an Interactive Voice Response (IVR) application and a corresponding voice prompt database. You can install the IVR application on the vendor-provided IVR, and use it to test the configuration of the IVR Interface Option 8.5 components and other Genesys applications.

Using the Customer Test Package

You can use the CTP with both IVR-In-Front and IVR-Behind-Switch configurations. However, some CTP functions are available in only one of these configurations.

Procedure:
Running the CTP application

Start of procedure

1. Install the CTP.
2. Load the `CTP.ppr` file, generate the `CTP.vex` file, and assign the test application to the IVR channels.
3. If you are using the IVR-Behind-Switch configuration, start the T-Server.

4. Start the `TServer_IVR` application.
5. Start the IVR Driver.
6. 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 8.5 with the Genesys Framework software.
7. To run the CTP, follow the voice menu prompts. For more information, see "CTP Voice Menu."

End of procedure

Next Steps

To learn how to invoke the Genesys IVR Interface Option 8.5 functions, follow the script example within the application.

For explanations of the calls within the CTP application, read the notes attached to the various calls.

CTP Voice Menu

Table 6 lists the voice menu options that are 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

Table 6: CTP Voice Menu Options (Continued)

Option	Key
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
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
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

Table 6: CTP Voice Menu Options (Continued)

Option	Key
Transfer Menu	
Transfer using the IVR	Press 1
Transfer using T-Server	Press 2
Initiate a transfer	Press 3
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

Table 6: CTP Voice Menu Options (Continued)

Option	Key
Statistics Menu	
Perform a StatPeek	Press 1
Return to the Main Menu	Press 0

Genesys

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 8.5 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 Documentation website.

Genesys

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

Information about supported operating systems and third-party software is available on the Genesys Documentation website in the following documents:

- *Genesys Supported Operating Environment Reference Guide*
- *Genesys Supported Media Interfaces Reference Manual*

Consult the following additional resources as necessary:

- *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 that are 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.

For additional system-wide planning tools and information, see the release-specific listings of System-Level Documents on the Genesys Documentation website.

Genesys product documentation is available on the:

- Genesys Customer Care website.
- Genesys Documentation website.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesys.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:

85fr_ref_06-2014_v8.5.001.00

You will need this number when you are talking with Genesys Customer Care 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 74).	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

A

B

C

D

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X