Genesys Voice Platform 7.6

Voice Application Reporter

SDK Developer’s Guide

About Genesys

Genesys Telecommunications Laboratories, Inc., a subsidiary of Alcatel-Lucent, is 100% focused on software for call centers. Genesys recognizes that better interactions drive better business and build company reputations. Customer service solutions from Genesys deliver on this promise for Global 2000 enterprises, government organizations, and telecommunications service providers across 80 countries, directing more than 100 million customer interactions every day. Sophisticated routing and reporting across voice, e-mail, and Web channels ensure that customers are quickly connected to the best available resource—the first time. Genesys offers solutions for customer service, help desks, order desks, collections, outbound telesales and service, and workforce management. Visit 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

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

Technical Support from VARs

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

Technical Support from Genesys

If you have purchased support directly from Genesys, please contact Genesys Technical Support at the following regional numbers:

Region	Telephone	E-Mail
North and Latin America	+888-369-5555 or +506-674-6767	support@genesyslab.com
Europe, Middle East, and Africa	+44-(0)-118-974-7002	support@genesyslab.co.uk
Asia Pacific	+61-7-3368-6868	support@genesyslab.com.au
Japan	+81-3-6361-8950	support@genesyslab.co.jp

Prior to contacting technical support, please refer to the *Genesys Technical Support 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 7 Licensing Guide*.**

Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

Document Version: 76gvp_dev-var_02-2008_v7.6.001.00

Table of Contents

Preface **5**

Intended Audience 6
Usage Guidelines 6
Chapter Summaries 8
Document Conventions 8
Related Resources 10
Making Comments on This Document 11

Chapter 1 **About the Voice Application Reporter SDK** **13**

Overview 13
Components 14
Scope of Use 14
Architecture 14
API Overview 16
- VoiceAppCall Class 16
- VARException Class 29

Chapter 2 **About the Code Samples** **31**

Using the Code Samples 31
VAR Code Samples 32
- Call Start Activities 32
- Call End Activities 33
- Start IVR Action Activities 34
- End IVR Action Activities 35
- Set Custom Data 35

Index **37**

Preface

Welcome to the *Genesys Voice Platform 7.6 Voice Application Reporter SDK Developer's Guide.* This document will show you how to develop VoiceXML applications that interface with the Voice Application Reporter (VAR) database and generate application reports. It includes a high-level overview of the features and functions of the VAR Software Development Kit (SDK) 7.6, as well as information about its architecture and deployment planning materials. This document is valid only for the 7.6 release of this product.

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

This preface provides an overview of this document, identifies the primary audience, introduces document conventions, and lists related reference information. It contains the following sections:

- Intended Audience, page 6
- Usage Guidelines, page 6
- Chapter Summaries, page 8
- Document Conventions, page 8
- Related Resources, page 10
- Making Comments on This Document, page 11

The VAR SDK is a Java class with a set of application programming interfaces (APIs) that enable VoiceXML application developers to interface with the VAR database and generate application reports. The Java VAR Client runs on the same machine as the VoiceXML application server.

Intended Audience

This document, primarily intended for programmers who develop Java-based applications for contact center supervisors or contact center managers, 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 Java programming and database design concepts.

Usage Guidelines

The Genesys developer materials outlined in this document are intended to be used for the following purposes:

- Creation of contact-center agent desktop applications associated with Genesys software implementations.
- Server-side integration between Genesys software and third-party software.
- Creation of a specialized client application specific to customer needs.

The Genesys software functions available for development are clearly documented. No undocumented functionality is to be utilized without Genesys's express written consent.

The following Use Conditions apply in all cases for developers employing the Genesys developer materials outlined in this document:

1. Possession of interface documentation does not imply a right to use by a third party. Genesys conditions for use, as outlined below or in the *Genesys Developer Program Guide,* must be met.

2. This interface shall not be used unless the developer is a member in good standing of the Genesys Interacts program or has a valid Master Software License and Services Agreement with Genesys.

3. A developer shall not be entitled to use any licenses granted hereunder unless the developer's organization has met or obtained all prerequisite licensing and software as set out by Genesys.

4. A developer shall not be entitled to use any licenses granted hereunder if the developer's organization is delinquent in any payments or amounts owed to Genesys.

5. A developer shall not use the Genesys developer materials outlined in this document for any general application development purposes that are not associated with the above-mentioned intended purposes for the use of the Genesys developer materials outlined in this document.
6. A developer shall disclose the developer materials outlined in this document only to those employees who have a direct need to create, debug, and/or test one or more participant-specific objects and/or software files that access, communicate, or interoperate with the Genesys API.
7. The developed works and Genesys software running in conjunction with one another (hereinafter referred to together as the “integrated solutions”) should not compromise data integrity. For example, if both the Genesys software and the integrated solutions can modify the same data, then modifications by either product must not circumvent the other product’s data integrity rules. In addition, the integration should not cause duplicate copies of data to exist in both participant and Genesys databases, unless it can be assured that data modifications propagate all copies within the time required by typical users.
8. The integrated solutions shall not compromise data or application security, access, or visibility restrictions that are enforced by either the Genesys software or the developed works.
9. The integrated solutions shall conform to design and implementation guidelines and restrictions described in the *Genesys Developer Program Guide* and Genesys software documentation. For example:
   a. The integration must use only published interfaces to access Genesys data.
   b. The integration shall not modify data in Genesys database tables directly using SQL.
   c. The integration shall not introduce database triggers or stored procedures that operate on Genesys database tables.

Any schema extension to Genesys database tables must be carried out using Genesys Developer software through documented methods and features.

The Genesys developer materials outlined in this document are not intended to be used for the creation of any product with functionality comparable to any Genesys products, including products similar or substantially similar to Genesys’s current general-availability, beta, and announced products.

Any attempt to use the Genesys developer materials outlined in this document or any Genesys Developer software contrary to this clause shall be deemed a material breach with immediate termination of this addendum, and Genesys shall be entitled to seek to protect its interests, including but not limited to, preliminary and permanent injunctive relief, as well as money damages.

Chapter Summaries

In addition to this preface, this document contains the following chapters:

- Chapter 1, "About the Voice Application Reporter SDK," on page 13, provides an overview of the VAR SDK and its architecture.
- Chapter 2, "About the Code Samples," on page 31, provides examples of the code that you can use to build Java interfaces to the VAR database with the VAR SDK.

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:

76fr_ref_02-2008_v7.6.000.00

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

Type Styles

Italic

In this document, italic is used for emphasis, for documents' titles, for definitions of (or first references to) unfamiliar terms, and for mathematical variables.

Examples:

- Please consult the *Genesys 7 Migration Guide* for more information.
- *A customary and usual practice* is one that is widely accepted and used within a particular industry or profession.
- Do *not* use this value for this option.
- The formula, $x + 1 = 7$ where x stands for . . .

Monospace Font

A monospace font, which looks like `teletype or typewriter text`, is used for 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; and code samples.

Examples:

- Select the `Show variables on screen` check box.
- Click the `Summation` button.
- In the `Properties` dialog box, enter the value for the host server in your environment.
- In the `Operand` text box, enter your formula.
- Click `OK` to exit the `Properties` dialog box.
- The following table presents the complete set of error messages T-Server® distributes 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.

Monospace is also used for any text that users must manually enter during a configuration or installation procedure, or on a command line:

Example:

- Enter `exit` on the command line.

Screen Captures Used in This Document

Screen captures from the product GUI (graphical user interface), as used in this document, may sometimes contain a minor spelling, capitalization, or grammatical error. 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.

Square Brackets

Square brackets indicate that a particular parameter or value is optional within a logical argument, a command, or some programming syntax. That is, the parameter's or value's presence is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information. Here is a sample:

```
smcp_server -host [/flags]
```

Angle Brackets

Angle brackets indicate a placeholder for a value that the user must specify. This might be a DN or port number specific to your enterprise. Here is a sample:

```
smcp_server -host <confighost>
```

Related Resources

Consult these additional resources as necessary:

- *Genesys Voice Platform 7.6 Voice Application Reporter Deployment and Reference Manual,* which describes how to install the Voice Application Reporter and how to use its interface.
- *Genesys Voice Platform 7.6 Studio Deployment Guide,* which provides installation instructions for Genesys Studio.
- *Genesys Voice Platform 7.6 VoiceXML 2.1 Reference Manual,* which provides information about developing VoiceXML 2.1 applications on Genesys Voice Platform (GVP). It presents VoiceXML 2.1 concepts and provides examples that focus on the GVP implementation of VoiceXML. It also describes the platform extensions to VoiceXML that Genesys provides.
- *Voice Extensible Markup Language (VoiceXML) Version 2.1, W3C Candidate Recommendation (CR) 13 June 2005.* A *Candidate Recommendation* is a mature technical report that, after wide review for technical soundness and implementability, the W3C (World Wide Web Consortium) has sent to the W3C Advisory Committee for final endorsement.
- *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library CD and which provides a comprehensive list of the Genesys and CTI terminology and acronyms used in this document.
- The *Genesys 7 Migration Guide*, also on the Genesys Documentation Library CD, which provides a documented migration strategy from Genesys product releases 5.1 and later to all Genesys 7.x releases. Contact Genesys Technical Support for additional information.
- *Genesys Technical Support Troubleshooting Guide,* which includes information about the GVP log files.
- Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at `http://genesyslab.com/support`.

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

- *Genesys 7 Supported Operating Systems and Databases*
- *Genesys 7 Supported Media Interfaces*

Genesys product documentation is available on the:

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

Making Comments on This Document

If you especially like or dislike anything about this document, please 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 information in this document only and to the way in which the information is presented. Speak to 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.

Chapter

1 About the Voice Application Reporter SDK

This chapter introduces the Voice Application Reporter (VAR) Software Development Kit (SDK) 7.6, its components, features, and scope of use. It contains the following sections:

- Overview, page 13
- Components, page 14
- Scope of Use, page 14
- Architecture, page 14
- API Overview, page 16

Overview

The Voice Application Reporter (VAR) Java Software Development Kit (SDK) enables you to interface with VAR and generate application reports. This SDK internally interacts with a VAR Client that collects data from the SDK and passes it to the VAR Server.

The VAR Java SDK, which is based on the Voice Application Call Model, offers a simple set of interfaces that help the application developer to send call data to VAR Server and view reports. This model reveals the technical details of the underlying VAR Client process. Together, the VAR SDK and the VAR Client form the components of the VAR Client Service.

Notes: This SDK is for the Java or Java Server Page (JSP) implementations only. It does not support Active Server Page (ASP) implementations.

If you intend to use VAR with Genesys Info Mart, you must obtain the Universal Connection ID from the Genesys Voice Platform and/or as an attribute in the interaction with the Genesys IVR Server.

Components

The VAR SDK consists of the following:

- A single `.jar` file—`VARSDK.jar`, which is under the `com.genesyslab.var` package
- The Java documents related to the classes and interfaces, which are defined under this package

The VAR Client consists of the following:

- An executable that runs as a daemon on UNIX, or as a service on Windows
- A set of configuration files that specify the VAR Server name and other runtime parameters
- A set of `.jar` files that form the backbone library of the Client Service.

Scope of Use

The following are the typical usage scenarios in which the SDK class and its methods are called from the VoiceXML application:

- At the time of a call initiation, to initialize communication to the Client Service, and to set some of the environment variables
- As the call progresses, to capture the start and end of an IVR Action, or to set any customer data that is related to the application
- At the time of call termination
- To query the status of a call

Architecture

The VAR Client architecture consists of two components—the Java VAR Client Service and the Java VAR SDK (see Figure 1 on page 15).

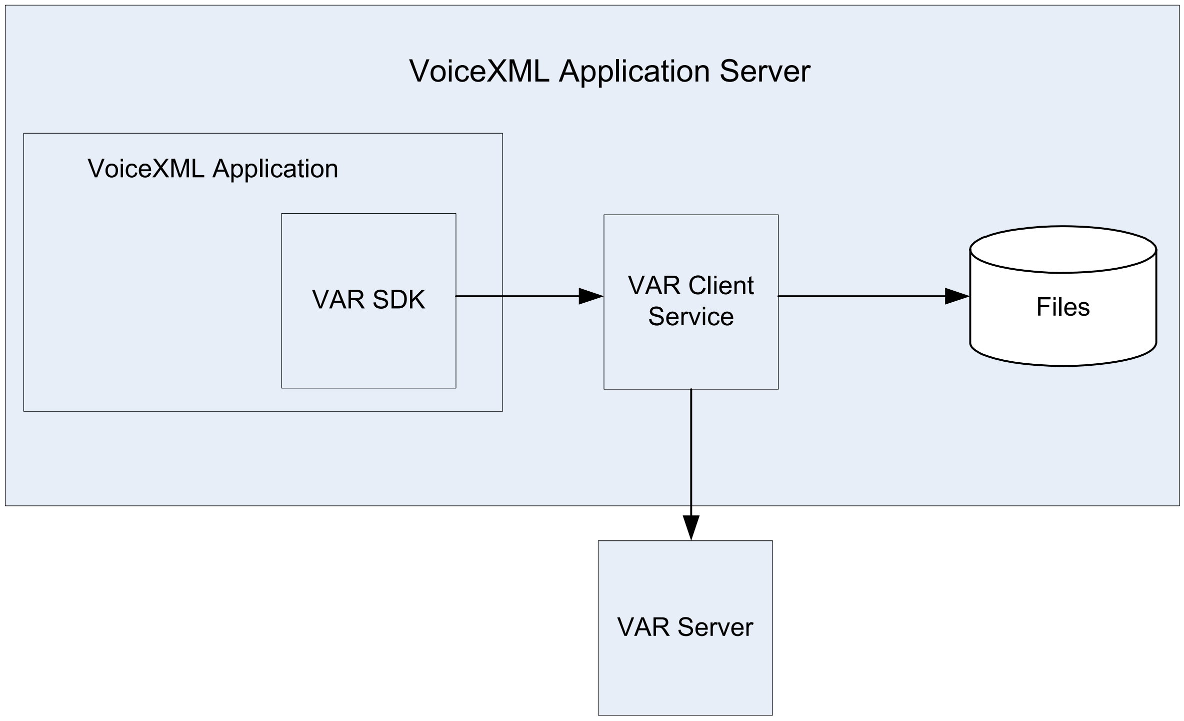

Figure 1: VAR Client Architecture

These components are used to capture and send events as follows:

1. The VoiceXML application is bundled with, and interacts with, the Java VAR SDK. The SDK is implemented as a Java class. This class provides methods that can be called with appropriate parameters in order to initiate VAR events. The VoiceXML application instantiates a session level object of this class during call start. It calls methods of this class at appropriate times.
2. The Java VAR Client Service runs as a service on the VoiceXML application server. It must be up and running all the time, waiting for events to be sent from the VoiceXML application.
3. On any given method call, the VAR SDK validates the input data, as well as the method calling sequence. If it finds errors, the SDK raises Java exceptions, which can be trapped by the calling program. The VAR SDK then synchronously replays the event information to the VAR Client Service. On completion, it returns control to the calling program.
4. When it receives an event, from the VAR SDK, VAR Client stores it immediately in a local file, and returns results (success/failure) to the VAR SDK.
5. At configurable intervals, the VAR Client Service sends the accumulated events to the VAR Server through existing HTTP POST mechanisms. If it is successful in sending the event, the VAR Client removes the events from

the local store. If it is unsuccessful, the VAR Client continues to keep the events on the local store, so that it can try to send them again in the next cycle.

6. Every five minutes, the VAR Client Service logs a status record in its log file. This provides information about the number of events received, the number of files posted and any errors found in the last five minutes. The log files have size control and rollover based on size and days. There is also automatic cleanup of old log files—for example, files that are more than seven days old.

API Overview

The Java SDK package consists of two Java classes:

- `VoiceAppCall`—The main class that models a voice application call and helps the user iniate call events.
- `VARException`—The class that extends the `java.lang.exception` class and supports an additional set of error codes. All runtime exceptions thrown by the `VoiceAppCall` class belong to this type.

VoiceAppCall Class

The typical usage scenario for the `VoiceAppCall` class of the Java SDK depends on the state of the call. Table 1 lists the call states and classes that are called for each.

Table 1: Call States and VoiceAppCall Class Methods

Call State	Method Called
Call Initiation	Constructor
	setTZOffset
	setVARClientPort
	startCall
Call Progress	StartIvrAction
	EndIvrAction
	setCustomData
	setGVPAttributes
Call Termination	endCall

Table 1: Call States and VoiceAppCall Class Methods (Continued)

Call State	Method Called
Call Status	isCallStarted
	isCallEnded
	isCallinProgress
	getlastMethodCalled
	getLastMethodCallList
Call Cleanup	finalize()

Tables 2–16 provide detailed information about how you can use each of the the methods in the `VoiceAppCall` class.

Table 2: Constructor

Purpose	Initializes the `VoiceAppCall` object.
Signature	`public VoiceAppCall (String sessionId, String appID, String appName) throws VARException`
When Called	On the `Start` page of the VoiceXML application, after the `sessionID` for the call is received from the platform, and after `appID` and `appName` are initialized within the application.
Validations Performed	• `sessionID` is 38 characters (after adding `{}`, if required). • `appID` is not empty. • `appName` is not empty.
Actions Performed	• Add `{}` to `sessionID` start and end if they are not in the input. • Discover the time zone offset, based on system settings. • Set the default VAR Client port to `9815`.

Table 2: Constructor (Continued)

Exceptions	Generates an instance of the VARException class with the possible error codes: • `10101` • `10107` (For more information, see Table 18 on page 30.)
Hints for Developer	Initialize this object on the `Start` page of the call, and assign it to the session object.

Table 3: setTZOffset

Purpose	Optionally sets the time zone offset.
Signature	`public void setTZOffset (float tzOffset) throws VARException`
When Called	Optional. Used only if you want to set an explicit time zone offset that is different from the system's time zone.
Validations Performed	• Call is not in progress. • Time zone value is between `-12` and `+14`.
Actions Performed	Set the call's time zone.
Exceptions	• `10102` • `10202` (For more information, see Table 18 on page 30.)
Hints for Developer	Use this method only if you need a different time zone. Call this method before `startCall()`.

Table 4: setVARClientPort

Purpose	Optionally sets the VAR Client port for sending events.
Signature	`public void setVARClientPort (int Port) throws VARException`
When Called	Optional. Used only if the VAR Client is configured to run on a port other than the default port number of 9815.
Validations Performed	• Call is not in progress. • Port number is greater than 128.

Table 4: setVARClientPort (Continued)

Actions Performed	Set the port for the call.
Exceptions	Generates an instance of the VARException class with the possible error codes: • `10103` • `10202` (For more information, see Table 18 on page 30.)
Hints for Developer	Use this method only if the VAR Client is running on a port other than the default port. Call this method before calling `startCall()`.

Table 5: startCall

Purpose	Indicates the start of the call, for reporting purposes.
Signature	`public void startcall (String ANI, String DNIS) throws VARException`
When Called	When the start of the call is supposed to be reported. Call it when the application is ready to answer the call, after it has made the required checks and validations.
Validations Performed	• Call is not already in progress. • `ANI` is not empty. • `DNIS` is not empty.
Actions Performed	• Mark the call as started. • Use the current timestamp as the start call timestamp. • Send the start call event (`01`).
Exceptions	Generates an instance of the VARException class with the possible error codes: • `10107` • `10202` • `10301` • `10302` (For more information, see Table 18 on page 30.)
Hints for Developer	Use the `isCallStarted()` function to validate whether the `startCall()` function has already been called.

Table 6: endCall

Purpose	Indicates the end of the call, for reporting purposes.
Signature	`public void endCall(varEndActions endState, varResults ivrResult, String ivrResultReason, String ivrNote) throws VARException` Possible `varEndAction` values: • `01 = COMPLETED` • `02 = TRANSFERRED` • `03 = ABANDONED` Possible `varResults` values: • `S = SUCCESS` • `F = FAILURE` • `U = UNKNOWN`
When Called	When the end of the call is supposed to be reported. Call it when the IVR is completed, and the call is ready to be disconnected or transferred.
Validations Performed	• Call is already in progress. • `endState` is not empty, and it has valid values. • `ivrResult` is not empty, and it has valid values. • `ivrResultReason` is not empty.
Actions Performed	• Use the current timestamp as the end call timestamp. • Send the custom data event (`20`) if the custom data is set. • Send the call end event (`02`). • Mark the call as ended.

Table 6: endCall (Continued)

Exceptions	Generates an instance of the VARException class with the possible error codes: • `10104` • `10105` • `10107` • `10201` • `10301` • `10302` (For more information, see Table 18 on page 30.)
Hints for Developer	Make sure that the page invoking this method is always called, regardless of any exceptions that are occurring. Calling this method is critical to proper reporting of the call. For example: `myVAR.endCall(varEndStates.TRANSFERRED, varResults.FAILURE, Thread.currentThread().getName(), "dsfdsfd");`

Table 7: startIvrAction

Purpose	Indicates the start of an IVR Action.
Signature	`public void startIvrAction(String actionID, String actionName, String parentActionID) throws VARException`
When Called	Optional. Used when the application needs to track a specific IVR Action.
Validations Performed	• Call is already in progress. • `actionID` is not empty. • `actionName` is not empty. • `parentActionID` is not empty.
Actions Performed	• Use the current timestamp as the start IVR Action timestamp. • Send the start IVR Action event (`03`).

Table 7: startIvrAction (Continued)

Exceptions	Generates an instance of the VARException class with the possible error codes: • `10107` • `10201` • `10301` • `10302` (For more information, see Table 18 on page 30.)
Hints for Developer	In order to obtain valid reporting values, it is essential to trace the IVR Actions that have started, and to ensure that they are closed at the correct times. This is especially true if nested IVR Actions are used. If IVR Actions are not closed by calling `endIvrAction`, VAR automatically closes them; however, the duration and result values will be incorrect.

Table 8: endIvrAction

Purpose	Indicates the end of an IVR Action that has already started.
Signature	`public void endIvrAction (String actionID, String parentActionID, varResult actionResult, String actionReason, String actionNote, boolean isLastAction) throws VARException`
When Called	Optional. Used when the application needs to track a specific IVR Action.
Validations Performed	• Call is already in progress. • `actionID` is not empty. • `parentActionID` is not empty. • `actionResult` is not empty and has valid values. • `actionReason` is not empty.
Actions Performed	• Use the current timestamp as the End IVR Action timestamp. • Send the End IVR Action event (`04`).

Table 8: endIvrAction (Continued)

Exceptions	Generates an instance of the VARException class with the possible error codes: • `10106` • `10107` • `10201` • `10301` • `10302` (For more information, see Table 18 on page 30.)
Hints for Developer	In order to obtain valid reporting values, it is essential to trace the IVR Actions that have started, and to ensure that they are closed at the correct times. This is especially true if nested IVR Actions are used. If IVR Actions are not closed by calling `endIvrAction`, VAR automatically closes them; however, the duration and result values will be incorrect.

Table 9: setCustomData

Purpose	Indicates that the VoiceXML application is to set custom data values.
Signature	`public void setCustomData (String customName, String customValue) throws VARException`
When Called	Optional. Used whenever the application is to set custom name-value pair data. It can be called multiple times. If the same name is set multiple times, the last value is used.
Validations Performed	• Call is already in progress. • `customName` is not empty.
Actions Performed	Check for the name among the custom names that have already been set in the call. If it already exists, change the value, otherwise, add a new name-value pair.

Table 9: setCustomData (Continued)

Exceptions	Generates an instance of the VARException class with the possible error codes: • `10107` • `10201` (For more information, see Table 18 on page 30.)
Hints for Developer	A single event for custom data is initiated before the end event is sent for the call. Even though the VAR SDK does not limit the number of name-value pairs that are set, VAR Server has limits on the number of name-value pairs that are processed.

Table 10: setGVPAttributes

Purpose	Sets GVP-specific attributes.
Signature	`public void setGVPAttributes (String GVPAppId, String FWConnId, String FWUnivConnid, String GVPVoiceServerIP) throws VARException` Where `GVPAppId` is the GVP configured AppID. Where `FWConnId` is the Genesys Framework Connection ID. Where `FWUnivConnID` is the Genesys Framework Universal Connection ID. Where `GVPVoiceServerIP` is the Voice Server IP of the GVP server.
When Called	Optional. Used if the VoiceXML application needs to set GVP data, this method is called once before the `endCall()` function. If it is called multiple times, the last call is used.
Validations Performed	Not Applicable.
Actions Performed	Pass values that are stored in the object.
Exceptions	Generates an instance of the VARException class with the possible error codes: • `10201` (For more information, see Table 18 on page 30.).
Hints for Developer	The information set in this method is passed to VAR Server through the IVR Call End event.

Table 11: isCallStarted

Purpose	Determines whether the call has been started using the `startCall()` function.
Signature	`public boolean isCallStarted()`
When Called	Optional. Used if the VoiceXML application needs to validate that the `startCall()` function has been called. Can be used for troubleshooting.
Validations Performed	Not Applicable.
Actions Performed	Return `true` if `startCall()` has been called successfully without exceptions; otherwise, return `false.`
Exceptions	Not Applicable.
Hints for Developer	Use this method for troubleshooting or error handling.

Table 12: isCallEnded

Purpose	Determines whether the call has been started using the `startEnd()` function.
Signature	`public boolean isCallEnded()`
When Called	Optional. Used if the VoiceXML application needs to validate that the `endCall()` function has been called. Can be used for troubleshooting.
Validations Performed	Not Applicable.
Actions Performed	Return `true` if `endCall()` has been called successfully, without exceptions; otherwise, return `false`.
Exceptions	Not Applicable.
Hints for Developer	Use this method for troubleshooting or error handling.

Table 13: isCallinProgress

Purpose	Determines whether the call is in progress. In progress means that `startCall()` has been called, but `callEnd()` has not.
Signature	`public boolean isCallinProgress()`
When Called	Optional. Used if the VoiceXML application needs to validate that the call is in progress.
Validations Performed	Not Applicable.
Actions Performed	Return `true` if the `startCall()` function has been called successfully, without exceptions, and if the `endCall()` function has not been called; otherwise, return `false`.
Exceptions	Not Applicable.
Hints for Developer	Use this method for troubleshooting or error handling.

Table 14: getLastMethodCalled

Purpose	Return information about the last method that the VoiceXML application called. This does not include status methods.
Signature	`public String getLastMethodCalled()`
When Called	Optional. Used if the VoiceXML application needs to capture, analyze or log the last method that was called.
Validations Performed	Not Applicable.
Actions Performed	Return a string in the following format: `<Date in format YYYMMDDHHMISS>:<method name>(Parameter list with values pased, separated by "=")` For example: `20060522221633:22:startCall(ANI=409299221,DNIS=234332232)`

Table 14: getLastMethodCalled (Continued)

Exceptions	Not Applicable.
Hints for Developer	Use this method to obtain information about what the last call was and from where in the application it was called. You can analyze this string to decipher more information.

Table 15: getMethodCallList

Purpose	Returns information about all method calls made to the `VoiceAppCall` object during an IVR call.
Signature	`public String getMethodCallList()`
When Called	Optional. Used if the VoiceXML application needs to capture, analyze, or log the list of method calls that were made.
Validations Performed	Not Applicable.

Table 15: getMethodCallList (Continued)

Actions Performed	Return a string with multiple lines, each corresponding to a method call or an exception, in chronological order.
	Each method call line is in the same format as the string returned by `getLastMethodCalled()`.
	Each exception line is in the following format:
	`EXCEPTION : <error code> : <error description>`
	`For example :`
	`20060522221633 : File -> TestVARSDK.java : Line -> 17 : <init>( callID = A81C586F-0FDD-F85C-F8B0-B88D2917AD05, appid = firstapp, appName = First Application )`
	`20060522221633 : File -> TestVARSDK.java : Line -> 20 : setTZOffset ( tzOffset = -26.0 )`
	`EXCEPTION : 10100 : Invalid value for tzOffset : '-26.0'. tzOffset should be between -12 and +14`
	`20060522221633 : File -> TestVARSDK.java : Line -> 21 : setVARClientPort ( tzOffset = 9813 )`
	`20060522221633 : File -> TestVARSDK.java : Line -> 22 : startCall( ANI = 409299221, DNIS = 234332232 )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 32 : startIvrAction( actionID = AccountBalance, actionName = Account Balance, parentActionID = 0 )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 33 : endIvrAction( actionID = AccountBalance, parentActionID = 0, actionResult = S, actionReason = Balance Found, actionNote = value 15,000, isLastAction = false )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 35 : setCustomData( customName = Customer, customValue = Kumaran )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 36 : setCustomData( customName = Company, customValue = Genesys )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 37 : setCustomData( customName = Customer, customValue = KumaranP )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 39 : setGVPAttributes( GVPAppID = 10012, FWConnID = 2342332, FWUnivConnID = sfdsfds, GVPVoiceServerIP = 12.1.22.1 )`
	`20060522221634 : File -> TestVARSDK.java : Line -> 40 : endCall( endState = 01, ivrResult = F, ivrResultReason = myreason, ivrNote = dsfdsfd )`

Table 15: getMethodCallList (Continued)

Exceptions	Not Applicable.
Hints for Developer	Use this method to obtain information about what the last call was, and from where in the application it was called.

Table 16: finalize

Purpose	Cleans the `VoiceAppCall` instance.
Signature	`finalize()`
When Called	Automatically during garbage collection.
Validations Performed	Not applicable.
Actions Performed	Validate that the call has ended.
Exceptions	Not Applicable.
Hints for Developer	Make sure that the `VoiceAppCall` object is released after the call, and that garbage collection occurs frequently.

VARException Class

The `VARException` class is an extension of the `java.lang.Exception` class. It has been added in order to enable the capture of error codes by using the `getLastErrorCode()` method. Table 17 provides detailed information about how you can use this method.

Table 17: getLastErrorCode

Purpose	Returns an error code for the generated exception.
Signature	`public String getLastErrorCode()`
When Called	Optional. Used if the VoiceXML application needs to capture, analyze, or log the error code for the last generated exception.
Validations Performed	Not Applicable.

Table 17: getLastErrorCode (Continued)

Actions Performed	Return the last error code as a string.
Hints for Developer	You can use the error code to troubleshoot exceptions that the VAR SDK encounters.

Table 18 lists the error codes raised by the `VoiceAppCall` class.

Table 18: Error Codes

Error Code	Error Message
10101	Invalid value for callID: <callID>.callID should be a GUID of length 38.
10102	Invalid value for tzOffset:<tzOffset>.tzOffset should be between -12 and +14.
10103	Invalid value for Port:<Port>.Port should be greater than 128.
10104	Invalid value for endState:<endstate>.Valid values are 01,02, and 03.
10105	Invalid value for ivrResult:<ivrResult>.Valid values are S, F, and U.
10106	Invalid value for actionResult:<actionResult>.Valid values are S, F and U.
10107	Invalid value for <paramName>.It cannot be an empty string.
10201	This call is not started or has already been completed. You can call <functionCall> only when the call is in progress.
10202	This call is already in progress. You can call <functionCall> only when the call has not started.
10301	Cannot open connection to VAR Client on Port:<Port>.
10302	Error sending VAR Event Data to VAR Client:<Error message>.

Chapter

2 About the Code Samples

This chapter introduces the code samples that accompany this *Developer's Guide*. It presents essential design considerations, and also some of the initial tasks that you must perform in order to use each kind of library. It contains the following sections:

- Using the Code Samples, page 31
- VAR Code Samples, page 32

Using the Code Samples

In order to develop applications with the Voice Application Reporter (VAR) Software Development Kit (SDK), you need a compiler, such as the compiler delivered in the Java 2 Standard Edition (J2SE) SDK. It must conform to release 1.4.2 or 1.5.

In this document, JDK 1.4.2 from Sun Microsystems was used to compile and run the code samples.

Before you can use the samples, you must do the following:

1. Install the VAR SDK Library. For more information, see *Genesys Voice Platform 7.6 Voice Application Reporter Deployment and Reference Manual.*
2. Set up the following environment variables:
   - `CLASSPATH:` In this environment variable, specify the `varsdk.jar` files.
   - `JAVA_HOME:` In this environment variable, specify the location of the Java Runtime Environment.

VAR Code Samples

This section describes how to use the SDK to report on application events.

Call Start Activities

The `Start` page of the VoiceXML application must include Call Start Activities. This action initializes the VAR SDK and sets the key parameters that are needed for a given call. An instance of the `VoiceAppCall` class must be maintained for each call, and it must correspond to a web session that covers multiple VoiceXML page visits for that call.

To use Call Start Activities:

1. Import the VAR SDK, using the following code snippet:

```
<%@ page import="com.genesyslab.var.*" %>.
```

2. Make sure that the `varsdk.jar` files are in the `Java CLASSPATH` environment variable.
3. Collect the correct VAR Client port from the configuration, and the call's Session ID, ANI, and DNIS from the VoiceXML session variables, and then initialize a new instance of the `VoiceAppCall` object and capture any error exceptions, by using the following code snippet:

```
try
{
   appCall = new VoiceAppCall(
                 sessionId,
                 appId,
                 appName
                 );
}
catch(VARException e)
{
   //Handle the exception using the VARException class
   System.out.println("Error while initiating VoiceAppCall " +
         e.getMessage());
}
```

4. If the VAR Client is not listening on port 9815, you can set the port number and capture any error exceptions by using the following code snippet:

```
try
{
     appCall.setVARClientPort(portInt);
}
```

```
catch(VARException e)
{
   //Handle the exception using the VARException class
   System.out.println("Error while setting VAR Client Port " +
            e.getMessage());
}
```

5. If you need to report on the time zone, use the following code snippet:

```
try
{
   appCall.setTZOffset((float)-6)
}
catch(VARException e)
{
   //Handle the exception using the VARException class
   System.out.println("Error while setting VAR Client Time zone"
            + e.getMessage());
}
```

6. Start the call by using the following code snippet:

```
try
{
   appCall.startCall(ani, dnis);
}
catch(VARException e)
{
   //Handle the exception using the VARException class
   System.out.println("Error while Starting Call " +
            e.getMessage());
}
```

Call End Activities

Call End Activities are executed when a call is completed. You must make sure that your code is always executed:

- At the end of the call, regardless of the call result.
- At the end of call processing (to ensure that the timestamps are accurate).

To use Call End Activities:

1. Retrieve the `appCall` object that is stored in the session, and copy it to the local `appCall` variable.

2. Set the values for the call end state (using the `varEndStates` enumerator), the call result (using the `varCallResults` enumerator), and the call result reason, by using the following code snippet:

```
try
{
appCall.endCall(endState,callResult,callResultReason,callNote);
}
catch(VARException e)
{
//Handle the exception using the VARException class
System.out.println("Error while Completing Call"+e.getMessage());
}
```

Start IVR Action Activities

The Start IVR Action Activities are executed when you logically start an IVR Action. Every IVR Action needs an Action ID, Action Name, and a Parent Action ID.

Note: If the IVR Action has no parents, `ivrParentActionID` will be set to `-1`.

To use Start IVR Action Activities:

1. Retrieve the `appCall` object that is stored in the session, and copy it to the local `appCall` variable.
2. Initialize the values for the IVR Action ID, Action Name and Parent Action ID by using the following code snippet:

```
try
{
appCall.startIVRAction
(ivrActionID,ivrActionName,ivrParentActionID);
}
catch(VARException e)
{
//Handle the exception using the VARException class
System.out.println
("Error while starting IVRAction"+e.getMessage());
}
```

End IVR Action Activities

The End IVR Action activities are executed when you logically end an IVR Action. The IVR Action ID and Parent Action ID must match those that are set in the Start IVR Action.

To use End IVR Action Activities:

1. Retrieve the `appCall` object that is stored in the session, and copy it to the local `appCall` variable.
2. Initiate the values for the IVR Action ID, the Parent Action ID, the IVR Action End State (using `varEndStates` enumerator), the IVR Action Result (using `varCallResults` enumerator), and the IVR Action Result Reason by using the following code snippet:

```
try
{
appCall.endIVRAction(ivrActionID,ivrParentActionID,
ivrActionEndState,ivrActionResult,ivrActionResultReason,
ivrActionNotes,isLastAction);
}
catch(VARException e)
{
//Handle the exception using the VARException class
System.out.println("Error while completing IVR Action"+
e.getMessage());
}
```

Note: If the IVR phase of the call is completed while the call context is within the IVR Action (that is, the start action has been called, but the end action has not been called), you must set `isLastAction` to `true`; otherwise, set `isLastAction` to `false`.

Set Custom Data

You can capture custom call information at any time during the life cycle of the call. The maximum number of custom name-value pairs that can be collected is determined by the VAR Server configuration. Genesys recommends setting it to eight. Also, if the same custom name is set multiple times, the last value that was set will be reported.

To use Set Custom Data:

1. Retrieve the `appCall` object that is stored in the session, and copy it to the local `appCall` variable.

2. Call the `setCustomData()` method, by using the following code snippet:

```
try
{
appCall.setCustomData(customName, customValue);
}
catch(VARException e)
{
//Handle the exception using the VARException class
System.out.println("Error while setting custom data"+
e.getMessage());
}
```

Index

A

API . . . 16
  call states . . . 16
  methods . . . 16
  varexception class . . . 29
  voiceappcall class . . . 16
architecture . . . 14
audience
  defining . . . 6

C

call end activities . . . 33
call start activities . . . 32
call states . . . 16
chapter summaries
  defining . . . 8
class
  varexception . . . 29
  voiceappcall . . . 16
client . . . 14
code samples . . . 31, 32
  call end activities . . . 33
  call start activities . . . 32
  end ivr action activities . . . 35
  installing . . . 31
  set custom data . . . 35
  start ivr action activities . . . 34
commenting on this document . . . 11
components . . . 14
constructor . . . 17

D

document
  conventions . . . 8
  errors, commenting on . . . 11
  version number . . . 8

E

end ivr action activities . . . 35
endcall . . . 20
endivraction . . . 22
error codes . . . 30

F

finalize . . . 29

G

getlasterrorcode . . . 29
getlastmethodcalled . . . 26
getmethodcalllist . . . 27

I

installing code samples . . . 31
iscallended . . . 25
iscallinprogress . . . 26
iscallstarted . . . 25

J

java SDK
  varexception . . . 16
  voiceappcall . . . 16

M

methods . . . 16

S

SDK . . . 13
  API . . . 16

architecture 14
code samples 31
components 14
overview 13
scope 14
set custom data 35
setcustomdata 23
setgvpattributes 24
settzoffset 18
setvarclientport 18
start ivr action activities 34
startcall 19
startivraction 21

T

typographical styles 8

U

use
SDK 14

V

VAR client 14
varexception class
errorcodes 30
getlasterrorcode 29
version numbering
document 8
voice application reporter SDK 13
voiceappcall class
constructor. 17
endcall 20
endivraction 22
finalize 29
getlastmethodcalled 26
getmethodcalllist. 27
iscallended 25
iscallinprogress 26
iscallstarted 25
setcustomdata 23
setgvpattributes 24
settzoffset 18
setvarclientport 18
startcall 19
startivraction 21